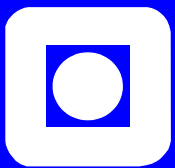


*Nils Kr. Rossing og Thor Inge Hansen*

**Kurshefte:  
Miljøovervåking med bøy  
Arduino MKR NB 1500**



NTNU



Trondheim

Institutt for fysikk

Skolelaboratoriet

for matematikk, naturfag  
og teknologi

Oktober 2023



# Kurshefte: Miljøovervåking med bøyе, Arduino MKR NB 1500

Nils Kr. Rossing,  
Skolelaboratoriet NTNU, Institutt for fysikk  
Thor Inge Hansen  
Skien videregående skole  
i samarbeid med Institutt for marin teknikk og  
Institutt for lærerutdanning

## Kurshefte: Miljøovervåking med bøye, Arduino MKR NB 1500

Trondheim 2023

Layout og redigering: Nils Kr. Rossing, Skolelaboratoriet ved NTNU

Trykk: NTNU Grafisk senter

Tekst og bilder: Nils Kr. Rossing, Skolelaboratoriet, NTNU,  
Jussi Evertsen, Vitenskapsmuseet, NTNU

Forsidebilde: Internett: Arduino MKR NB 1500

Programmering: Nils Kr. Rossing, NTNU  
Thor Inge Hansen, Skien videregående skole

Kursholdere: Nils Kr. Rossing  
(Elektronikk/Progr.) Thor Inge Hansen, Skien videregående skole  
Johannes Ravn Munkvold (læringsassistent)  
Skolelaboratoriet, Institutt for fysikk, NTNU

Faglige spørsmål rettes til:

**Skolelaboratoriet for matematikk, naturfag og teknologi**

**Institutt for fysikk**

v/ Nils Kr. Rossing, [nils.rossing@ntnu.no](mailto:nils.rossing@ntnu.no)

Skolelaboratoriet ved NTNU

Realfagbygget,  
Høgskoleringen 5,  
7491 Trondheim

Telefon: 73 55 11 91

<https://www.ntnu.no/skolelab/>

Rev 1.7 – 31.10.23

Prosjektet støttes av:



Institutt for marin teknikk



Institutt for lærerutdanning



Skolelaboratoriet





---

## Forord

Prosjektet “Miljøovervåking med bøye” handler om å samle inn måledata fra kystnære strøk eller i ferskvann eller elver i innlandet. Teknologien kan imidlertid brukes i mange ulike sammenhenger, i områder der man normalt ikke har tilgang til lokale nettverk slik at man kan overføre data via Wi-Fi. For å få til dette tar man i bruk ulike typer smalbånds mobildata som Telenor med flere tilbyr innen rammen av 4 og 5G.

Våren 2020 kom det en henvendelse fra Håvard Holm ved Institutt for marin teknikk som hadde en ide om å la elever bygge havgående bøyer med billige materialer og utstyre dem med elektronikk som kunne samle inn måledata fra havet og overføre disse til en server på land. Det ble i den forbindelse utført en mulighetsstudie ved Skolelaboratoriet som ble videreført av to studenter samme sommer.

Flere teleselskaper tilbyr smalbånds dataoverføring via det som går under navnet NB IoT. Telenor<sup>1</sup>, Com4<sup>2</sup> og Telia<sup>3</sup> er eksempler på slike aktører. Likeså er det flere som tilbyr skyløsninger for lagring og presentasjon av data f.eks. Arduino (IoT) Cloud<sup>4</sup>, Circus of Things<sup>5</sup>, Wappsto<sup>6</sup>, Streamlit<sup>7</sup> med mange flere. Felles for alle disse løsningene er at de krever pålogging, noe som kan være problematisk å tillate for skoleeiere både i grunnskole og videregående skole som gjerne krever pålogging via FEIDE og garanti for at regler for sikkerhet oppfylles. Inntil det kommer en slik sikker skytjeneste på plass, har vi valgt å legge dataene på en server ved Inst. for marin teknikk ved NTNU, for så å bruke Excel eller egenutviklet Python programmer for presentasjon og analyse av dataene.

Hefte beskriver gangen fra ide og til man har etablert en forbindelse mellom målesensorer til dataene kan lastes ned fra serveren teknikk og presenteres i Excel, Google Earth eller ved hjelp av egenutviklede Python script, noe som mange elever selv kan skrive.

Anvendelsen av denne type teknologiske løsninger er svært anvendelige. Tidligere er det kommet forespørsler fra Institutt for kjemisk prosesseteknologi som ønsker å gjøre målinger på temperatur, ledningsevne og oksygeninnhold m.m. i myr over lengre tid, f.eks. over en sommersesong. En ser også for seg at en slik datainnsamling er interessant i forbindelse med undervisningsopplegg i fag som Teknologi og forskningslære eller fag ved elektro yrkesfag. Det er derfor utviklet et eget kretskort ved Skien videregående skole som kan benyttes av de som ikke selv ønsker å utvikle all elektronikken.

Heftet inneholder også en gjennomgang av en rekke sensorer for registrering av parametere i vann som temperatur, pH, reduksjon-oksidasjons-potensial, ledningsevne/salinitet, turbiditet og oksygeninnhold. I tillegg er det laget et forslag til laboratoriejournal som skal være en hjelp til å

---

1. <https://www.telenor.no/bedrift/iot/>

2. <https://www.com4.no/>

3. <https://business.teliacompany.com/internet-of-things>

4. <https://cloud.arduino.cc/>

5. <https://www.circusofthings.com/>

6. <https://wappsto.com> (for micro:bit)

7. <https://streamlit.io/>



---

gjennomføre kalibrering og måling med en rekke sensorer. Journalen er ment for kursets 2. samling som inkluderer laboratoriearbeid med aktuelle sensorer.

En takk til student Johannes Ravn Munkvold som har lest korrektur og kommentert tidligere utgaver av heftet samt deltatt som veileder under flere kurs. Takk også til Jussi Evertsen Vitenskapsmuseet NTNU som har kommet med råd til framstilling av bøya og vært behjelpelig med å få bøya på vannet, videre ved å teste utvalgte sensorer.

Thor Inge Hansen ved Skien videregående skole har utviklet kretskortløsningen og bidratt til å løse problemer knyttet til sendinger som ikke kommer fram til serveren. Videre har han utviklet programvare i Python for presentasjon og analyse av måledata. Han har dessuten førstehånds kjennskap til hva som fungerer og ikke fungerer i klasserommet.

Skolelaboratoriet ved NTNU,  
Oktober 2023  
Nils Kr. Rossing  
Thor Inge Hansen



## Innhold

<b>1 Innledning .....</b>	<b>13</b>
1.1 Designprosessen – en didaktisk tilnærming .....	13
1.2 En spennende historie .....	16
1.3 Havbøye – Et eksempel .....	19
1.3.1 Test 1 – Sjøvannstest .....	19
1.3.2 Test 2 – Sjøvannstest .....	22
1.3.3 Test 3 - Landtest .....	33
<b>2 Prosjektet – Oppgavesamlingen .....</b>	<b>37</b>
2.1 Deloppgdrag 1: Installasjon av programvare .....	38
2.2 Deloppgdrag 2: Oppkobling av hårdvare og montering av SIM-kort .....	38
2.3 Deloppgdrag 3: Installer biblioteker og kjør eksempelprogrammet .....	39
2.4 Deloppgdrag 4: Monter sensoren for måling av vanntemperaturen .....	40
2.5 Deloppgdrag 5: Monter GPS-kortet og inkluder GPS data .....	41
2.6 Deloppgdrag 6: Oppkobling til GPRS-nettet og nedlasting til serveren .....	42
2.7 Deloppgdrag 7: Inkluder “vakt hund” .....	44
2.8 Deloppgdrag 8: Skriv dataene til SD-kort .....	45
2.9 Deloppgdrag 9: Presentasjon av måleresultatene .....	46
2.10 Deloppgdrag 10: Oppkobling av PLSC MKR-kretskort .....	47
<b>3 Programmeringen .....</b>	<b>49</b>
3.1 Installasjon av programvare .....	49
3.1.1 Arduino programeditor, IDE .....	49
3.2 Installasjon av ekstra pakke for programmering av MKR NB 1500 .....	51
3.3 Installasjon av bibliotek for programmering av MKR NB 1500 .....	53
3.4 IOT-teknologier (4G) og anskaffelse av SIM-kort .....	53
3.4.1 Bestilling av et antall fra 1 – 9 .....	55
3.4.2 Bestilling av et antall fra 10 eller flere .....	55
3.5 Montering av antenne og SIM-kort .....	56
3.6 Installasjon av biblioteker og kjøring av eksempelprogrammet .....	56
3.6.1 Installasjon av biblioteker .....	57
3.6.2 Installasjon av eksempelprogrammet (EVU-kurs23-Eksempelprogram.ino) .....	58
3.7 Programkoden (EVU-kurs23-Eksempelprogram.ino) .....	59
3.7.1 Detaljert gjennomgang av programkoden .....	59
3.7.2 Bygg opp tekststreng for overføring av en måleserie .....	70
3.8 Montering av PLSC MKR-kortet .....	71
<b>4 Teknologien .....</b>	<b>77</b>



4.1	Internet of Things .....	77
4.1.1	Definisjon og nytteverdi .....	77
4.1.2	Litt historie .....	78
4.1.3	Fordeler og ulemper med IoT .....	79
4.1.4	Internasjonal standard for IoT .....	80
4.2	Valg av teknologi .....	80
4.2.1	Dataoverføringskanaler .....	81
4.3	Valg av microcontroller-teknologi .....	82
4.3.1	Micro:bit – Wappsto:bit fra det danske firmaet Seluxit .....	82
4.3.2	Raspberry Pi Pico + SIM7080G .....	86
4.3.3	Arduino MKR NB 1500 .....	90
4.3.4	Vårt valg av teknologi – En begrunnelse .....	91
4.4	Arduino .....	92
4.4.1	MKR NB 1500 .....	92
4.5	Arduino shield-kort .....	95
4.5.1	MKR ENV Shield .....	96
4.5.2	MKR GPS Shield .....	103
4.5.3	MKR MEM Shield .....	111
4.5.4	MKR SD Proto Shield .....	111
4.5.5	Prototyp-kort .....	112
4.6	Energiøkonomisering og bruk av dvale .....	112
4.6.1	Vekking etter en bestemt tid .....	113
4.6.2	Vekking med ekstern trigger koblet til en port .....	114
4.6.3	Strømforbruk i dvaletilstand .....	116
4.6.4	Bruk av ekstern lav energi timer .....	117
4.6.5	Opplasting av programmer til mikrokontroller som er i deep sleep .....	117
4.7	Energikilder .....	118
4.7.1	Bruk av power bank .....	118
4.7.2	Bruk av dvale og Li-Po batteri .....	118
4.7.3	Bruk av Li-Po batterier av typen 18650 .....	119
4.8	Ladere .....	121
4.8.1	Enkel og billig USB-lader .....	121
4.8.2	Lading fra USB og/eller solceller .....	122
4.9	Bruk av power booster .....	123
4.9.1	Adafruit Powerboost 1000 Basic .....	123
4.9.2	Enkel og billig booster fra Aliexpress .....	125
4.10	Bruk av “vakthund” – Watchdog .....	125
4.10.1	Programvare basert “vakthund” .....	125



4.10.2	Bruk av PICAXE-08M2 som ekstern vakthund .....	126
4.10.3	Programvare og uttesting av “vakthunden” .....	128
4.10.4	Bruk av Sseed Studio XIAO SAMD21 .....	128
4.11	Datalogger med lader, power booster og ekstern “vakthund” .....	129
4.11.1	Forslag til systembeskrivelse .....	129
4.11.2	Virkemåte .....	130
4.11.3	Kretskort .....	133
<b>5</b>	<b>Presentasjon og behandling av måledata .....</b>	<b>134</b>
5.1	Skrijving til og henting av data fra server .....	134
5.1.1	Skrijving til fil på serveren .....	134
5.1.2	Lesing av data fra fil på serveren .....	134
5.2	Skrive data til fil .....	135
5.2.1	Lagre rådata .....	135
5.2.2	Tidsangivelse .....	135
5.2.3	Skilletegn .....	135
5.2.4	Simulerte data .....	135
5.3	Bruk av Excel for visualisering av data .....	137
5.3.1	Importer data fra en tekst-fil til Excel .....	137
5.3.2	Lage grafer .....	140
5.4	Bruk av Python for å presentere og analysere dataene .....	141
5.4.1	Installasjon og oppstart med Python .....	141
5.4.2	Organisering av data i filen .....	142
5.4.3	Lesing og presentasjon av data fra file .....	143
5.4.4	Analyse av data med Python .....	148
5.5	Bruk av Google Earth for visning av posisjon fra GPS .....	150
5.5.1	Plotting av en enkeltposisjon .....	150
5.5.2	Plotting av en trase i Google Earth .....	151
5.5.3	Editering av kml-fila .....	154
<b>6</b>	<b>Sensorer .....</b>	<b>155</b>
6.1	Aktuelle målinger i havet langs kysten .....	155
6.1.1	Bøya ved Munkholmen .....	156
6.1.2	Bøya i Vestfjorden .....	157
6.1.3	Meteorologisk institutt .....	158
6.2	Organisering av oppkobling .....	159
6.3	Temperaturmåling .....	160
6.3.1	Temperaturfølsom motstand (NTC- og PTC-motstander) .....	160
6.3.2	NTC-motstanden .....	160
6.3.3	Bruk av DS18B20 .....	167



6.4	Sensor for måling av oppløst oksygen i vann (SEN0237-A) .....	170
6.5	Sensor for måling av vannets ledningsevne og saltholdighet (DFR0300-H) ....	181
6.6	Sensor for måling av turbiditet (SEN-0189) .....	193
6.7	Sensor for måling av pH (SEN-0161) .....	202
6.8	Sensor for måling av ORP (SEN-0165) .....	211
6.9	Sensor for måling av vanntrykk (SEN-0257) .....	215
<b>7</b>	<b>Referanser .....</b>	<b>221</b>
<b>Vedlegg A</b>	<b>Komponentliste .....</b>	<b>222</b>
<b>Vedlegg B</b>	<b>Kretsskjema og PCB-layout av kretskort .....</b>	<b>225</b>
<b>Vedlegg C</b>	<b>Feilfinning og problemer .....</b>	<b>227</b>
C.1	Problemer med opplasting av programmet .....	227
C.2	Problemer med opplasting av programmet pga dvale .....	227
C.3	Problemer med å lese verdier fra MKR GPS sammen med MKR ENV .....	228
C.4	Problemer med resetting av programmet .....	229
C.5	Programmet stopper å samle inn data .....	230
C.6	Problemer med å lese riktig lufttrykk fra ENV-kort når GPS er tilkoblet .....	231
C.7	Programmet klarer ikke å skrive til monitoren etter reset .....	232
C.8	Programmet henger seg opp og kommer ikke videre .....	233
C.9	Forsøk på oppkobling mot server synes å utebli .....	233
C.10	Programmet klarer ikke å restarte etter dvale .....	234
<b>Vedlegg D</b>	<b>Sensor laboratoriejournal .....</b>	<b>235</b>
D.1	Måling av temperatur i vann med NTC .....	236
D.2	Måling av temperatur i vann med DS18B20 .....	238
D.3	Måling av oppløst oksygen i vann (SEN0237-A) .....	240
D.4	Måling av vannets ledningsevne og saltholdighet (DFR0300-H) .....	245
D.5	Måling av turbiditet i vann (SEN-0189) .....	251
D.6	Måling av pH (SEN-0161) .....	255
D.7	Måling av ORP (SEN-0165) .....	260
D.8	Måling av trykk og beregning av dybde (SEN-0257) .....	263
<b>Vedlegg E</b>	<b>Omregning fra ledningsevne til saltholdighet .....</b>	<b>266</b>
E.1	Ledningsevne som funksjon av saltholdighet, temperatur og trykk .....	266
<b>Vedlegg F</b>	<b>Python-programmer for presentasjon av måleresultater .....</b>	<b>267</b>
F.1	Grunnleggende program for presentasjon av data .....	267
<b>Vedlegg G</b>	<b>Programmer for bruk av sensorer .....</b>	<b>271</b>
G.1	Måling av temperatur med DS18B20 .....	271
G.2	Kalibrering og måling av sensor for oppløst oksygen SEN0237-A .....	271



---

G.3	Kalibrering og måling av sensor for måling av salinitet DFR0300-H .....	275
G.4	Kalibrering og måling Turbiditet SEN-0189 .....	277
G.5	Kalibrering og måling med pH-sensoren SEN-0161 .....	279
G.6	Kalibrering og måling med ORP-sensoren SEN-0165 .....	282
G.7	Kalibrering og måling av trykk og dybde med SEN-0257 .....	287
<b>Vedlegg H</b>	<b>Testprogrammer for MKR NB 1500 .....</b>	<b>289</b>
H.1	Testprogram for å legge i dyp søvn og vekkes fra en port .....	289
H.2	Testprogram for å legge i dyp søvn og vekkes av intern “time out” .....	290
<b>Vedlegg I</b>	<b>Testprogrammer for shield-kort .....</b>	<b>292</b>
I.1	Testprogram for MKR ENV .....	292
I.2	Testprogram for MKR ENV skriving til SD-kort .....	293
I.3	Testprogram for MKR GPS .....	296
I.4	Testprogram for MKR ENV og GPS sammen .....	297
I.5	Testprogram for kalibrering av temperatursensoren – NTC .....	299
I.6	Testprogram for testing av temperatursensoren DS18B20 .....	301
<b>Vedlegg J</b>	<b>Eksempelprogram for EVU-kurset .....</b>	<b>303</b>
J.1	Gjennomgående eksempelprogram (EVU-kurs23-Eksempelprogram.ino) .....	303
<b>Vedlegg K</b>	<b>Operativt program med ekstern “vakt hund” (Ferdig_program_med_PICAXE_MILLIS_TEST_2) .....</b>	<b>312</b>







# 1 Innledning

Heftet er en oppsummering av hvordan man etablerer en sensornode ved bruk av smalbånd mobil-data via 4G som kommunikasjonskanal. Dette gir oss et langt større dekningsområde enn Wi-Fi, som er avhengig av lokale nettverk. Vi har valgt å bruke Arduino MKR NB 1500 og basere oss på SIM-kort fra Telenor. Vi har inntil videre valgt kommunikasjonskonseptet NB IoT som synes å tilfredsstillere våre behov i en uttestingsfase med en sensornode ombord på en havgående bøye. Inntil videre legger vi dataene ned på en lokal server ved Institutt for marin teknikk og visualiserer av dataene ved hjelp av Python skript eller Excel og Google Earth. Prismessig er dette tilbudet også gunstig med et etableringsgebyr på kr. 15,- inkludert 1 MB og deretter kr. 8,- for 1 MB/mnd. pluss kr. 1,49 pr. MB ut over de første 1 MB<sup>8</sup>. Det er imidlertid ugunstig siden en må administrere SIM-kort lisenser.

Vi har så langt ønsket å gjøre det så enkelt som mulig slik at man kan komme fort igang. Vi har derfor inntil videre valgt å legge dataene i CSV-filer på serveren. Det finnes imidlertid flere alternative skytjenester for lagring og visualisering av data, men ingen tilfredsstillende foreløpig de krav som kommuner og fylkeskommuner stiller mht sikkerhet for sine elever. Firmaet Company of Things arbeider imidlertid med et lovende konsept skreddersydd for bruk i skolen. De ser for seg at det så smått kan tas i bruk i løpet av 2024.

Vi har holdt flere etterutdanningskurs, og nå også videreutdanningskurs. Kursdeltakerne får en ferdig utstyrspakke med SIM-kort, dernest får de hjelp til å bygge opp hardware, komme seg på nett og hente ut data. I første omgang kan man sende over dummy-data før man får anskaffet og koblet til aktuelle sensorer. Tanken er at elevene selv bygger opp bøyene med sensornoder under veiledning av lærer, og utstyrene bøyene i henhold til egenutviklede prosjekter for å dekke lokale behov og interesser.

Heftet er organisert på følgende måte: **Innledningen** gir en oversikt over prosjektet og tips til hvordan det kan gjennomføres. Beskrivelse av en eksempelbøye kan evt. brukes for motivasjon og for å vise hvordan bøya kan bygges. Dernest følger **Prosjektet – oppgavesamlingen** som gir en kortfattet hjelp til å strukturere arbeidet med elektronikk og programmering. For utdypende kommentarer og hjelp til gjennomføringen av prosjektet henvises til kapitlene **Programmeringen** og **Teknologien**. Kapitlet **Presentasjon av data** viser hvordan bruk av Excel, Python og Google Earth kan brukes til å behandle og presentere innhentede data. **Sensor**-kapitlet gir en grundig innføring i noen aktuelle sensorer for måling i vann, hvordan de fungerer, kalibreres og kobles opp. Dette kapitlet vil bli særdeles viktig når vi skal fordype oss i sensorer under samling 2. **Vedleggene** inneholder oversikt over komponentene og gir flere programeksempler.

## 1.1 Designprosessen – en didaktisk tilnærming

Det er viktig å presisere at det er elevene som i størst mulig grad skal designe og teste ut ulike bøyekonstruksjoner. Det antas at det er lettere for elevene å være kreative mht. valg av utforming og materialer ved bygging av bøya enn ved utvikling og bygging av elektronikken. Innføringen i

---

8. E-post fra Telenor v/Jesper Lade 23.03.22



elektronikk og programmering er derfor mer oppskriftspreget enn bygging av selve bøya, men med rikelige muligheter for å avvike og gå sine egne veier, spesielt mht. valg av sensorer) etter hvert som de blir tryggere på teknologien.

Uansett er det greit å forholde seg til en strukturert prosess når det gjelder et prosjekt av denne størrelsen. Her er et forslag:

## 1. Motivasjon

Som en motivasjon kan man se den tyske dokumentaren “The North Drift”. Traileren finnes på følgende side: <https://thenorthdrift.com/>, og filmen finnes i sin helhet på YouTube: [https://www.youtube.com/watch?v=\\_2SpKYkOkKQ](https://www.youtube.com/watch?v=_2SpKYkOkKQ)

En diskusjon om temaet filmen tar opp er en naturlig oppfølger og en god innledning til prosjektet: “Miljøovervåking med bøye”. Filmen er på 45 minutter og er dels på tysk og dels på engelsk. Automatisk simultanoversettelse til norsk kan velges på YouTube. Se også omtalen her avsnitt 1.2 på side 16.

## 2. Marinbiologi og målinger

Som bakgrunn for valg av sensorer er det naturlig å ta en runde innom marinbiologi. Som en introduksjon kan man benytte Jussi Evertsen sin presentasjon som finnes på <https://www.ntnu.no/skolelab/bla-hefteserie> under overskriften “Programmering av Tingenes internett (IoT)” og fanen “EVU-kurs: Miljøovervåking med havbøye” og punktet “Miljøbiologi (pdf)”<sup>9</sup>. I avsnitt 1.3 på side 19 i dette heftet finner man dessuten en oversikt over noen aktuelle målinger. Dette kan være et godt grunnlag for senere å bestemme hvilke målinger man skal gjøre og dernest hvilke sensorer man skal velge. Det er lurt å begynne med få målinger for så å bygge ut etter hvert.

## 3. Systemoversikt

På dette tidspunktet kan det være greit å få en viss oversikt over systemet og noen av problemstillingene man vil møte. Dette er primært en oppgave for læreren som i neste omgang kan videreformidle det han/hun mener er viktig å vite for studentene. Ta gjerne utgangspunkt i eksempelet som er beskrevet i avsnitt 1.3 på side 19.

## 4. Valg av sensorer

Siden det kan ta tid å få tak i sensorene man trenger, bør man ikke vente for lenge med å velge hvilke målinger man ønsker å gjøre. Et utvalg sensorer er beskrevet i kapittel 6 side 155. En oversikt over leverandører og priser finnes i vedlegg A side 222. Man må være klar over at de sensorene som er beskrevet er beregnet for laboratoriebruk og ikke industriell bruk, det er derfor ikke sikkert at de vil fungere tilfredsstillende over lengre tid i sjøen.

## 5. Valg av teknologi

I dag er det mange leverandører av mikrokontroller-utviklingskort som inkluderer kommunikasjon via NB IoT. Vi har valgt å se på en Arduinobasert løsning, en Micro:bit-basert løsning og en løsning som baserer seg på Raspberry Pi Pico. Her står elever selvfølgelig fritt

---

9. <https://www.ntnu.no/documents/2004699/1313956950/havb%C3%B8yer+marinbiologi+og+oseanografi+feb+2023.pdf/ea8fc566-205b-836d-c396-3a8627200d11?t=1676559843931>



til å velge, men vi har valgt å gå videre med Arduino MKR NB 1500. De ulike teknologiene er beskrevet i avsnitt 4.2 side 80 og 4.3.

## 6. Bestilling av SIM-kort

Vi har valgt å bruke SIM-kort beregnet for NB IoT fra Telenor. Disse kan bestilles fra Telenor, se avsnitt 3.4 side 53, men undersøk også om Inst. for marin teknikk tilbyr slike kort kostnadsfritt som et tilbud til de som ønsker å delta i OSR.

## 7. Bestilling av komponenter

Ved påmelding til konkurransen OSR, undersøk om Institutt for marin teknikk tilbyr startpakker med komponenter. Dersom det ikke er tilfelle eller man ønsker tilleggskomponenter se gjennom komponentlistene i vedlegg A side 222. Her finnes også oversikt over noen sentrale sensorer.

## 8. Forståelse av grunnleggende programmering av Arduino

Velger man å bruke Arduino så har vi basert oss på tekstbasert programmering (C++). For å komme i gang må man installere verktøyet. Dette er beskrevet i avsnitt 3.1.

Selv om programmet er relativt enkelt, så er det ikke opplagt for nybegynnere. Det anbefales derfor å gjennomføre grunnleggende programmering av Arduino før man går igang med prosjektet. Her finnes det mye å velge i, se f.eks. referanse [5], [6] og [7].

Opplæringen tar utgangspunkt i noen få eksempelprogrammer som består av en rekke funksjoner. I løpet av kursdagen bringes stadig nye funksjoner inn i programmet for uttesting. På den måten bygges programmet gradvis opp sammen med forståelsen.

## 9. Valg av energikilder, energiøkonomisering og energihøsting

Det er ønskelig at bøyene skal foreta målinger over lengre perioder, uker eller måneder. I så fall blir energitilførselen kritisk. En må derfor, eksempelvis, ta stilling til type batteri, bruk av dvale eller lading ved hjelp av solceller. Som en første tilnærming kan man bruke batterier som er operative til de er tomme. Strategier for energisparing og energihøsting er omtalt i avsnitt 4.6 side 112 til avsnitt 4.7 side 118.

## 10. Bruk av sensorer

I utgangspunktet leveres kretsen kun for måling av vanntemperatur og GPS. Det er naturlig at man i tillegg velger noen andre sensorer som f.eks. lufttemperatur, lufttrykk og en eller flere sensorer for måling av vannparametere. I så fall må man sette seg inn i bruken av de aktuelle sensorene. Sensorene er beskrevet i detalj i kapittel 6 side 155. Det er lagt ved programvare for å teste ut, kalibrere og lese av sensorene i vedlegg G side 271.

## 11. Innhenting og analyse av måledata

I vårt tilfelle legges måledataene, som nevnt, i en file på en server hos Inst. for marin teknikk. Hver måling består av en linje hvor måleverdiene er skilt med komma. Etter hvert som flere målinger gjøres, legges det nye linjer til filen. Dette er en enkel, men nokså primitiv måte å håndtere data på. Det er derfor naturlig at dataene bearbeides av programvare for visualisering av resultatene. Dette kan gjøres ved hjelp av Excel, Google Earth eller egenutviklede programmer i Python, evt. andre alternativer. Hvordan dette gjøres er beskrevet i kapittel 5



side 134. Hvordan man kan utvikle egne Python-programmer for presentasjon og analyse er beskrevet i avsnitt 5.4 side 141.

## 12. Bygging av bøya

Her er det lettere for elevene å bruke sine kreative evner. Det er mange måter å lykkes på, men også mange måter å mislykkes på, så en viss rettleiding er på sin plass. Det viktigste er å sørge for at bøya flyter, for at elektronikken holdes tørr og tåler litt røft vær. Det er også å foretrekke at det legges vekt på servicevennlighet. Læreren må vurdere i hvilken grad elevene skal få tips om utforming eller ikke. Eksempelbøya i avsnitt 1.3 side 19 kan være en hjelp for læreren.

## 13. Dokumentasjon

Det anbefales på det sterkeste at elevene fører en laboratoriejournal, da dette er en hjelp til å strukturere arbeidet. Journalen kan gjerne inneholde bilder, grafer og for den saksskyld små videoopptak. Det viktigste er at de får hjelp til å gjøre valg på bakgrunn av erfaringer og en støtte når den endelige rapporten skal skrives, som også er et krav dersom man skal delta i OSR.

### 1.2 En spennende historie

For noen år siden gjestet den tyske filmskaperen *Steffen Kronen* fra Dresten, Norge. I løpet av turen besøkte han bl.a. en liten øy i Lofoten. Han ble slått av hvor mye søppel som hadde samlet seg på strendene, og overraskelsen ble stor da han fant en tysk ølflaske blant alt søppelet. Kom den virkelig helt fra Tyskland, eller var det en tysk turist som hadde slengt den fra seg?

Hjemme i Dresden klarer han ikke å bli kvitt tanken på at søppel som havner i elva Elben så langt sør som til Dresden, kan tenkes å havne nord for polarsirkelen.

Steffen bestemmer seg for at dette må han finne ut av. Sammen med sin gode venn og nabo *Paul Weiss* begynner de å bygge havgående bøyer med en innebygget GPS-mottaker og en satellittradio slik at bøyesens posisjon kan følges fra time til time. Bøyene ble blant annet bygget av plastavfall og utstyrt med en FleetMon S1-C GPS Tracker som vist under<sup>10</sup>.

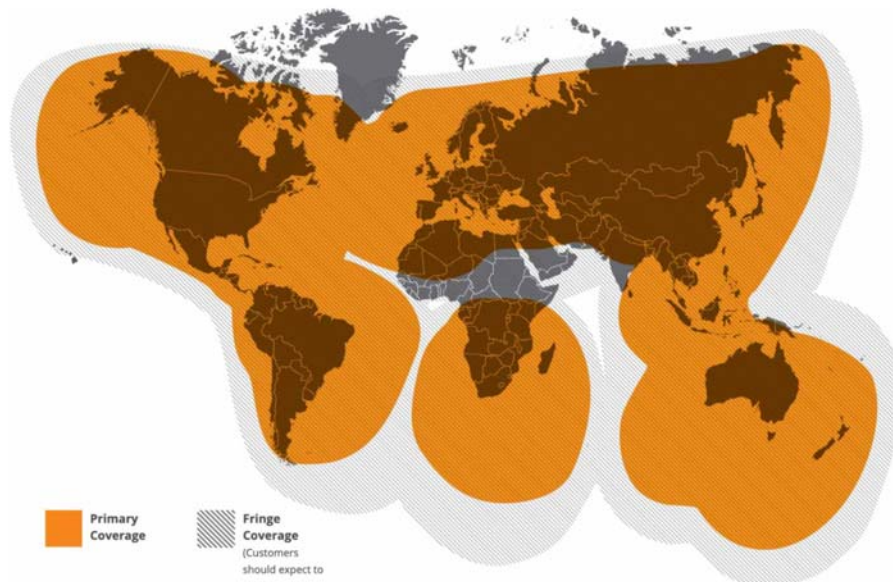


Trackeren har en vekt ca. 100 g og kan utstyres med 4 x AAA 1,5 V batterier som gir en brukstid på mer enn 1,5 år under normale forhold. Brukstiden er imidlertid sterkt avhengig av hvor ofte data sendes til satellittene. Enheten kan i tillegg forsynes med ekstern strømkilde. Trackeren kan

10. [https://static.fleetmon.com/static/downloads/FleetMon\\_S1-C\\_Satellite\\_Transponder.pdf](https://static.fleetmon.com/static/downloads/FleetMon_S1-C_Satellite_Transponder.pdf)



stilles inn for å overføre data hver halve time opp til en gang i døgnet, og benytter *Global LEO Satellittene* og *Globalstar Simplex Data Network* for overføring av data. Dekningsgraden er nærmest global når en ser bort fra polområdene som mangler dekning av dette systemet, se kartet under<sup>11</sup>. Det er verdt å merke seg at dekningsområdet til våre bøyer vil bli langt mindre og kun omfatter landområdene og kystnære strøk der det er dekning med 4G.



Bøyene var enkle, men bygget slik at de skulle kunne følge Elben ut til Nordsjøen og evt. videre ut på havet. Etter slipp av et titalls bøyer, fant de at flere endte opp langs norskekysten. Ett par landet blant annet utenfor Trøndelag og til og med så langt som til Lofoten<sup>12</sup>. Bøyene overførte posisjonen sin hver 4. time.



Simulering av bøya laget av resirkulert plast og kork



Bøya stikker opp av havet



En bøye havnet i Lofoten

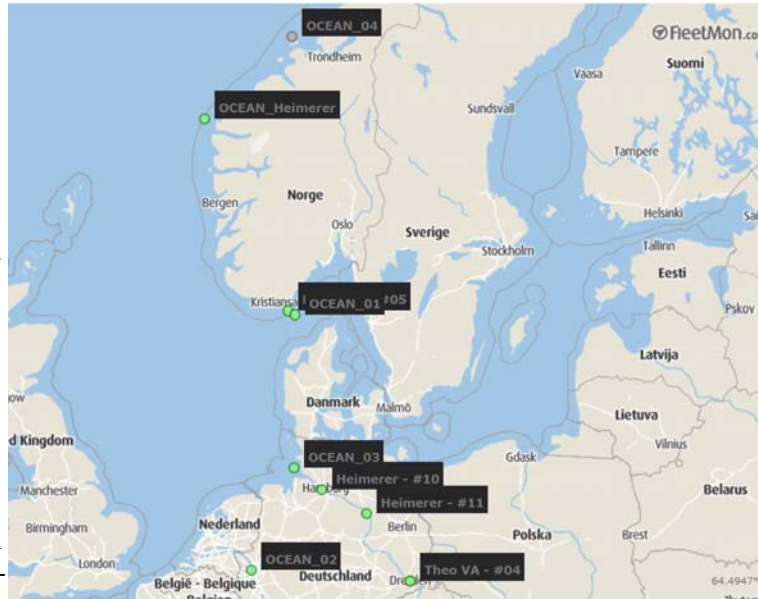
11. <https://www.fleetmon.com/services/satellite-tracker/>

12. Bildene er hentet fra <https://www.mdr.de/tv/programm/sendung-712450.html>





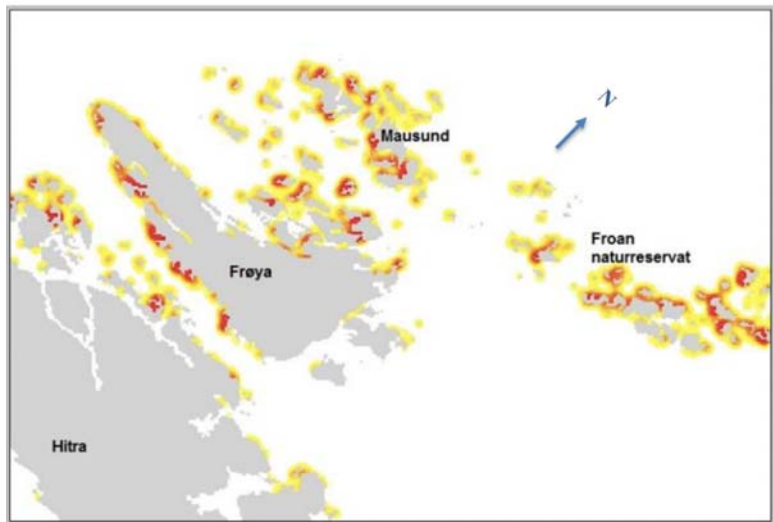
Under arbeidet med filmen søkte Steffen Krones kontakt med anerkjente tyske forskere som **Dr. Melanie Bergmann** og **Dr. Lars Gutow** ved **Alfred Wegener Institute** og fikk hjelp til å forstå hva som skjer med søppel som havner i de tyske elvene og etter hvert ender i havet. Han fikk også kontakt med den norske inuiten **Kris** som kjenner norskekysten godt etter utallige turer i kajakk. Tre år jobbet han med filmen som ble lansert våren 2022. Filmen ligger i sin helhet på YouTube<sup>13</sup>.



Kartet over viser posisjonen til noen av bøyene på et gitt tidspunkt under ferden. En av årsakene til at historien er gjengitt her, er at et par av bøyene ble sporet til øyene utenfor Mausundvær.

I følge en simulering gjort av Havforskningsinstituttet i 2017, er området rundt Mausundvær spesielt utsatt for søppel drevet inn fra havet. De gule og røde markeringene på kartet er spesielt utsatte. Figuren er lånt fra heftet “Miljøovervåking i tareskogen” av **Hilde Ervik** [1].

Selv om våre bøyer kommer til å befinne seg i kystnære strøk, er



13. [https://www.youtube.com/watch?v=\\_2SpKYkQkKQ](https://www.youtube.com/watch?v=_2SpKYkQkKQ)



det mange ting man kan tenke seg å måle. I det neste avsnittet skal vi konkretisere hvordan en bøye kan se ut. Senere skal vi også gi eksempler på aktuelle målinger (se avsnitt 6.1 side 155).

### 1.3 Havbøye – Et eksempel<sup>14</sup>

Eksempelet under er kun en hjelp til læreren som skal veilede elevene, slik at tiden til forberedelse kan reduseres. Egne erfaringer gjennom bygging og uttesting er uansett nyttig, dette gjelder så vel elever som lærer.

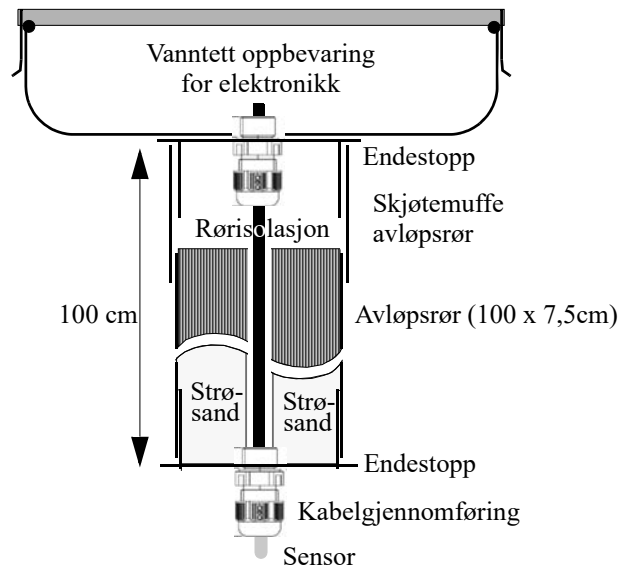
#### 1.3.1 Test 1 – Sjøvannstest

##### Utforming av bøya

I dette eksempelet skal vi bruk avløpsrør med tilhørende koblingstykker<sup>15</sup>. Elektronikken plasseres vi i en vanntett boks på toppen av røret. Dette kan være en instrumentboks fra ELFA eller en boks beregnet på frysevarer fra Biltema. Ledninger kan evt. føres gjennom røret og ned til bunnen gjennom et elektriskerrør. Vi bruker “vanntette” gjennomføringer for å føre ledninger og sensorer gjennom bunnen til bøya og ut i vannet. Vi brukte litiumfett for å gjøre gjennomføringene tett. Det er ikke gitt at dette er nok.

Avløpsrøret fylles med sand i bunnen slik at røret flyter stående i vannet.

Sand har normalt en egenvekt på 1,4 – 1,5 avhengig av type sand. Volleyball-sand er den med minst partikler og har en tetthet på 1,5. Øverste del av røret fylles med isolasjonsmateriale for rør. Dette er for å unngå at røret synker selv om røret fylles med vann. Det finnes isolasjonsrør som passer akkurat inne i det 75 mm avløpsrøret. Bunnen av avløpsrøret tettes med en endestopp og i toppen brukes en muffe og en endestopp.



14.Råd gitt av Odd Arne Arnesen – Mausund Feltstasjon

15.Ideen stammer fra Håvard Holm IMT, NTNU



En kabelgjennomføring brukes til å koble sammen boks og endestopp i toppen. Dermed oppnår man både en vanntett gjennomføring av kabelen og at boks og endestopp henger sammen. Gjenestykket hos kabelgjennomføringene kan være litt marginale. Det er derfor lite plass til pakninger.



Bildene under viser detaljer fra første forsøk med bøya.



Nederste del av bøya er fylt med strøsand (ca. 50 cm). Øverste del er fylt med isolasjonmateriale for rør, slik at man skal unngå at den synker om det slippes inn vann. Bøya er forberedt både for feste i bunnen og i toppen. Sistnevnte kan f.eks. brukes til å feste bøya til en blåse.



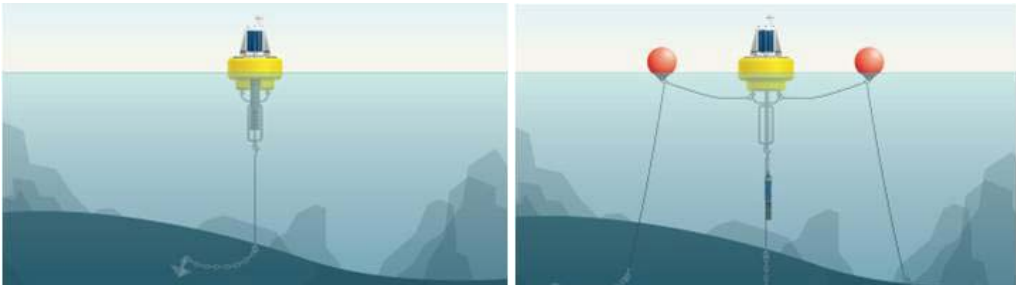


## Oppankring av bøye

En eventuell oppankring må meldes til Sjøfartsdirektoratet med posisjon dersom den plasseres i åpent farvann. Oppankrede bøyer må dessuten utstyres med lys slik at bøyen er godt synlig for trafikk i all slags vær. Det er også påkrevd at den merkes med navn og kontaktdata.

Det er viktig å montere elektronikken og antennene så høyt som praktisk mulig over vannflata, jo høyere, jo bedre signal, spesielt i urolig hav. En løsning kan være å gjenbruke gamle bøyer, elektronikken kan i så fall monteres på toppen av bøylene i egen boks. Bøyene kommer i alle størrelser. Jo mindre bøyen er, jo større utfordringer vil man kunne få i grov sjø.

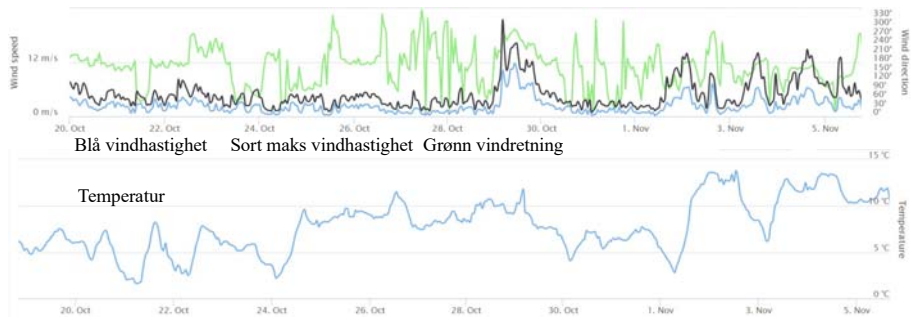
Elektronikken kan evt. bygges inn i en egen vanntett boks, (ikke integrert i bøya) som vil gjøre håndteringen av bøyer og elektronikk enklere. Dessuten vil en slik boks kunne gi god plass til batterier. En kan også vurdere bruk av solceller. Det er også viktig å sette av plass til montering av antenner, lys og radarreflektor. Når elektronikken er montert på denne måten, kan den lett demonteres uten å løfte hele bøya på land. Dette er spesielt viktig dersom målinger skal skje over lang tid slik at det kan være nødvendig med batteriskifte eller reparasjoner.



På figuren over er vist en-punkt og to-punkts oppankring. Det er viktig at bøya ikke blir dratt under ved flo og høy sjø, samtidig som den ikke har for slakk line slik at den driver av.

## Erfaringer – Sjøvannstest 1

Bøya uten elektronikk ble lagt ut ved Trondheim Biologiske Stasjon (i Trondheimsfjorden ved Trolla) fredag 21. oktober og tatt inn fredag 4. nov., se bildene under. Etter 14 dager i havet så er det ikke observert fuktighet hverken inne i avløpsrøret som befant seg under vann, eller i instrumentboksen som var montert på toppen av avløpsrøret. Været i perioden var godt og det var svært lite nedbør (< 0,2 mm/døgn) og lite vind, dermed har ikke bøya fått prøvd seg (<https://trondheimhavn.no/vaerstasjon/>).





Bildet under viser bøya plassert i vannet utenfor Trondheim Biologiske Stasjon.



Foto: Jussi Evertsen



Foto: Nils Kr. Rossing



Foto: Nils Kr. Rossing

### 1.3.2 Test 2 – Sjøvannstest

#### Utforming og bestykning for sjøvannstest 2

I denne testen ønsker vi å ha hovedfokus på lekkasje og groing. Derfor er det også interessant å måle temperatur og fuktighet inne i boksen.

Bøya skal være utstyrt på følgende måte:

- Mikrokontroller for innsamling av måleresultater og kommunikasjon med server
- GPS-mottaker for å bestemme lokasjonen til bøya og se om GPS-antenne fungerte slik den var plassert (opp ned inne i boksen).
- En enkel temperatursensor i bunnen av bøya
- Et utvendig sensorkammer med ledning og fiktiv sensor, for gi erfaring med montering og innhenting av måledata på denne måten.

#### Design av elektronikken

Til dette formålet valgte vi en oppkoblingen med mikrokontrolleren påmontert et “miljø shield” (MKR ENV) og en GPS-mottaker. En ekstern mikrokontroller (Picaxe) fungerte som en “vakt-hund” som holder rede på tiden mellom målingene. Utenom måleperioden slår den av kretsen helt ved å slå av en power booster. Oppkoblingen får spenning fra tre Li-Po-batterier.

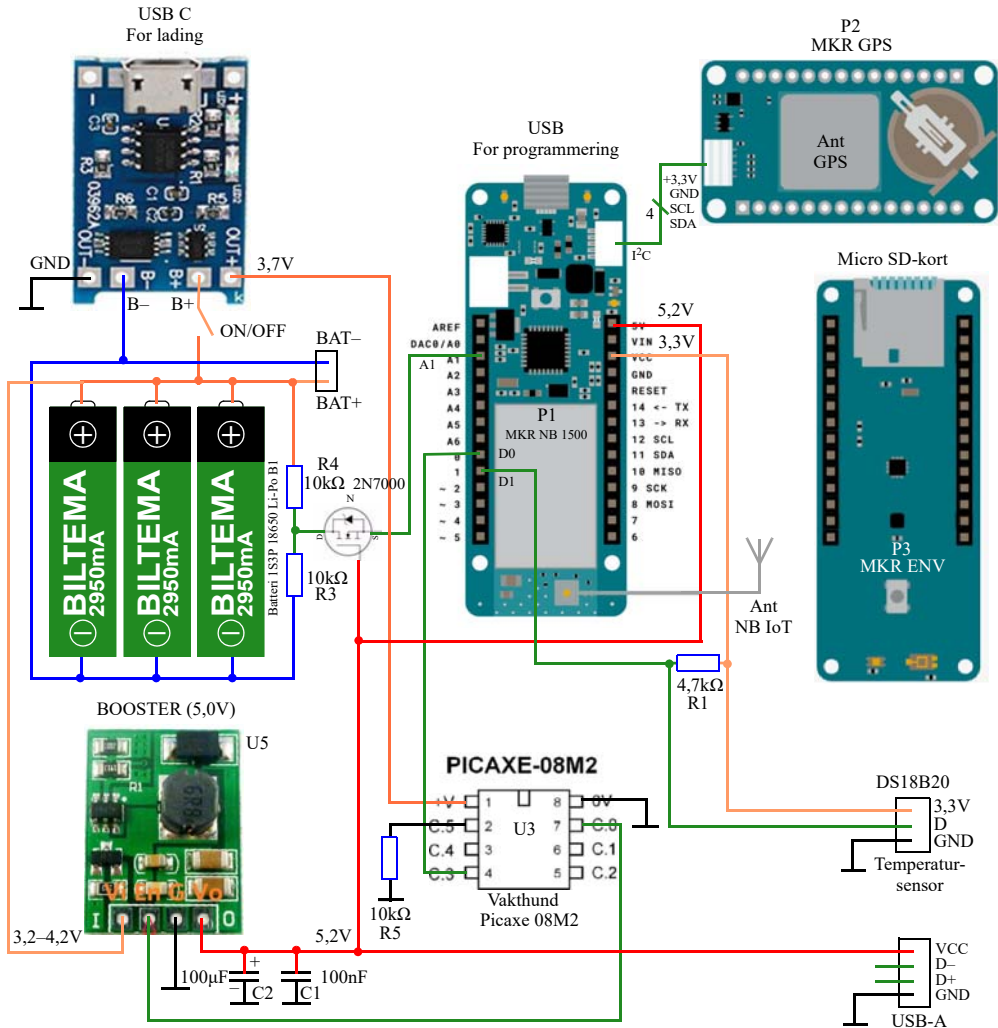
Vi velger følgende parametere:

- Måling ca. hvert 6. minutt (middelverdien for intervallet mellom målingene er 5.7 min)
- Måling av sjøtemperatur og GPS-koordinater



- Måling av batterispenning
- Måling av lysstyrke, lufttemperatur og luftfuktighet (inne i boksen)
- Total batterikapasitet ca. 8 – 9000 mAh
- Velger å la den eksterne “vaktkunden” stå tilkoblet batteriet hele tiden. Strømtrekket fra “vaktkunden” er ca. 0,6 mA.

For nærmere beskrivelse av kretsen se avsnitt 4.11 side 129.





## Utvendig sensorkammer

Ett av flere forslag er å montere sensorer av typen pH, konduktivitet, oksygeninnhold og turbiditet i utvendige sensorkammer, også bygget opp av deler til avløpsrør som vist på figuren under.



Det er 3D-printet en brakett (hvit) som stripses fast til selve bøya. Det er viktig at denne er under vann hele tiden. Både fordi sensoren skal måle i (sjø)vann og for å hindre at bølger slår mot huset slik at det løsner. En kan se for seg at hele braketten deler seg i to. Den kan gjøres sterkere ved å:

- La stripsene gå om hverandre
- Skrive ut huset med 100% fylling
- Legge en tape eller en lang strips rundt både sensorhuset og selve bøyekroppen.

En tuss tjener som “sensor” inntil videre. Ledningen kommer ut av en nippel på toppen og går opp til elektronikken i boksen på toppen av bøya. Det er avgjørende at sensorhuset er vanntett. Det er en utfordring. Tiltak for å gjøre den tettere:

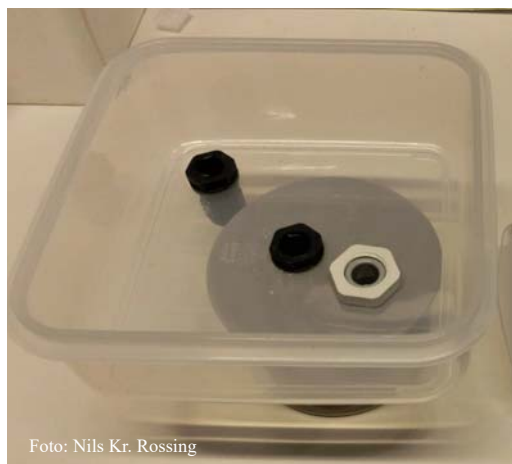
- Det brukes kabelgjennomføringer for å føre ledninger inn og ut av boksen. Disse smøres med litiumfett.
- Det brukes silikonpakninger mellom kabelgjennomføringene og endestykkene. Disse smøres inn med litiumfett
- Pakningene på innsiden av rørmuffene smøres også med litiumfett, både på innsiden og på utsiden





## Instrumentboks

Mikroprosessen, GPS, batteripakke og øvrig elektronikk er foreløpig plassert i en plast matboks på toppen av bøya. Elektronikken er festet i lokket. Dette gjør at elektronikken er litt beskyttet mot fuktighet. Dessuten frigjøres bunnen av boksen til gjennomføring av kabler som vist på bildet under. Her brukes et kretskort<sup>16</sup> som inneholder lader for batteriene, spenningsboosteren for å heve spenningen til 5,2 V og “vaktunden” som holder rede på når målingene skal gjennomføres. I tillegg holder kortet microkontrolleren MKR NB 1500 med et ENV-shield-kort. Vi ser også GPS-mottakeren nederst til høyre på bildet under. Temperatursensoren DS18B20 kobles til kretskortet på tre koblingsplinter og føres ned til bunnen av avløpsrøret.



Det er flere kritiske momenter her:

- Innslag av fuktighet, noe som meget vel kan skje
- 4G-antenne får kummerlige betingelser nær batteriene.
- Likeså GPS-antennen som er plassert ned mot vannet og ikke opp mot himmelen.

---

16. Utviklet og fremstilt av Thor Inge Hansen ved Skien vgs.





## Lekkasjetest

Å gjøre lekkasjetester før utplassert er helt avgjørende, spesielt for deler som skal senkes ned under vann. Figurene under viser hvordan sensorhuset ble dyppet ca. 80 cm ned i en sylinder for å se om det lekket.



Foto: Nils Kr. Rossing

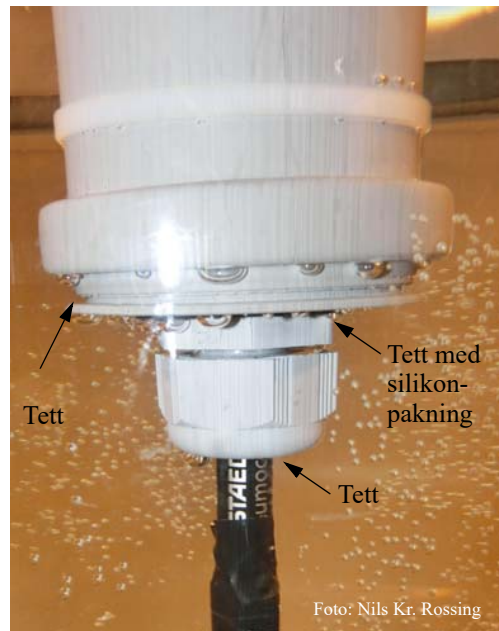


Foto: Nils Kr. Rossing

Et lodd henges under for å holde det under vann. En relativt effektiv måte å teste tetthet på er å blåse luft inn i kammeret og se hvor lufta tyter ut. Det viste seg at ledningen ikke var gjennomgående tett slik at det var mulig å blåse luft inn i kammeret. Vi oppdaget da at pakningen rundt kabelgjennomføringen var uttett. Da denne ble byttet med en silikonpakning og smurt inn med litiumfett så ble den tett.

Videre viste seg at det sivet luft ut mellom muffa og endestykkene til tross for at disse var innsmurt med fett. Årsaken var at pakningene ikke var smurt med fett på baksiden. Da vi tok ut hele pakningen og smurte den på alle sider, så det ut til at muffa med endestykker ble tett.

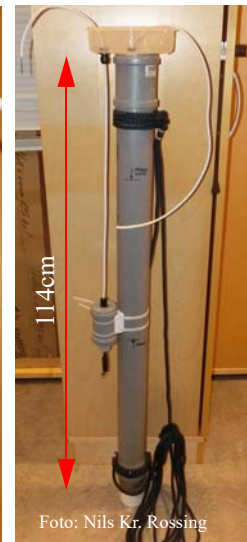
En lignende test ble gjort med selve bøya med temperatursensor montert i enden. Med litiumfett smurt på alle sider av pakningen og rundt sensoren, så det ut til at også denne ble vanntett.





## Sammenstilling av sensorkammer og bøye

Tilslutt monterte vi sensorkammeret på siden av bøya med lange strips og trakk ledningen opp til kabelgjennomføringen under instrumentboksen som vi monterte på toppen av bøya med to kabelgjennomføringer gjennom et endestykke. I tillegg er det montert en kabelgjennomføring i bunnen av instrumentboksen hvor ledningen kommer opp fra sensorkammeret.



## Gjennomføring av sjøvannstest 2

Bøya ble satt ut 07:00 på morgenen torsdag 8. juni utenfor brygga til Trondheim biologiske stasjon i Trolla, ca. 5 km vest for Trondheim sentrum. Det var stille vær og meldt rolig og varmt i dagene fremover.

Bøya synes å flyte noe lavt og burde nok ha vært hevet noe for å unngå at større bølger slår over instrumentboksen. For sikkerhetsskyld hadde vi lagt en plastpose over boksen festet med en tape rundt røret som vist på bildet til høyre.

Bøya var som sist forankret i bunnen med et drivanker, med et tau opp til gelenderet på brygga og et tau bort til trappa for sideveis stabilisering.

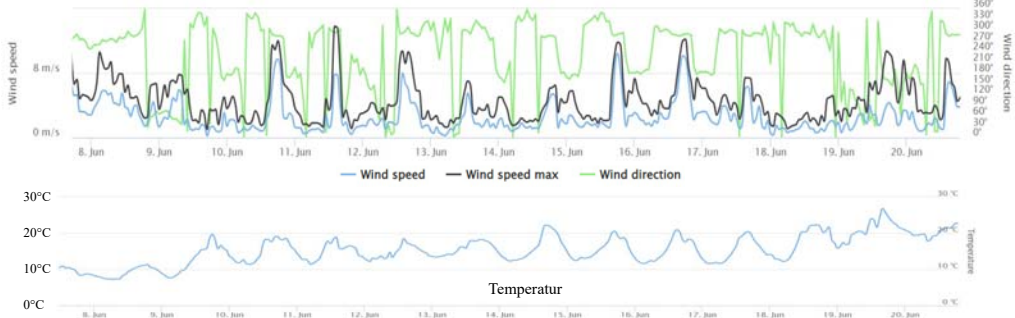
Hensikten var hovedsakelig å undersøke om boksen var tett nok for vårt formål og om sensorkammeret på siden av bøya tok inn vann.



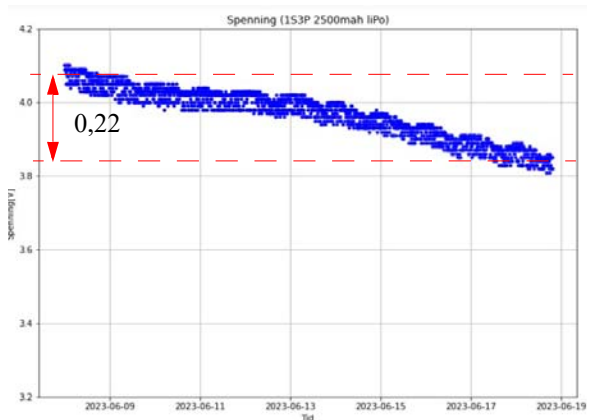


## Erferinger og observasjon etter sjøvannstest 2 (etter ti døgn i sjøen)

Været i perioden bøya har stått ute fra kl. 07:00 08.06.23 til 20:15 19.06.23 har vært det aller beste med lite vind og oppholdsvær som vist på figuren under (<https://trondheimhavn.no/vaerstasjon/>).



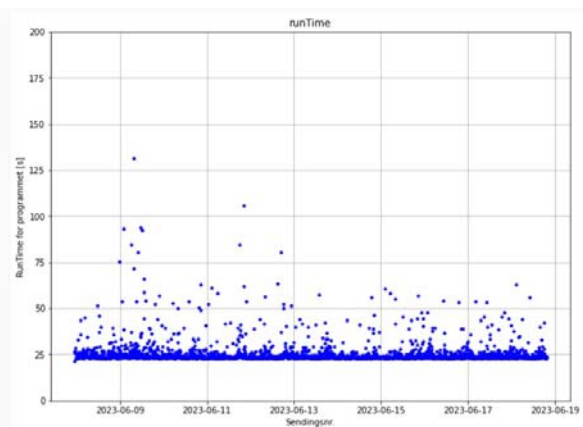
Etter 2732 målinger med en gjennomsnittlig på-tid på 25,4 sekunder, har batterispenningen falt med ca. 0,22V. Den totale på-tiden er målt til 19,3timer. Det ser også ut til at fallet i batterispenningen flater noe ut før det igjen faller. Vi registrerer at det fortsatt er støy på spenningen selv etter filtrering med kondensatorer.



Hver måleperiode har i gjennomsnitt vært på 25,4sekunder. Med en gjennomsnittlig strømstyrke på 50mA pr. måleperiode, vil dette utgjøre ca.

0,96Ah. I tillegg kommer strømtrekket fra “vakthunden” som er på 0,6mA og utgjør ca. 0,06 Ah og som alltid er påkoblet. Batterikapasiteten er ideelt sett på 8850mAh.

Figuren til høyre viser målesondens på-tid for hver måling. Vi legger merke til at hovedtyngden ligger på rundt 25 sekunder, med en del målinger opp mot 50 sekunder. Noen befinner seg også i området 50 – 75 sekunder, mens noen få tar lengre tid. I gjennomsnitt 25,4 sek. som nevnt foran.

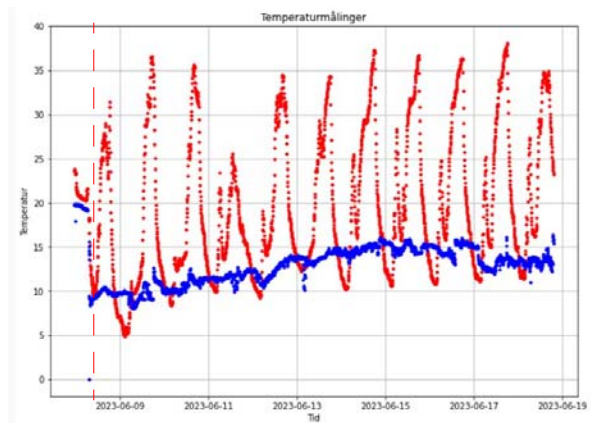


I tillegg til de 2732 vellykkede målingene mistet vi 65 (2,4%), sannsynligvis fordi det tok for lang tid å låse til satellittene slik at den interne “vakthunden” avsluttet programmet.





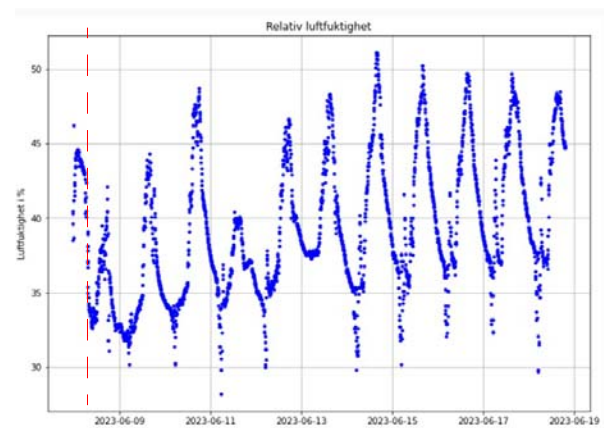
Figuren til høyre viser temperaturen. Den røde sterkt varierende kurven er lufttemperaturen inne i boksen som varmes opp til over 35°C på dagtid når sola står på. Den blå kurven viser temperaturen i vannet ca. 70 cm under overflata, og vi kan se en stigende tendens gjennom måleserien før den flater ut og faller litt av. Den stiplede vertikale streken til venstre angir tidspunktet da bøya ble satt ut i sjøen. Vi ser at vanntemperaturen stiger fra ca. 9°C til nesten 16°C på det varmeste i løpet av 6 dager noe som overrasket oss.



For å verifisere målingene våre, sammenlignet vi dem med det som var målt ved SINTEFs målebøye ved Munkholmen<sup>17</sup> i den samme perioden. Her ser vi den samme tendensen som vi målte.



Figuren til høyre viser relativ fuktighet inne i den vannrette boksen. Siden hensikten bl.a. var å sjekke om vi hadde vanninntrengning, så er fuktighetsmålinger av interesse. Vi legger merke til at fuktigheten øker når temperaturen øker. Normalt skulle relativ fuktighet synke med økende temperatur dersom vanninnholdet i lufta er konstant, siden varm luft kan ta opp

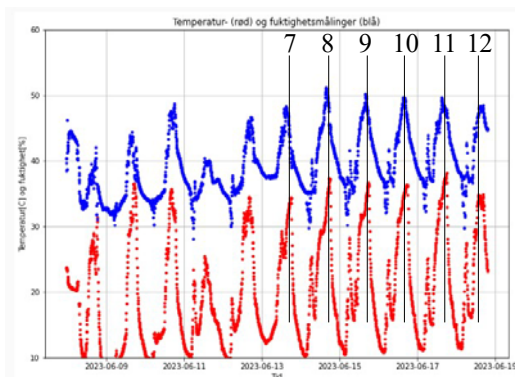


17.[https://oceanlab.azure.sintef.no/d/FWE\\_ctR4k/water-temperature](https://oceanlab.azure.sintef.no/d/FWE_ctR4k/water-temperature)

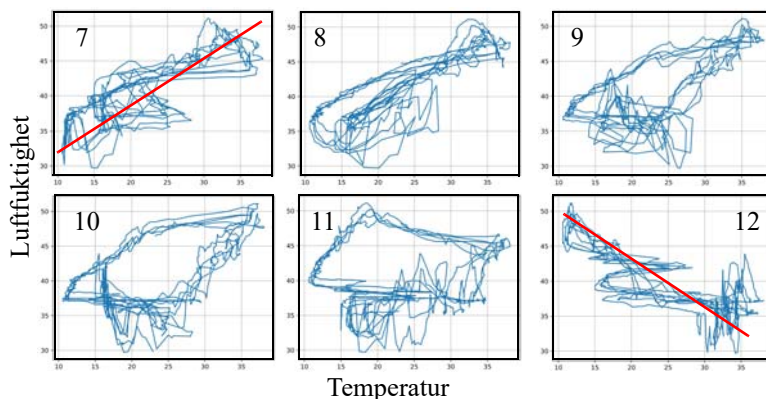


mer vanddamp. Siden det i dette tilfellet ikke er slik, må det bety at det er fuktighet inne i boksen som fordamper og dermed gjør at den relative fuktigheten øker med økende temperatur. Dette er jo egentlig ikke noe godt tegn.

I figuren under har vi plottet temperatur og luftfuktighet i samme diagrammet, for tydeligere å se sammenfallet av maksimal temperatur og fuktighet. Vi har i tillegg markert temperaturtoppene hos de seks siste døgnene, for ytterligere å tydeliggjøre forskjeller.



Studerer man de seks siste døgnene, kan man se en gradvis forskyvning, som kommer tydeligere fram dersom vi tegner opp fuktighet som funksjon av temperaturen (korrelerer)<sup>18</sup>.



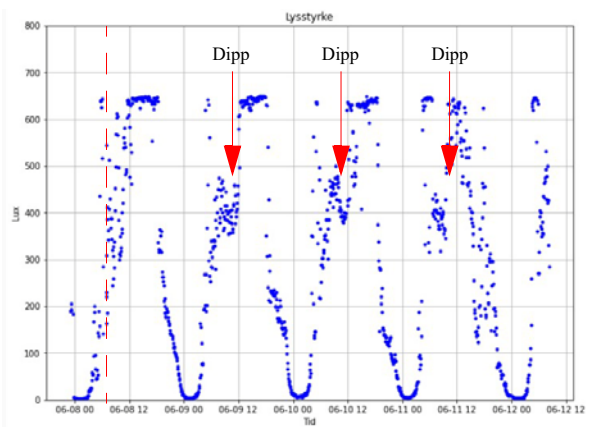
Her ser vi hvordan korrelasjonen dreier fra positive verdier til negativ verdier over de seks døgnene. En hypotese er at MDF-platene inne i boksen tar opp og avgir fuktighet gjennom døgnet, en prosess som tar tid og som er med å påvirke korrelasjonen mellom luftfuktighet og temperatur.

---

18. Beregningene og illustrasjonene er utført av førsteamanuensis Tore Haug-Warberg ved Institutt for kjemisk prosess teknologi, NTNU



Figuren til høyre viser lysinnstråling i boksen. Lysmåleren er noe uheldig plassert, da den vender ned mot bunnen av boksen, men som figuren viser så gir målingene en antydning om variasjonen over døgnet. Vi legger merke til at det er en dipp i lysstyrken på formiddagen. Dette kan skyldes at bøya kommer i skyggen av strukturer på brygga på dette tidspunktet. Figuren viser kun målinger fra de første dagene slik at formen på den daglige kurven skal komme tydeligere fram.



Forøvrig var lysmåling noe vi fikk med på kjøpet og som hadde lite for seg i denne sammenhengen.

I tillegg gjorde vi GPS-målinger, hovedsakelig for å få erfaringer med bruk av GPS og låsetider, og for å få referansetidspunkter (epoketid fra satellittene) som vi brukte i presentasjonene som vist over.

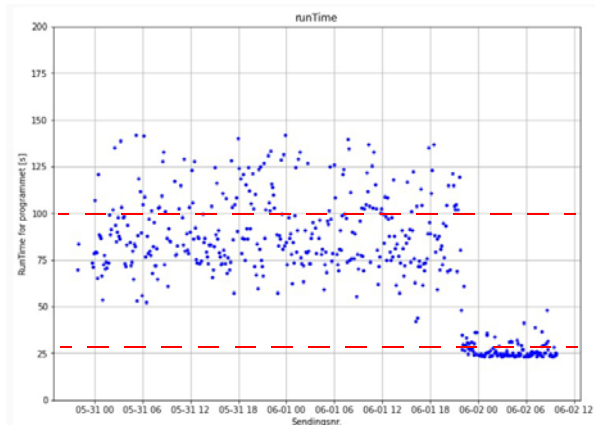




Det ble foretatt målinger ca. hvert 5–6 minutt (i gjennomsnitt 5,7 min.). I starten stusset vi over at det var så stor spredning i på-tiden. Ved å logge tidsmålinger internt i programmet, fant vi at årsaken hovedsakelig skyldtes lang låsetid for GPS'en. Dette undret vi oss over siden vi brukte GPS med backup batteri og hadde forventet raskere låsetid. Dette løste seg imidlertid da vi sjekket tilstanden på batteriene som viste seg å være utladet etter drøyt ett års bruk.

Her ser vi hvordan på-tiden falt fra et gjennomsnitt på typisk 100 sekunder og ned til 26 sek. da vi skiftet batteri. Låsetiden til satellittene utgjør ca. halvparten av på-tiden.

Det vi foreløpig ikke vet er hvor mye batteriet betyr for låsetiden dersom intervallene mellom målingene blir på timer.



### Observasjoner ved innhenting

Det vi først og fremst la merke til var at utstyret denne gangen var langt mer begrodd enn da vi hentet inn bøya etter 14 dager i sjøen i november 2022. Dette er ikke overraskende på bakgrunn av hvordan livet i havet har en oppblomstring på våren.



Inspeksjon av instrumentboksen med elektronikken viste imidlertid ingen tegn på inntrengning av fuktighet, slik vi kanskje kunne frykte. Det gjorde derimot sensorboksen montert på siden av avløpsrøret. Den hadde vært under vann hele tiden mens bøya var utplassert. Ved åpning av boksen viste det seg at den var ca. 1/5 full med vann. Det var ikke særlig vanskelig å skru opp





kabelgjennomføringene, men desto vanskeligere å få tatt av endestykket. Alle gjennomføringer og pakninger ble smurt med litiumfett før vi satte ut bøya. En hypotese er at lekkasjen har skjedd via kabelgjennomføringene for “sensor” og ledning



Konklusjon, det må gjøres en bedre jobb med hensyn til tetning.

### 1.3.3 Test 3 - Landtest

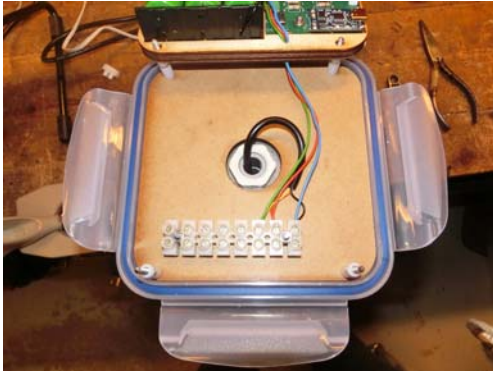
Test 3 gjøres på land med instrumentet montert på en brakett på mønet av et hus. Hensikten med denne testen er å se hvordan et lite solcellepanel påvirker batterikapasiteten på sikt. Dessuten vil det bli en test på hvor godt “matboksen” fungerer når den blir montert på hodet med fire skruer ført gjennom bunnen og en kabelgjennomføring på siden for antennekabelen.



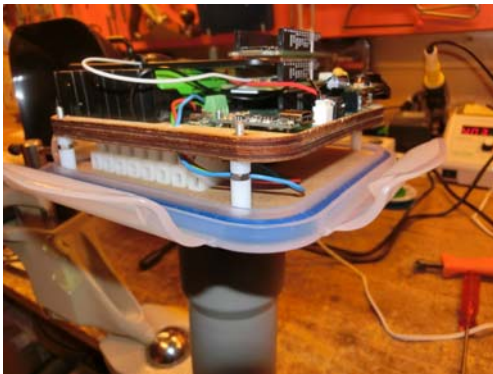
Solcellepanelet gir typisk 6V og er skråstilt og vendt mot syd. Panelet kan vippes opp og ned dersom støtta bak løsnes. Kabelen fra panelet går inn gjennom en kabelgjennomføring på forsiden. Elektronikken er montert på to plattformer inne i boksen. En nedre der ledningene fra temperatursensoren (inne i avløpsrøret) kommer opp og monteres til en rekkeklemme som igjen er ført opp til øverste plattform der batteriene og resten av elektronikken er plassert.



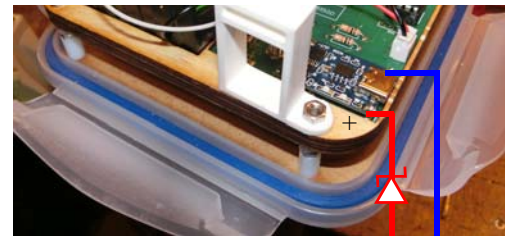
Det er tilstrekkelig plass på nedre plattform til å legge forsterkerelektronikken til pH-, salinitets- og sensor for måling av oksygeninnhold, kan plasseres. Bildet under til venstre viser nederste plattform med rekkeklemmen.



Bildet over til høyre viser øvre plattform som inneholder batteripakken og elektronikken med mikrocontrollerkortet. Vi ser hvordan NB IoT-antenna er festet i to 3D-printede braketter. Bildet under til venstre viser de to plattformene sett fra siden, og bildet under til høyre viser et nærbilde av den ene braketten som holder antenna.



Solcellepanelet er montert parallelt med USB-C inngangen til ladekretsen. Dette er ikke en ideell montering, og vil sannsynligvis ikke gi en optimal utnyttelse av solcellepanelet. En må også koble fra solcellepanelet ved lading via USB-C-pluggen. Skal man bruke begge bør det monteres en Schottky-diode mellom solcellepanelet og ladekretsen, som vil hindre at det går strøm fra USB-C inngangen gjennom solcellepanelet.



Solcellepanelet kobles til + og – terminalen av ladekretsen via en Schottky-diode.



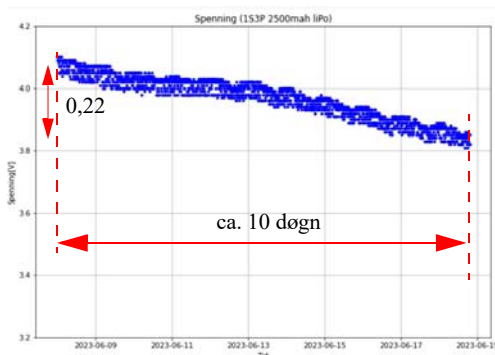


Det hele er plassert på toppen av et avløpsrør og montert i en brakket på mønet av huset som vist på bildet til høyre. Kretsen er juttstyrt med to temperatursensorer, en i selve boksen og en inne i avløpsrøret som er montert i braketten.

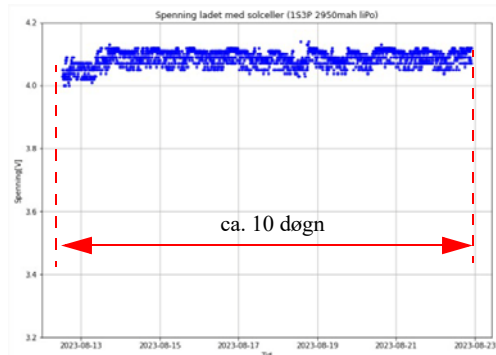


### Måling av batterikapasitet

Et viktig resultat fra denne testen er å undersøke hvordan batterispenningen endrer seg med tiden når solcellen er tilkoblet ladekretsen, Figuren under sammenligner spenningen med og uten solceller:



Uten solcellelading

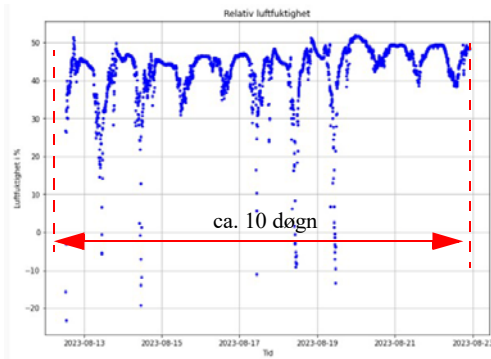


Med solcellelading

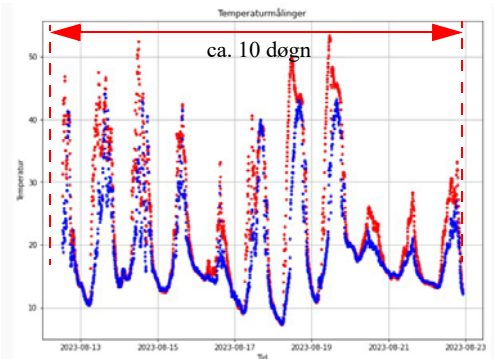
Som vi ser holder batterispenningen seg omtrent konstant med solcellelading. Vi registrerer også at batteriene får et løft etter første dag med sol, slik at batterispenningen kommer opp på topp.



Vi oppdaget også at fuktighetssensoren fikk problemer når temperaturen ble for høy. Når temperaturen inne i boksen overskred 50°C ga fuktighetssensoren negative verdier for relativ fuktighet. Dette er litt overraskende siden spesifikasjonen sier at den skal tåle et temperaturområde på -40 til +120°C.



Relativ fuktighet



Temperatur i boksen (rød) i røret (blå)





## 2 Prosjektet – Oppgavesamlingen

I denne delen skal vi ta for oss de enkelte delprosjektene som vi skal gå gjennom under kurset og bygge opp programmet bit for bit.

Som nevnt tar vi sikte på å bygge opp kompetansen gradvis gjennom å bringe inn stadig nye elementer i eksempelprogrammet vårt. Det endelige produktet (oppdraget) kan beskrives slik:

**Oppdraget:** *Det skal lages et instrument for måling av miljødata, inntil videre bare vann-temperatur. Instrumentet skal utstyres med en GPS-mottaker slik at målingene kan tid- og stedfestes. Dataene skal legges på et minnekort (SD-kort) i tillegg til at de skal overføres til en server hvor måleserien skal legges i en CSV-fil. De lagrede dataene skal presenteres som tabeller og grafer, gjerne med bruk av Python-skript, Excel, Google Earth, eller annen passende programvare. Om ønskelig kan man også bygge opp et styringskort for styring av måletidspunkt og som slår av strømmen mellom målingene for å spare energi.*

### Oversikt over deloppdrag

I løpet av 10 deloppdrag skal vi bygge opp og overføre data fra sensornoden vår til serveren og presentere dataene. Denne gangen skal vi konsentrere oss om måling av vanntemperatur og posisjon.

Vi har delt opp prosjektet i følgende 10 deloppdrag:

**Deloppdrag 1:** *Installer programeditor for Arduino og tilleggsmoduler for programmering av MKR NB 1500.*

**Deloppdrag 2:** *Koble opp hårdvaren, monter antennen og installer SIM-kortet.*

**Deloppdrag 3:** *Installer nødvendige biblioteker og last opp og kompiler eksempelprogrammet. Gå gjennom koden og få oversikt over programmet.*

**Deloppdrag 4:** *Les av sensoren for måling av vanntemperatur og skriv resultatet ut til monitoren. Stikk sensoren ned i et glass med vann og undersøk om temperaturen er som forventet. Avlesningen av temperaturen legges inn i datastrengen i hovedprogrammet.*

**Deloppdrag 5:** *Monter MKR GPS-kortet og installer det tilhørende biblioteket. Bruk eksempelprogrammet og inkluder funksjonen for lesing av GPS-mottakeren. Kontroller at avleste data er som forventet.*

**Deloppdrag 6:** *Koble opp sensornoden til NB IoT-nettverket og overfør data til serveren. Gå inn på serveren og kontroller at lagrede data er som forventet.*

**Deloppdrag 7:** *Inkluder en “vakhund” som resetter programmet dersom det skulle stoppe opp.*

**Deloppdrag 8:** *Monter en SD-kort terminal, installer nødvendig bibliotek, opprett en fil og skriv måldataene til filen.*

**Deloppdrag 9:** *Presenter måleresultatene med passende programvare, f.eks. Excel eller et Python-script evt. Google Earth. Kommenter resultatene.*

**Deloppdrag 10:** *Lodd opp PLSC MKR-kortet og sjekk at det fungerer som tiltenkt.*



## 2.1 Deloppdrag 1: Installasjon av programvare

I dette deloppdraget skal vi installere programvaren som vi trenger for å bygge opp sensornoden vår.

**Deloppdrag 1:** *Installer programeditor for Arduino og tilleggsmoduler for programmering av mikrokontrollerkortet MKR NB 1500.*

1. *Installasjon av Arduino IDE*  
Beskrivelsen av installasjonen finner du i avsnitt 3.1 på side 49
2. *Installasjon av tilleggsmodul for håndtering av MKR NB 1500*  
Du skal nå installere programvare for å kunne bruke kortet MKR NB 1500. Beskrivelse av installasjonen og valg av mikrokontrollerkort finner du i delkapittel avsnitt 3.2 på side 51.
3. *Installasjon av bibliotek for å kunne programmere MKR NB 1500*  
Her skal du installere biblioteket som gjør det enkelt å programmere MKR NB 1500. Beskrivelse av installasjonen av biblioteket finner du i delkapittel avsnitt 3.3 på side 53.

Du er nå klar til å bygge opp og programmere MKR NB 1500.

## 2.2 Deloppdrag 2: Oppkobling av hårdvare og montering av SIM-kort

I dette deloppdraget skal vi montere hårdvaren og SIM-kort.

**Deloppdrag 2:** *Koble opp hårdvaren, monter antennen og installer SIM-kortet*

1. *Anskaffelse av hårdvare*  
I denne sammenhengen har vi valgt å bruke Arduino MKR NB 1500 som krever en antenne som monteres til en plugg på mikrokontrollerkortet. I tillegg kan følgende kort være greie å ha: MKR ENV (Miljøkort med SD-kortterminal) og MKR GPS (GPS-kort) med batteri (CR1216). Et prototypkort kan også være godt å ha dersom man ønsker å eksperimentere med egne sensorer. I denne sammenhengen har vi valgt et prototypkort med SD-kort terminal for lagring av data (MKR SD Proto Shield). DS18B20 er en sensor for måling av vanntemperatur, se avsnitt 6.3.3 på side 167. Denne har vi valgt for enkelhetsskyld å montere til stifter slik at den lett kan plugges ned i mikrokontrollerkortet eller et shield-kort. For enkel montering av DS18B20 se avsnitt 6.3.3 på side 167. Se forøvrig også bestillingsliste i vedlegg A side 222. Mikrokontrollerkortet og shield-kortene er beskrevet i avsnittene 4.4.1 side 92 og 4.5 side 95.
2. *Anskaffelse av SIM-kort*  
Bestilling av SIM-kort er relativt enkelt, men man må regne med at det tar inntil 14 dager å få dem i posten. Du må også bestemme deg for hvilken IoT-teknologi du trenger, LTE-M eller NB-IoT. Se avsnitt 3.4 side 53. Her er det flere muligheter avhengig av om man ønsker innkjøp av inntil 9 SIM-kort eller opprettelse av bedriftsavtale for mer enn 9 SIM-kort. Pass på å merke av i bestillingen at kortene skal være uten krav om inntasting av PIN-kode.  
I forbindelse med kurset skal kortene være innkjøpt og tilgjengelige for kursdeltakerne.



3. *Monter antennen og SIM-kort*  
Antennen og SIM-kortet monteres på mikrokontrollerkortet MKR NB 1500 som omtalt i avsnitt 3.5 side 56. Vær forsiktig når du monterer antennen da antennekontakten er liten og skjør.

## 2.3 Deloppdrag 3: Installer biblioteker og kjør eksempelprogrammet

Her skal vi forberede for overføring av dummy-data, det vil si data som genereres av programvaren og ikke av sensorer. Før vi kobler til sensorene er disse verdiene 0.

**Deloppdrag 3:** *Installer nødvendige biblioteker og last opp og kompiler eksempelprogrammet. Gå gjennom koden og få oversikt over programmet.*

1. *Hent og installer biblioteker og eksempelprogrammet:* EVU-kurs23-Eksempelprogram.ino. Dette er beskrevet i avsnitt 3.6 side 56.
2. *Last opp og studer koden til eksempelprogrammet*  
Last opp eksempelprogrammet: EVU-kurs23-Eksempelprogram.ino som er vedlagt i vedlegg J side 303. Programmet kan også lastes ned fra:

[www.ntnu.no/skolelab/bla-hefteserie](http://www.ntnu.no/skolelab/bla-hefteserie)

Gå ned til overskriften: *Programmering av Tingenes internett (IoT)*

Åpne fanen: *EVU-kurs: Miljøovervåking med havbøye*

Last ned og installer programmet: *EVU-kurs23-Eksempelprogram.ino*

Gå gjennom programmet og forsøk å få oversikt over de ulike delene. Bruk gjennomgangen i avsnitt 3.7 side 59.

3. *Sjekk Debug modus*  
Langt oppe i programmet finnes følgende programlinjer (:

```
// Deklarasjon av konstanter:  
#define DEBUG // Kommenter bort denne dersom Serial.print ikke skal inkluderes  
//#define LED // Kommenter bort denne dersom LED indikasjon ikke ønskes
```

Sjekk at #define DEBUG ikke er kommentert bort, mens //#define LED er kommentert bort. Dersom konstanten DEBUG er deklartert så vil vi kunne skrive ut kommentarer i monitoren. Det er bare aktuelt når kortet er koblet til PC'en med en kabel. Dersom dette ikke er tilfelle må vi kommentere bort denne linjen.

4. *Velg et unikt filnavn*  
På linje ca. 43 finner man følgende kode:

```
char filename[]="<filnavn>.txt"; // Velg navn på fil for lagring av data på serveren
```

Erstatt <filnavn> med et unikt filnavn, der du kan finne igjen dataene som lastes ned til serveren. Filen blir liggende sammen med andres filer. Husk at filnavnet skal ende på .txt, da er det lettere å åpne med en tekst-editor.

5. *Sett opp loop()-funksjonen*  
Gå ned til loopen og velg inn funksjoner som er merket med rødt, resten kan foreløpig være kommentert bort:



```
void loop() {  
  
    //readGPSdata();    // Inkluder GPS  
    //readWaterTemp(); // Inkluder måling av temp. i vann  
    //readBatVolt();   // Inkluder måling av baterispennning om den er implementert  
  
    makeString();     // Bygge opp bufferet for overføring av data  
    //sdPrint();      // Skriv data til SD-terminal  
    //connectToGPRS(); // Koble til GPRS-nettverket  
    //connectToServer(); // Koble opp mot server og overfør data  
  
    //printData();    // Skriv data til monitoren  
    printDataString(); // Skriv ut bufferet til monitoren  
    //MyWatchDoggy.clear(); // Resetter vakthunden  
  
    num++;           // Målingens nummer fra start  
    //GoToSleep();   // Legg mikrokontrolleren og GPS'en i dvale  
    delay(Pause);    // Ev. legg inn en pause i loopen  
}
```

6. *Sjekk at linjene med data skrives ut i monitoren*

Åpne monitoren og se at programmet skriver ut dummy-dataene som forventet. Det er den samme datastrengen som legges i fila på serveren og på SD-kortet. Det er funksjonen `printDataString()`; som skriver ut datastrengen.

## 2.4 Deloppgdrag 4: Monter sensoren for måling av vanntemperaturen

Vi skal bruke den vanntette temperatursensoren DS18B20 for måling av temperatur i vann. Det er ingen ting i veien for at vi også kan bruke denne for måling av temperatur i luft, men responsen på temperaturodringer vil være langsommere enn for vann.

**Deloppgdrag 4:** *Les av sensoren for måling av vanntemperaturmåleren og skriv resultatet ut til monitoren. Stikk sensoren ned i et glass med vann og undersøk om temperaturen er som forventet. Avlesningen av temperaturen legges inn i datastrengen i hovedprogrammet og sendes over til serveren.*

1. *Monter sensoren*

Plugg temperatur-sensoren DS18B20 (se bilde til høre) inn i MKR NB 1500 som omtalt i avsnitt 6.3.3 side 167 - *Enkel oppkobling* (side 168).

2. *Installer biblioteket*

Last ned og installer biblioteket for sensoren DS18B20 og biblioteket for bruk av entrådsbussen. Dette skal allerede være gjort under Deloppgdrag 3.

3. *Bruk eksempelprogrammet* (EVU-kurs23-Eksempelprogram.ino)

Bruk eksempelprogrammet for avlesning av sensoren DS18B20 og inkluder funksjonen





`(readWaterTemp())` i `loop()`-funksjonen og hent inn temperaturen fra sensoren, som vist på programutsnittet under. Kompiler og kjør programmet. Åpne monitoren og se at du får rimelige temperaturverdier.

```
void loop() {  
  
    //readGPSdata();    // Inkluder GPS  
    readWaterTemp();    // Inkluder måling av temp. i vann  
    //readBatVolt();    // Inkluder måling av baterispennning om den er implementert  
  
    makeString();    // Bygge opp bufferet for overføring av data  
    //sdPrint();    // Skriv data til SD-terminal  
    //connectToGPRS();    // Koble til GPRS-nettverket  
    //connectToServer();    // Koble opp mot server og overfør data  
  
    printData();    // Skriv data til monitoren  
    printDataString();    // Skriv ut bufferet til monitoren  
    //MyWatchDoggy.clear();    // Resetter vakthunden  
  
    num++;    // Målingens nummer fra start  
    //GoToSleep();    // Legg mikrokontrolleren og GPS'en i dvale  
    delay(Pause);    // Ev. legg inn en pause i loopen  
}
```

For nærmere beskrivelse av sensoren DS18B20 se avsnitt 6.3.3 side 167.

#### 4. Mål en kjent temperatur i vann

Stikk sensoren i vann med ulike temperaturer og undersøk at den gir rimelige resultater. Kontroller målingene med et kalibrert termometer. Om nødvendig kalibrer sensoren.

## 2.5 Deloppgave 5: Monter GPS-kortet og inkluder GPS data

Vi skal nå montere og hente inn data fra MKR GPS-kortet.

**Deloppgave 5:** Monter MKR GPS-kortet og installer det tilhørende biblioteket. Bruk eksempelprogrammet og inkluder funksjonen for lesing av GPS-mottakeren. Kontroller at avleste data er som forventet

#### 1. Hent og installer MKR GPS-biblioteket

Hent ned og installer MKR GPS-biblioteket i Arduino editoren. Dette skal være gjort i Deloppgave 3, se evt. avsnitt 3.6.1 side 57.

#### 2. Monter GPS-kortet

Monter GPS-kortet ved hjelp av kabelen som kommuniserer med mikrokontrolleren via I<sup>2</sup>C-bussen. Dersom GPS-kortet plugges ned som et shield-kort, så overfører GPS-kortet data via Rx/Tx (UART), i så fall må vi bruke et annet bibliotek siden det vi bruker er beregnet til å hente inn data via I<sup>2</sup>C-bussen (kabelen). Vi anbefaler derfor å koble opp kortet med kabelen som programmet er skrevet for, se side 105.



**NB!** Det er viktig at GPS-mottakeren legges et sted der mottakeren har tilgang til et sett med GPS-satellitter, f.eks. utendørs eller nær et vindu med fri sikt til en større del av himmelen. Første gang den slås på kan det ta flere minutter før den låser til et akseptabelt antall satellitter. Ved bruk av batteri på GPS-kortet vil informasjonen lagres slik at det senere går fortere å låse til satellittene.

3. Inkluder funksjonen for avlesning av GPS (EVU-kurs23-Eksempelprogram.ino)  
Inkluder funksjonen (`readGPSdata1()`;) som leser av GPS-mottakeren.

```
void loop() {  
  
    readGPSdata1();      // Inkluder GPS  
    readWaterTemp();    // Inkluder måling av temp. i vann  
    //readBatVolt();     // Inkluder måling av baterispennning om den er implementert  
  
    makeString();       // Bygge opp bufferet for overføring av data  
    //sdPrint();        // Skriv data til SD-terminal  
    //connectToGPRS();  // Koble til GPRS-nettverket  
    //connectToServer(); // Koble opp mot server og overfør data  
  
    printData();        // Skriv data til monitoren  
    printDataString();  // Skriv ut bufferet til monitoren  
    //MyWatchDoggy.clear(); // Resetter vakthunden  
  
    num++;              // Målingens nummer fra start  
    //GoToSleep();     // Legg mikrokontrolleren og GPS'en i dvale  
    delay(Pause);      // Ev. legg inn en pause i loopen  
}
```

Med biblioteket følger det også med eksempelkode som bl.a. inneholder et testprogram for lesing og utskrift av GPS-data til monitoren.

4. *Kontroller resultatet*  
Åpne monitoren og studer om koordinatene som skrives ut og som legges i datastrengen ser rimelige ut.
5. *Kontroller målt posisjon med Google maps*  
Klipp ut lengde- og breddegrad og lim det inn i Google maps og kontroller om den målte posisjonen stemmer. Se avsnitt 5.5 side 150 for bruke Google maps.

## 2.6 Deloppdrag 6: Oppkobling til GPRS-nettet og nedlasting til serveren

I dette deloppdraget skal vi koble oss til NB IoT-nettverket (4G) og sende databufferet til serveren. Deretter skal vi åpne fila på serveren og se at dataene er kommet fram.

**Deloppdrag 6:** Koble opp sensornoden til NB IoT-nettverket og overfør data til serveren. Gå inn på serveren og kontroller at lagrede data er som forventet.

Vi skal nå gjøre forberedelser for overføring av data til serveren.



1. *Hent og installer bibliotek og eksempelprogrammet* (EVU-kurs23-Eksempelprogram.ino)  
Bibliotekene for å bruke NB IoT teknologien skal være installert og beskrevet i avsnitt 3.3 side 53.
2. *Inkluder kode for oppkobling og nedlasting*  
Koden for oppkobling til GPRS-nettet finnes i funksjonen `connectToGPRS()`; og nedlasting til serveren i funksjonen `connectToServer()`; . Som det framgår så er mesteparten av funksjonene varsler om at alt går greit eller om at det oppstår feil.  
  
Inkluder funksjonene `connectToGPRS()`; og `connectToServer()`; i `loop()`-funksjonen. Det kan også være lurt å **droppe avlesningen av GPS-mottakeren** under denne uttestingen for å unngå å ha for mange usikre faktorer i spill samtidig.

```
void loop() {  
  
    readGPSdata();           // Inkluder GPS  
    readWaterTemp();        // Inkluder måling av temp. i vann  
    //readBatVolt();        // Inkluder måling av baterispennning om den er implementert  
  
    makeString();           // Bygge opp bufferet for overføring av data  
    //sdPrint();            // Skriv data til SD-terminal  
    connectToGPRS();        // Koble til GPRS-nettverket, om det ikke alt er gjort i setup  
    connectToServer();      // Koble opp mot server og overfør data  
  
    printData();            // Skriv data til monitoren  
    printDataString();     // Skriv ut bufferet til monitoren  
    //MyWatchDoggy.clear(); // Resetter vakthunden  
  
    num++;                  // Målingens nummer fra start  
    //GoToSleep();         // Legg mikrokontrolleren og GPS'en i dvale  
    delay(Pause);          // Ev. legg inn en pause i loopen  
}
```

3. *Kompiler og kjør eksempelprogrammet*  
Når programmet kjører vil det sende over de tallverdiene som ligger i datastrengen (buffer).
4. *Sjekk data på serveren*  
Gå inn på følgende nettadresse: <http://sensor.marin.ntnu.no/logs/> og finn filen med navnet dere valgte, og sjekk at de forventede dataene ligger der. Bruk oppfriskningsverktøyet i fila for å laste inn nye innkomne data.
5. *Evt. inkluder GPS-data*  
Dersom nedlastingen til serveren gikk greit, kan man inkludere avlesning av GPS-mottakeren.





## 2.7 Deloppdrag 7: Inkluder “vakhund”

I dette deloppdraget skal vi inkludere en “vakhunden” som sørger for å resette programmet dersom det stopper opp. Dette er den interne vakhunden som er realisert i programmet.

**Deloppdrag 7:** *Inkluder en “vakhund” som resetter programmet dersom det skulle stoppe opp.*

### 1. Inkluder en programvarebasert “vakhund”

Inkluder funksjoner i programmet som styrer oppsett av “vakhunden”. Oppsettet av “vakhunden” gjøres i `setup()`-funksjonen. Under er det viset et utsnitt av `setup()`-funksjonen. Her fjerner vi kommentartegnene foran de to “vakhund” funksjonene (`MyWatchDoggy . . .`) som vist under:

```
// Initialisering av "vakhund"
MyWatchDoggy.attachShutdown(myshutdown); // Starter funksjonen myshutdown() ved reset
MyWatchDoggy.setup(WDT_SOFTCYCLE1M);    // Setter intervalltiden for "vakhunden"
```

Her har vi valgt et “vakhund”-intervall på ca. 1 min. (64 sek). Dvs. at “vakhunden” resetter programmet dersom en syklus i programmet tar lengre tid enn 64 sek. Intervallet settes opp med konstanten `WDT_SOFTCYCLE1M`. Her kan man velge intervaller på inntil drøyt 16 min. Se avsnitt 4.10.1 side 125 for mer informasjon.

I tillegg må vi inkludere klarering av “vakhunden” i `loop()`-funksjonen (`MyWatchDoggy.clear()`). Dette gjør vi for å hindre at “vakhunden” resetter programmet så lenge programmet går som normalt og ikke stopper opp.

```
void loop() {

    readGPSdata1();      // Inkluder GPS
    readWaterTemp();    // Inkluder måling av temp. i vann
    //readBatVolt();    // Inkluder måling av baterispennning om den er implementert

    makeString();       // Bygge opp bufferet for overføring av data
    //sdPrint();        // Skriv data til SD-terminal
    connectToGPRS();    // Koble til GPRS-nettverket om det ikke alt er gjort i setup
    connectToServer();  // Koble opp mot server og overfør data

    printData();        // Skriv data til monitoren
    printDataString();  // Skriv ut bufferet til monitoren
    MyWatchDoggy.clear(); // Resetter vakhunden

    num++;              // Målingens nummer fra start
    //GoToSleep();     // Legg mikrokontrolleren og GPS'en i dvale
    delay(Pause);      // Ev. legg inn en pause i loopen
}
```

### 2. Sjekk at “vakhunden” fungerer

Kjør programmet og legg inn en pause som er lenger enn intervallet “vakhunden” kan tillate og se at den resetter og starter opp programmet på nytt. Dvs. at verdien på pausen må settes til mer enn 64000 ms. Dette gjør vi i starten av programmet der vi deklarerer våre variabler.



```
long Pause = 64000; // En evt. pause i programloopen
```

Mot slutten av pausen vil man høre at mikrokontrolleren kobler seg fra USB-porten og resetter programmet. Ved en slik resetting vil forbindelsen til monitoren brytes og ikke kobles opp igjen før vi lukker og starter den på nytt dersom vi bruker IDE 1x, slik er det nødvendigvis ikke for IDE 2x.

Vi har også erfart at mikrokontrolleren mister kontakten med GPRS-nettverket og forbindelsen til serveren under lengre pauser.

## 2.8 Deloppgdrag 8: Skriv dataene til SD-kort

I dette deloppgdraget skal vi skrive datastrengen til et SD-kort

**Deloppgdrag 8:** *Monter en SD-kort terminal, installer nødvendig bibliotek, opprett en fil og skriv måledataene til filen.*

### 1. Monter SD-kort terminalen

Den enkleste måten å inkludere en SD-kort terminal er å bruke et av shield-kortene som tilhører MKR-serien. En slik terminal finnes på følgende kort: MKR ENV (miljø-kortet), se avsnitt 4.5.1 side 96, et MKR MEM kort som både inneholder mer lagerplass, en SD-kort terminal og plass for prototyping, se avsnitt 4.5.3 side 111. Så finnes det et shield-kort kun med en SD-kortterminal og plass til prototyping (MKR SD Proto Shield), vi har valgt å bruke dette kortet her, se avsnitt 4.5.5 side 112.

Alle disse kortene kan plasseres på toppen av mikrokontrollerkortet, som vist i avsnitt 4.5.1 side 96, der miljøkortet er plassert på toppen av mikrokontrollerkortet. Ta det kortet som du har tilgjengelig og plasser det på toppen av mikrokontrollerkortet. Hylsekontaktene er merket med port-nummer. **Pass på at riktig bein kommer i riktig hylsekontakt.**

### 2. Sjekk filnavnet

I setup()-funksjonen opprettes fila. Her kan man sette opp filnavnet som skal brukes ved skriving til SD-kortet. En må gjerne gjenbruke filnavnet som ligger der (DataFile.txt) siden denne filen kun blir liggende på SD-kortet. Dersom man velger et eget navn, *vær klar over at antall tegne i selve navnet ikke overskrider 8*. Blir det flere opprettes ikke fila.

```
// Lag en file med navnet DataFile.txt
myFile = SD.open("DataFile.txt", FILE_WRITE);
delay(1000);
myFile.close();
delay(100);
```

### 3. Inkluder skriving til SD-kort

For å få skrevet dataene til SD-kortet kaller vi opp funksjonen `sdPrint()` i `loop()`-funksjonen som vist under:

```
void loop() {

    readGPSdata1(); // Inkluder GPS
    readWaterTemp(); // Inkluder måling av temp. i vann
    //readBatVolt(); // Inkluder måling av baterispennning om den er implementert
```



```
makeString();           // Bygge opp bufferet for overføring av data
sdPrint();              // Skriv data til SD-terminal
connectToGPRS();       // Koble til GPRS-nettverket
connectToServer();     // Koble opp mot server og overfør data

printData();           // Skriv data til monitoren
printDataString();     // Skriv ut bufferet til monitoren
MyWatchDoggy.clear(); // Resetter vakthunden

num++;                 // Målingens nummer fra start
//GoToSleep();        // Legg mikrokontrolleren og GPS'en i dvale
delay(Pause);         // Ev. legg inn en pause i loopen
}
```

#### 4. Lesing av fil på SD-kort

For å sjekke at dataene er kommet inn i fila på SD-kortet, tar man ut SD-kortet og setter det inn i en tilsvarende leser i PC'en. Fila åpnes med en tekst-editor, som f.eks. Notepad++ eller lignende.

#### 5. Inkluder GPS-data

Dersom GPS-data fortsatt er kommentert bort, så kan du jo prøve å inkludere denne igjen. Sørg imidlertid for at mottakeren ligger nær vindu.

## 2.9 Deloppdrag 9: Presentasjon av måleresultatene

I dette deloppdraget skal vi presentere måleresultatene.

**Deloppdrag 9:** *Presenter måleresultatene med passende programvare, f.eks. Excel eller et Python-script evt. Google Earth. Kommenter resultatene.*

#### 1. Lag grafer av målinger

Presenter måledata som funksjon av tiden. Vis gjerne vanntemperatur som funksjon av tiden ved hjelp av Excel (avsnitt 5.3 side 137) eller ved Python skript (avsnitt 6 side 155).

#### 2. Plott posisjonen i Google Earth

Hent fram kolonnene med lengde- og breddegrad og legg verdiene inn i Google Earth og vis på kartet hvor målingene er gjort. Se avsnitt 5.5 side 150.

#### 3. Sjekk måledataene

Gå gjennom måledataene og sjekk at de er rimelig. Evt. beskriv avvik.

#### 4. Plott eksempel filen med Python eksempelprogram

På serveren ligger målefila SeptKursTest\_5.txt ([http://sensor.marin.ntnu.no/logs/SeptKursTest\\_5.txt](http://sensor.marin.ntnu.no/logs/SeptKursTest_5.txt)). Denne fila kan leses og dataene kan plottes og dels analyseres med Python-scriptet i vedlegg F.1 side 267. For å kunne kjøre programmet må man installere programmet Spyder. Hvordan dette installeres og litt om bruken er vist i avsnitt 5.4.1 side 141.

Fila er skrevet spesielt for å lese den nevnte fila. For å lese andre filer med samme format og fra samme nettsadresse må filnavnet endres.



## 2.10 Deloppgdrag 10: Oppkobling av PLSC MKR-kretskort

I dette deloppgdraget skal vi montere og lodde opp PLSC MKR-kortet (PLSC – Power Loop Sleep Control MKR). Dette er et kort som sørger for lading av batteriene, opprettholde den ekstern “vakthund”-funksjonen, dvale-funksjonen og oppkonvertering av batterispenningen med mer.

**Deloppgdrag 10:** *Lodd opp PLSC MKR-kortet og sjekk at det fungerer som tiltenkt.*

1. *Monter og lodd opp kortet*

Til dette trengs en normalt god loddebolt (30W) med en passende spiss. Spissen må ikke være for bred, men egne seg til denne typen arbeid. Se kapittel 3.8 side 71 for detaljert beskrivelse av oppkoblingen.

2. *Installer operativt program*

Vi har kalt programmet *operativt* da det er et program vi har brukt for å samle inn data i felten. Programmet finnes i vedlegg K side 312, men hentes helst fra:

[www.ntnu.no/skolelab/bla-hefteserie](http://www.ntnu.no/skolelab/bla-hefteserie)

Gå ned til overskriften: *Programmering av Tingenes internett (IoT)*

Åpne fanen: *EVU-kurs – Miljøovervåking med havbøye*

Last ned og installer programmet: *Ferdig\_program\_med\_PICAXE\_MILLIS\_TEST\_2.ino*

3. *Legg inn personlig filnavn:*

Vennligst legg inn et unikt filnavn for deres prosjekt. Dette gjøres ved å endre filnavnet vist på følgende programlinje:

```
char filename[] = "<unikt filnavn>.txt";
```

Bytt ut <unikt filnavn> med et unikt navn, f.eks. `VoldSkole_Erik`. Slik at det blir stående:

```
char filename[] = "VoldSkole_Erik.txt";
```

4. *Kjør programmet og sjekk data*

Kjør programmet over noe tid og sjekk at de ønskede dataene kommer inn på filen med det unike navnet.

5. *Visualisering av data*

Bruk Excel, Python eller Google Earth for å visualisere dataene.





## 3 Programmeringen

Kapittelet er en detaljert støtte for å gjennomføre de 10 deloppgavene beskrevet i oppgavesamlingen i kapittel 2 side 37.

### 3.1 Installasjon av programvare

Avsnittet beskriver hvordan vi installerer programpakker og biblioteker for å kunne operere MKR NB 1500.

*Om du har installert Arduino IDE tidligere kan du hoppe over dette avsnittet og gå direkte til avsnitt 3.2 side 51.*

La oss først se hvordan vi kan installere: Arduino programeditoren, IDE.

#### 3.1.1 Arduino programeditor, IDE

##### Nedlasting av programvare

Arduino programeditor og kompilator hentes fra:

<http://arduino.cc/hu/Main/Software>

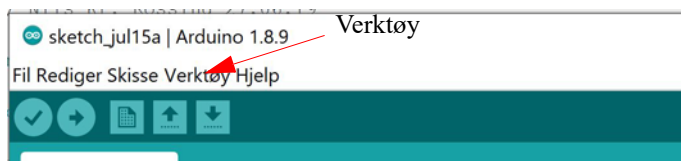
Versjonen som er brukt i denne sammenheng er 1.8.19. Filen som har navnet *arduino-1.8.19-windows*, er pakket som en zip-fil. En tilsvarende fil er tilgjengelig for Mac fra samme nettsted.

Det er også helt greit å bruke “Windows installer” versjonen som finnes på samme side. Det er imidlertid erfart noen uregelmessigheter ved bruk av Windows app-versjonen, så unngå gjerne den.

Det er også greit å installere den nye utgaven, versjon 2.1.0. Denne har en del fordeler, ved at den bl.a. gir økte muligheter til etterspore funksjonen til programmet (debugging) og feilfinning.

##### Installasjon av programvaren:

1. Klikk på *Windows installer* og følg anvisningen under installasjonen. For MAC-brukere klikk på *MAC OS X*
2. Programmet startes ved å klikke på programikonet:  .
3. Koble USB-kabelen til ønsket USB-port på PC-en.
4. Klikk på *Verktøy (Tools)* på menylinjen og velg *Kort*. Her velges hvilken variant i Arduino-familien du ønsker å jobbe med.




Etter at vi har installert tilleggsprogramvare for MKR NB 1500 (se avsnitt 3.2 på side 51), velges Arduino MKR NB 1500.











5. Klikk på *Verktøy* på menylinjen og velg *Port*. Sjekk at riktig port (Com?) er valgt. Normalt står det hvilken Arduino-variant porten er koblet til eller velg det høyeste nummeret.

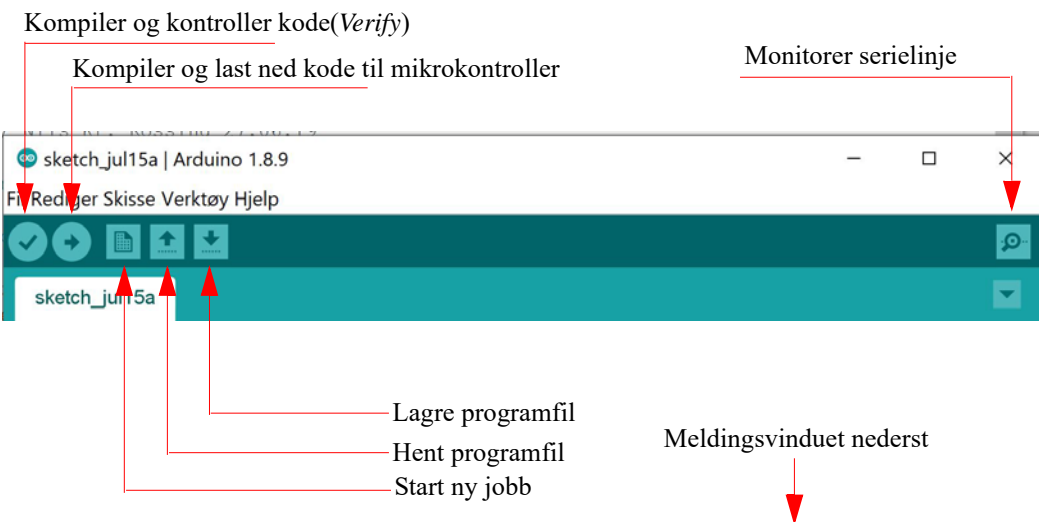
Programmet er nå klart til bruk og du kan skrive inn programlinjene. Når programmet er ferdig skrevet, skal det *kompileres*, dvs. overføres til en binærkode som mikrokontrolleren forstår. Der- som programmet inneholder ulovlige kommandoer eller skrivefeil, vil kompilatoren varsle om det og vise på hvilken linje feilen er oppdaget. Det er ikke nødvendigvis alltid der feilen befinner seg.

Derneft skal programmet lastes ned til mikrokontrollerens minne (Arduino-kortet). Dette gjøres ved å trykke på knappen .

### Kort oversikt over Arduino-editoren

Man finner følgende kommandoikoner på den grafiske menylinjen:

-  Kompiler og verifiser at koden er riktig – overfører ikke koden til mikrokontrolleren.
-  Kompiler og last ned programmet til mikrokontrolleren
-  Hent nytt “arbeidsark” også kalt skisse (“sketch”), start ny jobb
-  Hent en eksisterende programfil
-  Lagre programfilen
-  Monitorer data sendt tilbake fra mikrokontrollerkortet på serielinjen



### Manglende kontakt med kortet

Det hender at en ikke oppnår umiddelbar kontakt med Arduino-kortet når en forsøker å laste ned et program. Feilmeldingen: *avrdude: stk500\_getsync(): not in sync: resp=0x00* i meldingsvinduet betyr at det ikke oppnås kontakt med kortet. Dette kan skyldes flere ting:

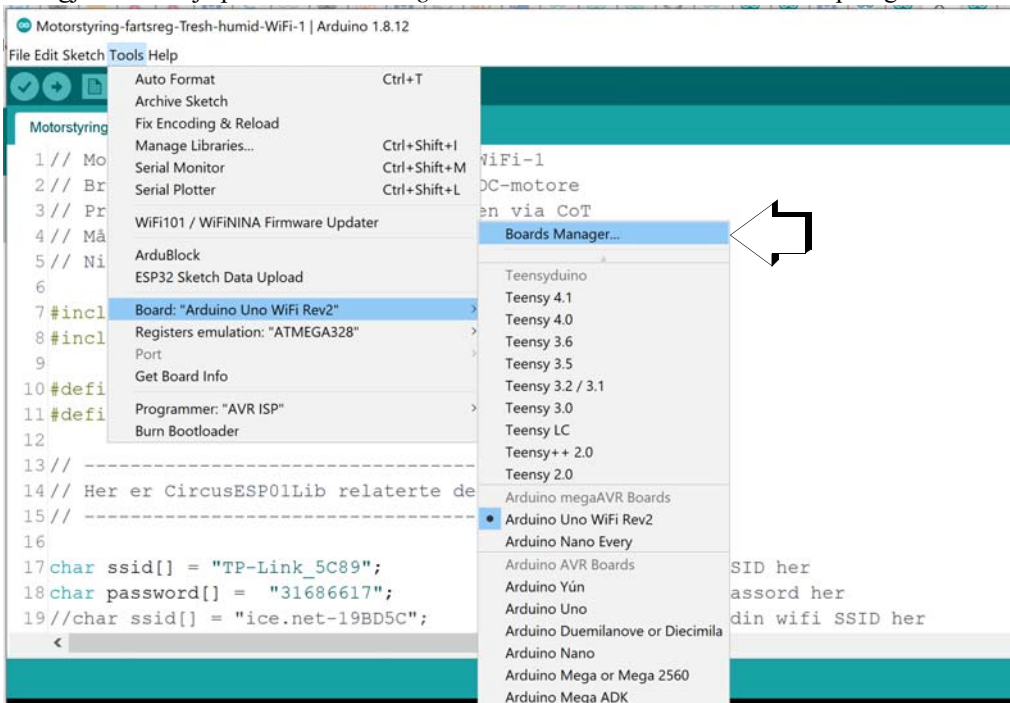


- Kabelen er ikke tilkoblet, eller defekt
- Feil port er valgt av programeditoren, som kan endres ved å velge: *Verktøy* og *Port* fra menylinjen i editoren
- Feil type mikrokontroller-kort er valgt  
Endres ved å velge: *Verktøy* og *Kort* fra menylinjen, for så å velge rett kort, i vårt tilfelle *Arduino MKR NB 1500*. Dette er først mulig etter at ekstrapakken for MKR NB 1500 kortet er installert (se avsnitt 3.2 side 51).
- Manglende drivere, i så fall må driverne installeres manuelt
- Eller rett og slett at programmet har hengt seg opp. I så fall kan man løse problemet med å lukke og starte programmet på nytt.
- Enkelte andre programmer kan ta kontroll over USB-porten, og slik blokkere for overføring til Arduino-kortet. Ett slikt program er CURA (Ultimaker)<sup>19</sup>. I så fall lukkes programmet som stenger for overføringen og man prøver å overføre Arduino-programmet på nytt.

### 3.2 Installasjon av ekstra pakke for programmering av MKR NB 1500

Vi skal nå installere programpakken vi trenger når vi skal programmere MKR NB 1500.

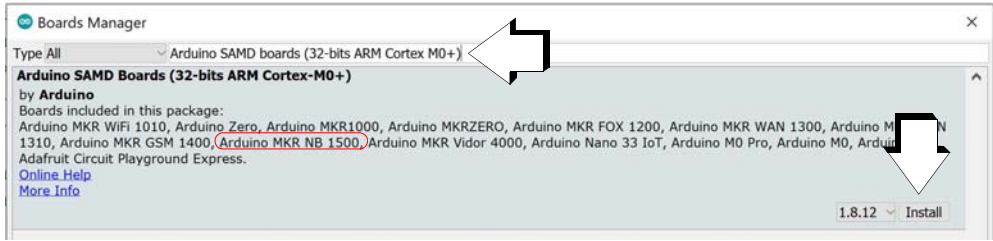
Dette gjør vi ved hjelp av *Boards Manager* som vi finner under *Tools* som vist på figuren under.



19. Kan se ut som om dette problemet er fjernet på nyere utgaver av CURA.

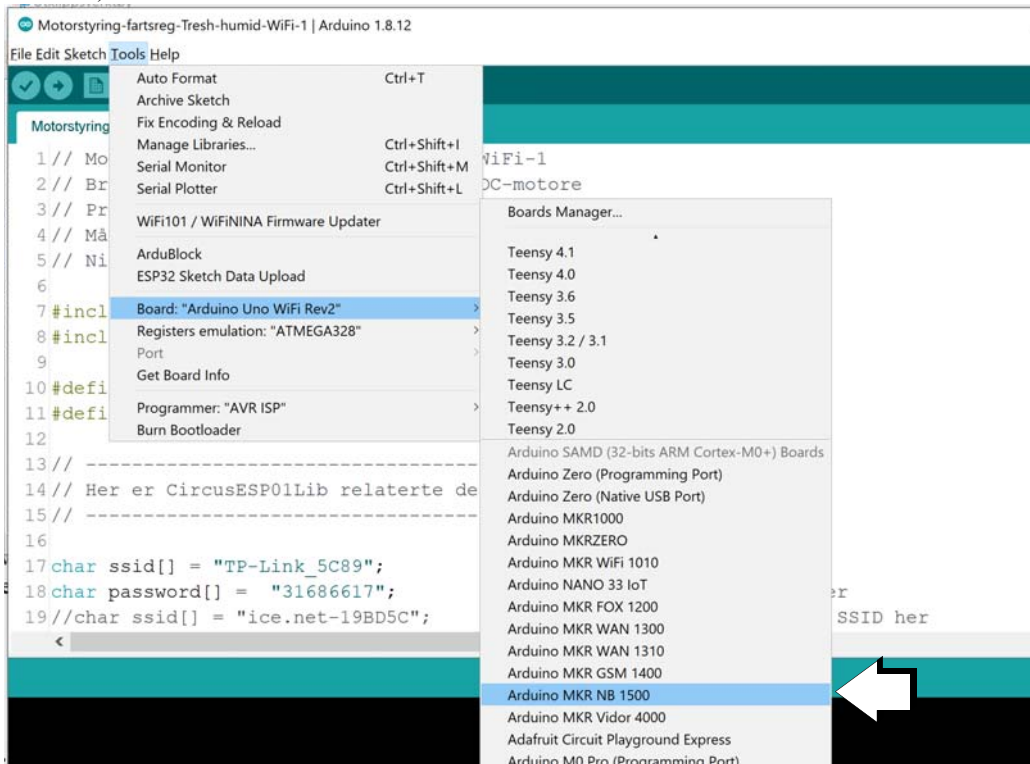


Velger vi *Board Manager* så kommer det opp et vindu med et søkefelt der vi kan skrive navnet på den familien av kort vi ønsker å installere: *Arduino SAMD Boards (32-bits ARM Cortex M0+)*. Etter noe tid kommer følgende alternativ opp:



Vi velger det kortet (rammen) som kommer opp og trykker INSTALL nederst til høyre. Programvaren for denne kort-familien består av 6 pakker og det tar litt tid å laste ned og installere den.

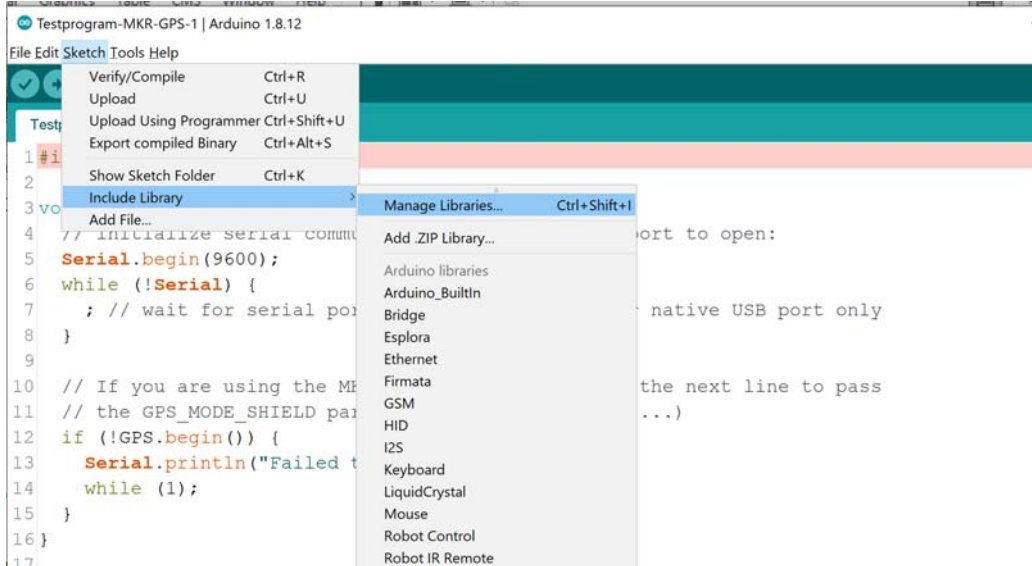
Når vi skal programmere kortet MKR NB 1500, må vi huske å velge dette kortet fra menyen av mulige kort, se figuren under. Vi velger Tools |> Boards -> Arduino SAMD(32-bits ARM Cortex-M0+) Boards -> Arduino MKR NB 1500.



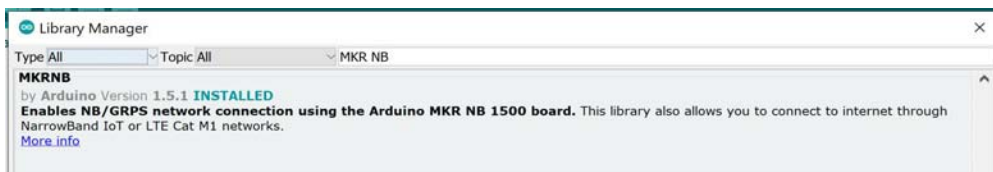


### 3.3 Installasjon av bibliotek for programmering av MKR NB 1500

Det finnes en rekke shield-kort som kan plugges på MKR NB 1500 med tilhørende biblioteker. Her skal vi i første omgang kun installere biblioteket vi trenger for å programmere MKR NB 1500 og for å kunne kommunisere via NB IoT (4G). Vi bruker da *Library manager* som vi finner ved å gå inn i menyen *Sketch/Inklude Library/Manage Libraries* som vist på figuren under:



Vi får da opp følgende vindu hvor vi skriver inn *MKR NB* i søkefeltet og får opp et alternativ kalt: *MKRNB* by Arduino Versjon 1.5.1 eller nyere. Vi velger å installere biblioteket.



Senere vil vi installere biblioteker for å håndtere de ulike shield-kortene som f.eks. MKR GPS og MKR ENV.

### 3.4 IOT-teknologier (4G) og anskaffelse av SIM-kort

Før vi kan bli operative med å samle og overføre data, må vi skaffe et SIM-kort som passer formålet. Vi har i dette tilfellet valgt å bruke Telenor som teleoperatør<sup>20</sup>. Vi må også bestemme hvilken teknologi vi ønsker å bruke. Det er spesielt to som er aktuelle<sup>21</sup>:

20. Det er flere leverandører av denne tjenesten. Com4 er en helnorsk mobiloperatør med hovedfokus på M2M- og IoT-kommunikasjon. Selskapet ble startet i slutten av 2011 og er en av få aktører som opererer med eget kjernenet for mobilproduksjon i Norge. <https://www.com4.no/loesninger/m2miot-kommunikasjon/>

21. <https://www.telenor.no/bedrift/iot/abonnement/>



## NB-IoT

*Narrowband IoT (NB-IoT) er en IoT-teknologi basert på 4G-nettet som fungerer praktisk talt hvor som helst i landet. Den egner seg godt der hvor "tilkoblede ting" er lokalisert i avsidesliggende eller vanskelig tilgjengelige områder. Med lange avstander fra den neste mobile basestasjonen, eller i skjermede områder dypt inne i bygninger eller underjordiske områder. NB-IoT passer for løsninger som har behov for å overføre små datamengder, er skreddersydd for sensorer som bare skal sende data av og til, behov for lang batterilevetid og har behov for høy dekningsgrad i grise-grendte strøk. Et godt valg ved bruk på logistikk av fraktgods, containere, transportmidler, jordbruk, smarte byer, smarte hjem og sporing av mennesker og dyr. Telenor har en befolkningsdeknning på 99,9% for NB-IoT.*

Et naturlig spørsmål er hva Telenor legger i begrepet *befolkningsdeknning*, da vårt behov gjerne ligger utenfor områder der det bor folk.

## LTE-M

*LTE-M er også basert på 4G-nettet, og er en IoT-teknologi som ligner på NB-IoT. LTE-M gir imidlertid noe høyere hastigheter og er beregnet på overføring av større datamengder enn NB-IoT. LTE-M har forbedret batterilevetid og deknning, samt etterhvert mulighet for tale over nettverket. Et godt valg ved bruk av mer aktive sensorer som f.eks. sporing av eiendeler, smartklokker, autonome kjøretøyer, smartmålinger, medisinsk utstyr og hjemmesikkerhetsapplikasjoner.*

Begge disse er tilgjengelige ved bruk av MKR NB 1500. Man kan også be om råd med tanke på om man bør velge NB-IoT eller LTE-M. Prisen er avhengig av datamengden som skal overføres, men er ikke høy, fra 8 – 20 kr./mnd.<sup>22</sup>

NB IoT har et etableringsgebyr på kr. 15,-. Deretter er kostnaden kr. 8,- pr måned som inkluderer 1MB overført data. Bruker man mer enn dette betaler man ca. kr. 1,50 i tillegg pr. MB.

Med bakgrunn i en middels stor datamengde, 50 – 300 byte pr. melding, inntil 12 ganger i timen, for utendørs bruk og med vekt på batteriøkonomisering, anbefales NB-IoT. I denne sammenheng opptrer vi opptre som bedrift.

22. <https://bedriftsbutikk.telenor.no/m2m/>

## Velg ditt IoT abonnement

Vi har to IoT-abonnement du kan velge mellom. IoT Total passer for alle ting og sensorer med høy hastighet. NB IoT er optimalt for bruk av sensorer med lavt strøm- og dataforbruk.

IoT Total	NB IoT
	
<p>9 av 10 kunder har dette</p>	
<ul style="list-style-type: none"><li>Passer for alle ting og sensorer</li><li>Fleksibelt abonnement som tilpasser seg faktisk databruk</li><li>Gir deg høy hastighet</li><li>Mulighet for 4G tale (VoLTE) og SMS</li><li>APN-lås som beskytter mot uønsket trafikk</li></ul>	<ul style="list-style-type: none"><li>Passer for sensorer med lavt batteri, strøm- og dataforbruk</li><li>Bedre deknning enn vanlig 4G</li><li>Opptil 10 års levetid på batteriene</li><li>1MB er inkludert i måneden, deretter betaling per MB brukt</li><li>NB! Det er ikke mulig med SMS/tale</li></ul>
<p>Frå kr 29,- / mnd</p>	<p>Frå kr 8,- / mnd</p>



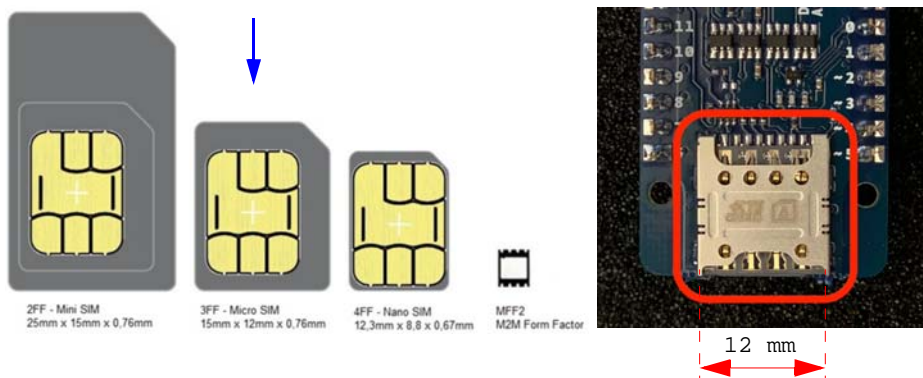
Med bakgrunn i dette kan vi gå til anskaffelse av SIM-kort:

<https://bedriftsbutikk.telenor.no/order-setup>

eller

<https://bedriftsbutikk.telenor.no/m2m/> (dette gir oversikt over flere alternativer)

Ved bestilling blir man bedt om å spesifisere SIM-kortets *formfaktor*, dvs. utformingen av kortet som må passe til den holderen MKR NB 1500 har. Sammenligner vi med holderen på MKR NB 1500 så ser vi at det er 3FF varianten, som har en bredde på 12 mm, som passer til vårt teknologiske valg.



### 3.4.1 Bestilling av et antall fra 1 – 9

Gå til adressen: <https://bedriftsbutikk.telenor.no/m2m>

Velg: Bedrift og IoT/M2M og NB IoT (Return)

Velg: Velg formfaktor: Micro SIM (3FF)

Antall SIM-kort: 8

Skal testmengder legges på?: Ja

PIN-kode for SIM-kort: Deaktivert/Ingen PIN

Klistremerke med informasjon?: Ja

Kommentar: Ingen

Utenlandssending: Nei

Velg: Til bedriftsopplysninger

Velg: Oppgi org. nr. og hent bedriftsopplysninger

Velg og gå til kassen og betal. Dette er ikke en engangsinvestering, men en lisens som må betales for hvert kvartal eller lignende.

### 3.4.2 Bestilling av et antall fra 10 eller flere

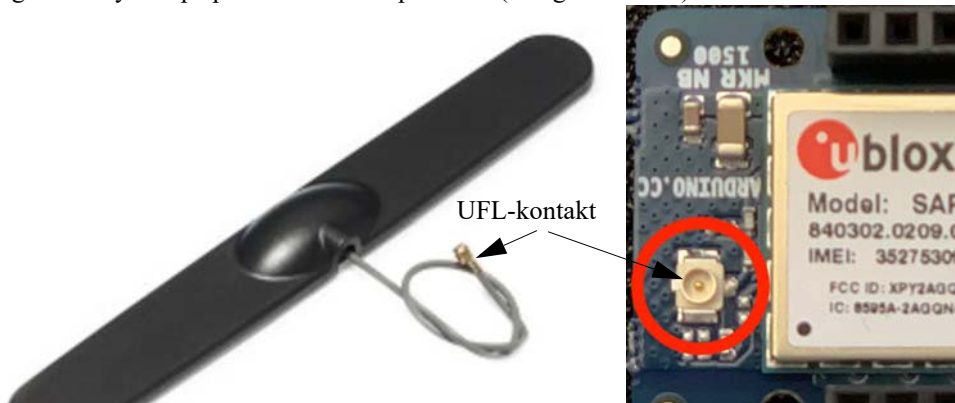
Dersom antallet kort er lik eller større enn 10 så må man opprette en bedriftskonto og være en autorisert bestiller i bedriften. Man går først til adressen: <https://www.telenor.no/bedrift/min-bedrift/> hvor man blir registrert som bedriftens autoriserte innkjøper.



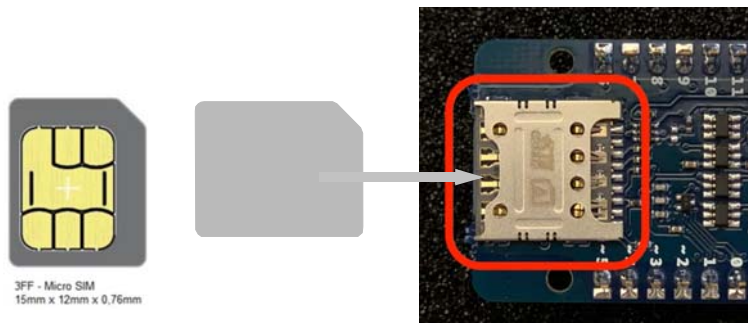


### 3.5 Montering av antenne og SIM-kort

Når vi har mottatt SIM-kortet og elektronikken, er vi klare til å montere antennen og SIM-kortet. Vi har valgt en liten dipolantenne med en UFL-kontakt som er særdeles liten og kan by på utfordringer å få trykket på plass i kontakten på kortet (se figuren under).



SIM-kortet skyves inn i kortholderen på undersiden av kretskortet. Det er så vidt antydnet hvilken vei kortet skal skyves inn i holderen. Legg merke til at kontaktpunktene på kortet skal vende nedover, dvs. inn mot kretskortet.



Vi er nå klare til å bestemme *kortets identitet* som vi kan bruke for å identifisere bruken av tjenesten hos Telenor. Dette er ikke nødvendig for vår anvendelse.

### 3.6 Installasjon av biblioteker og kjøring av eksempelprogrammet

Vi skal i dette avsnittet laste opp og teste ut Eksempelprogrammet: `EVU-kurs23-Eksempel-program.ino` Programmet kan utføre følgende funksjoner:

- Leser av temperatursensoren DS18B20 (trenger to biblioteker)
- Leser av GPS-mottakeren (trenger eget bibliotek)
- Bygger opp datastreng for overføring og lagring
- Skriver til SD-kort-terminal (trenger eget bibliotek)



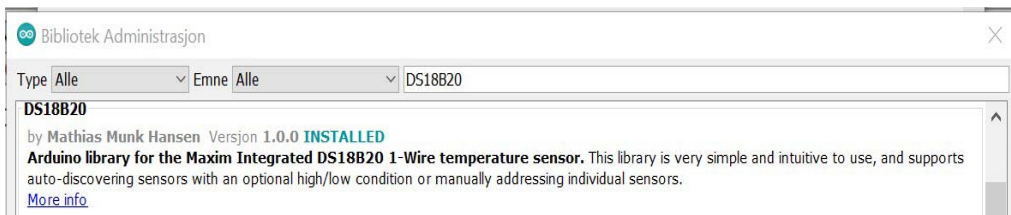
- Sender over data via 4G til server (biblioteket skal være installert under deloppdrag 1, avsnitt 3.1 på side 49).
- Holder styr på den programvare-baserte “vakhunden” som passer på å restarte programmet dersom det henger seg (trenger eget bibliotek).
- Legger mikrokontrolleren i dvale (trenger et eget bibliotek)

### 3.6.1 Installasjon av biblioteker

#### Installer bibliotekene til temperatursensoren DS18B20 og entrådsbussen

For å kunne bruke DS18B20 må vi installere to biblioteker: Ett for bruk av entråd buss (oneWire.h) og en for sensoren (DS18B20.h).

**DS18B20:** Gå til: *Skisse > Inkluder bibliotek > Administrer bibliotek* og skriv DS18B20 i søkefeltet og velg biblioteket DS18B20 skrevet av **Mathias Munk Hansen**:

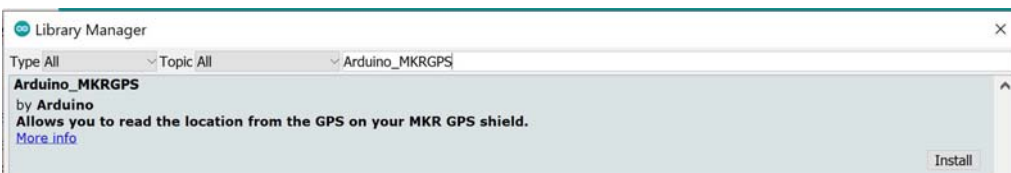


**OneWire:** Gå til: *Skisse > Inkluder bibliotek > Administrer bibliotek* og skriv OneWire i søkefeltet og velg biblioteket OneWire skrevet av **Paul Stoffregen**:



#### Installer biblioteket til GPS-mottakeren

Gå til: *Skisse > Inkluder bibliotek > Administrer bibliotek* og skriv *Arduino\_MKRGPS* i søkefeltet og velg å installere biblioteket som kommer opp som forslag.





## Installer biblioteket til SD-kort terminalen

Dette ligger ved når vi installerer biblioteket til Arduino MKR ENV som også har en SD-kort terminal.

Gå til: *Skisse > Inkluder bibliotek > Administrer bibliotek* og skriv *Arduino\_MKRENV* i søkefeltet og velg å installer biblioteket som kommer opp som forslag.



## Installer biblioteket for dvalefunksjoner

For å kunne legge mikrokontrolleren i dvale så må vi installere biblioteket: *ArduinoLowPower*. Gå til: *Skisse > Inkluder bibliotek > Administrer bibliotek* og skriv *Arduino Low Power* i søkefeltet og velg å installer biblioteket som kommer opp som forslag.



## Installer biblioteket til programvare baserte “vakthunden” (WDTZero)

Dette biblioteket må vi installere manuelt. Det gjør vi på følgende måte:

- En zip-utgave av biblioteket kan lastes ned fra denne siden:  
<https://github.com/javos65/WDTZero/archive/refs/heads/master.zip>
- Legg zip fila på et sted der du finner den igjen
- Gå til: *Skisse > Inkluder bibliotek > Legg til .ZIP Bibliotek* og finn zip biblioteket du nettopp lagret, og installer biblioteket

Eksempelkoder på bruk av “vakthunden” finnes bl.a. her:

<https://github.com/javos65/WDTZero/blob/master/examples/WDTZero1/WDTZero1.ino>

### 3.6.2 Installasjon av eksempelprogrammet (EVU-kurs23-Eksempelprogram.ino)

Til dette bruker vi det generelle testprogram (EVU-kurs23-Eksempelprogram.ino) som inneholder det vi trenger for å lese av temperatursensoren og GPS-mottakeren og legge dataene ned på serveren.

Hent testprogrammet: *EVU-kurs23-Eksempelprogram.ino* (F side 267). Eller helst hent programmet fra nettsiden:



[www.ntnu.no/skolelab/bla-hefteserie](http://www.ntnu.no/skolelab/bla-hefteserie)

Gå ned til overskriften: *Programmering av Tingenes internett (IoT)*

Åpne fanen: *EVU-kurs: Miljøovervåking med havbøye*

Last ned og installer programmet: *Ferdig\_program\_med\_PICAXE\_MILLIS\_TEST\_2.ino*

Det neste avsnittet går gjennom programkoden linje for linje slik at vi forstår hva programmet gjør.

### 3.7 Programkoden (EVU-kurs23-Eksempelprogram.ino)

Dette avsnittet skal være en hjelp til å forstå programkoden slik at man er istand til å gjøre personlige justeringer.

#### 3.7.1 Detaljert gjennomgang av programkoden

Vi skal nå gå gjennom eksempelprogrammet og forklare de ulike delene.

##### Inkludering av biblioteker

Følgende biblioteker trengs:

```
#include <DS18B20.h> // Bibliotek for avlesning av temperatursensoren
#include <OneWire.h> // Bibliotek som etablerer entrådss buss til temp. sensoren
#include <Arduino_MKRGPS.h> // Bibliotek som leser av GPS
#include <MKRNB.h> // Bibliotek som overfører data til server via GPRS
#include <Arduino_MKRENV.h> // Bibliotek for lesing av miljøkortet ENV om det brukes
#include "ArduinoLowPower.h" // Bibliotek som legger MKR i dvale
#include <WDTZero.h> // Bibliotek som håndterer vakthunden
#include <SD.h> // Bibliotek for håndtering av SD-kort
#include <SPI.h> // Bibliotek for håndtering av serie-bus
```

Bibliotekenes header-filer (.h) plasseres normalt i toppen av programmet. Dersom man ønsker å legge dataene inn på et SD-kort, så finner man en slik terminal både på miljø-shieldet (MKR ENV) og på Prototypkortet (MKR SD Proto Shield). Bruker man miljø-shieldet må man ta med biblioteket for dette kortet i tillegg til at man må initialisere det i `setup()`-funksjonen, se under `setup()`-funksjonen.

##### Deklarasjon av objekter

```
WDTZero MyWatchDoggy; // Deklarerer instansen til vakthunden MyWatchDoggy av typen WDTZero
NBClient client; // Deklarerer instansen som opererer mot serveren
GPRS gprs; // Deklarerer instansen som opprettet kontakt med GPRS-nettet
NB nbAccess; // Deklarerer instansen som gir tilgang til GPRS-nettet
DS18B20 ds(0); // Deklarerer instansen som kommuniserer med temp. sensoren
File myFile; // Deklarerer et objekt for filbehandling, kun ved lagring på SD-kort
```

*Objekter* er sentrale i C++ som er et objektorientert språk. Et objekt eller instans er en deklarasjon på linje med deklarasjon av variabler. Mens en variabel gjerne inneholder en verdi eller et array av verdier av samme type, så kan et objekt inneholde et pakke med ulike variabler av ulike typer. Ja, ikke bare variabler med også ulike *funksjoner*. Når vi deklarerer våre objekter gir vi et *navn* til



en *pakke* av en bestemt type. F.eks. så deklarerer vi et objekt kalt *MyWatchDoggy* av typen *WDT-Zero*. For å nå de ulike elementene i objektet så bruker vi pakkenavnet, punkt og navnet på variabelen eller funksjonen f.eks. *MyWatchDoggy.setup(WDT\_SOFTCYCLEIM)*;

## Deklarasjon av konstanter

I motsetning til variabler som kan endre innhold mens programmet kjører, så må konstantene beholde sitt innhold gjennom hele programmet. De kan bare endres i selve programkoden.

```
#define DEBUG           // Kommenter bort denne dersom Serial.print ikke skal inkluderes
#define LED             // Kommenter bort denne dersom LED indikasjon ikke ønskes
```

Konstanter plasseres gjerne før *setup()*-funksjonen og vil derfor være globale, dvs. kunne brukes i alle funksjoner. Konstantene *DEBUG* og *LED* har spesielle funksjoner. Avhengig av om vi deklarerer hver av disse to så påvirker dette hva som kan vises i monitoren. Vi kommer snart tilbake til dette.

## Deklarasjon av variabler

Variabler kan endre verdi mens programmet kjører. Dersom de defineres utenfor alle funksjoner blir de globale og kan brukes i alle funksjoner. Vi har her valgt å gruppere dem i henhold til i hvilken sammenheng de brukes i:

```
// Temperatursensor
uint8_t address[] = {40, 250, 31, 218, 4, 0, 0, 52};
uint8_t selected;

// Oppkobling til server
char server[] = "sensor.marin.ntnu.no"; // Navnet på serveren der data legges
char path[] = "/cgi-bin/tof.cgi?";     // Angir cgi-kode for plassering i file
char filename[]="NTNU-1.txt";         // Navnet på fil for lagring av data
int port = 80;                        // Port 80 er default for HTTP
boolean connected = false;            // Status oppkobling til GPRS
char buffer[128];                     // Deklarasjon av buffer med tilhørende lengde

volatile unsigned long num = 0;       // Måling nummer
float w_temperature = 0;              // Vanntemperatur
float BatVolt = 0;                   // Variabel for å holde batterispenningen
int Pause = 5000;                    // En evt. pause i programloopen
int SleepTime = 5000;                // Tid for dvale

float GPSlat = 0;                    // GPS breddegrad
float GPSlong = 0;                   // GPS lengdegrad
unsigned long epochTime = 0;         // GPS tidsstempel fra 1.1.1980
int numSat = 0;                      // GPS antall satellitter
int maxNoChecks = 5000;              // Antall runder for lesing av GPS-data
```

Disse deklarasjonene plasseres gjerne før *setup()*-funksjonen.

## Deklarasjon av porter

Her kan vi sette navn på de portene vi ønsker å bruke.



```
int BatVoltPin = A3;           // Analog port for tilkobling av batterispenning
const int SD_CS_PIN = 4;      // Port for styring av SD-kort terminal, fabrikkbestemt
```

Port 4 er fabrikk-koblet til SD-kort-terminalen på shield-kortene så denne kan ikke endres.

## Setup()-funksjonen

Som kjent kjøres denne funksjonen kun ved oppstart og inkluderer initiering av de ulike objektene og lignende:

```
void setup() {

    pinMode(LED_BUILTIN, OUTPUT); // Port for innebygget LED, definert som utgang
    analogReadResolution(12);     // Angir oppløsning for AD-konverter

    #ifndef DEBUG
        Serial.begin(115200);     // Setter opp data hastighet til monitor
    #endif

    if (!GPS.begin())             // Initialisering av GPS
    {
        #ifndef DEBUG
            Serial.println("GPS initialisering mislyktes");
        #endif
    }
    SPI.begin();
    ENV.begin();

    if(!SD.begin(SD_CS_PIN))
    {
        Serial.println("Initialisering SD terminal mislyktes");
    }

    // Lag en file med navnet DataFile.txt
    myFile = SD.open("DataFile.txt", FILE_WRITE);
    delay(1000);
    myFile.close();
    delay(100);

    // Initialiserer dvale-funksjon
    LowPower.attachInterruptWakeup(RTC_ALARM_WAKEUP, dummy, CHANGE);

    // Initialisering av "vakthund"
    MyWatchDoggy.attachShutdown(myshutdown); // Starter funksjonen myshutdown() ved reset
    MyWatchDoggy.setup(WDT_SOFTCYCLE1M);    // Setter intervalltiden for "vakthunden"

    // Opprett tilkobling til 4G - GPRS-tjeneste
    connectToGPRS();

    delay(1000);
}
```



Som vi ser kalles det en rekke funksjoner som initialiserer eller setter parametere for de funksjonene vi senere skal bruke i programmet. Kommandoen `analogReadResolution(12);` setter f.eks. oppløsningen til AD-konverteren til 12 bit. Om denne utelates vil oppløsningen automatisk settes til 10 bit.

Ellers initialiseres GPS-mottakeren (`GPS.begin()`), SPI-bussen (`SPI.begin()`) som kommuniserer med SD-kortet, miljø-kortet (MKR ENV) dersom dette brukes som SD-kortterminal (`ENV.begin()`), selve SD-kortterminalen (`SD.begin(SD_CS_PIN)`), og det opprettes en fil hvor dataene skal legges dersom man lagrer dataene på SD-kort (`SD.open("DataFile.txt", FILE_WRITE);`).

Vi legger spesielt merke til følgende programlinjer som vi finner igjen flere steder i programmet:

```
#ifdef DEBUG
    Serial.println("GPS initialisering mislyktes");
#endif
```

Dette er en måte å gi beskjed til kompilatoren om utskrifter ala `Serial.println ...` ønskes tatt med i programmet eller ikke. Dersom konstanten `DEBUG` er definert under deklarasjon av konstanter, så blir programkoden mellom `#ifdef DEBUG ...` og `#endif` tatt med i programmet. Dersom `DEBUG` ikke er deklartert så utelates programlinja av kompilatoren. Dette har vi gjort da det er lite hensiktsmessig å forsøke å skrive til monitoren dersom bøya ligger i sjøen og dermed ikke kan kommunisere over USB-kabelen. Ja, det kan til og med se ut til at programmet har en tendens til å henge seg om det forsøker å skrive til monitoren uten at USB-kabelen er tilkoblet.

## Loop()-funksjonen

Som vi ser så inneholder `loop()`-funksjonen nesten bare kall av funksjoner, som enten ligger i et bibliotek eller som vi selv har definert under.

```
void loop() {

    //readGPSdata(); // Inkluder GPS
    readWaterTemp(); // Inkluder måling av temp. i vann
    //readBatVolt(); // Inkluder måling av baterispennning om den er implementert

    makeString(); // Bygge opp bufferet for overføring av data
    sdPrint(); // Skriv data til SD-terminal
    connectToServer(); // Koble opp mot server og overfør data

    printData(); // Skriv data til monitoren
    printDataString(); // Skriv ut bufferet til monitoren
    MyWatchDoggy.clear(); // Resetter vakthunden

    num++;
    //GoToSleep(); // Legg mikrokontrolleren og GPS'en i dvale
    delay(Pause); // Ev. legg inn en pause i loopen
}
```





På denne måten kan vi inkludere de funksjonene vi ønsker å teste ut og på den måten gradvis utvide funksjonaliteten til programmet. F.eks. har vi i dette tilfellet kommentert bort avlesning av batterispenningen (`//readBatVolt();`) og GPS-mottakeren (`//readGPSdata1();`), og foreløpig droppet å legge mikrokontrolleren i dvale (`//GoToSleep();`).

La oss se på de enkelte egendefinerte funksjonene:

## Egendefinerte funksjoner

### *Funksjon for avlesning av vanntemperatur*

```
void readWaterTemp()
{
    w_temperature = ds.getTempC();
}
```

`ds.getTempC()` er en biblioteksfunksjon som leser av temperaturen. Verdien legges i variabelen `w_temperature`.

### *Funksjon for avlesning av batterispenningen*

```
void readBatVolt()
{
    BatVolt = analogRead(ADC_BATTERY)*(4.25/4096);
}
```

Her leser vi av den analoge inngangen (`ADC_BATTERY`) og regner den om til en spenning. Inngangen er koblet til en intern spenningsdeler på mikrokontrollerkortet og vil bare fungere dersom vi har koblet til et batteri. Legg merke til at vi multipliserer den avleste verdien med 4.25. Årsaken er at forsøk har vist at denne verdien gir den riktigste verdien når vi sammenligner den internt målte verdien på pinne `ADC_BATTERY`, med målt spenning på batteriet med et voltmeter.

Vi kan også koble opp vår egen eksterne måling av spenningen på batteriet, i så fall må vi sørge for å bruke en spenningsdeler slik at vi unngår spenninger som overgår driftspenningen til mikrokontrolleren på 3,3V. Det kan være praktisk å bruke en spenningsdeler som halverer batterispenningen da et nyladet batteri kan ha en spenning på opp til 4,2V. Når målingen er utført så må vi multiplisere resultatet med to for å få riktig spenning som vist i programkoden under:

```
void readBatVolt()
{
    BatVolt = 2*analogRead(BatVoltPin)*3.3/4096;
}
```

Her har vi brukt en av de analoge inngangene hos mikrokontrolleren som vi har kalt `BatVoltPin`.

En analog-til-digitalkonverter (ADC) omformer spenningen til et digitalt tall. Med 12 bits ADC vil maksimalspenningen på 3,3V tilsvare tallet 4096. Vi finner da spenningen i Volt ved å dele måleverdien med 4096 og multiplisere med arbeidsspenningen på kortet, 3,3V.



## Funksjon for avlesning av GPS-mottaker

```
void readGPSdata1()
{
  int noChecks = 0;

  for (int i = 0; i < maxNoChecks; i++)
  {
    // Check if there is new GPS data available
    if (GPS.available())
    {
      // If there is, read GPS values
      GPSSlat = GPS.latitude(); // Avlesning av breddegrad
      GPSSlong = GPS.longitude(); // Avlesning av lengdegrad
      epochTime = GPS.getTime(); // Avlesning av epoketid, ant. sek. siden 1.1.1980
      numSat = GPS.satellites(); // Kontakt med antall satellitter

      #ifdef DEBUG
        Serial.print("Antall GPS sjekk: ");
        Serial.println(noChecks);
      #endif

      break;
    }

    noChecks++;

    #ifdef DEBUG
      if (noChecks == maxNoChecks) Serial.println("Mislykket henting av posisjon fra GPS");
    #endif
  }
}
```

Legg merke til at avlesningen av GPS-mottakeren er lagt inn i en for()-loop med et maksimalt antall gjentakelser på `maxNoChecks`. Dette har vi gjort fordi det kan ta litt tid å etablere forbindelse med satellittene. Ved å legge inn et maksimalt antall forsøk på avlesning, unngår vi at programmet henger seg og forårsaker at “vakhunden” resetter mikrokontrolleren. I tillegg teller vi hvor mange runder den må gå for å få gyldige data, evt. varsler at maksimalt antall avlesninger er nådd slik at denne gangen er avlesningen mislykket. Deretter går programmet videre og forsøker på nytt i neste runde.

## Funksjon for oppbygging av databufferet

```
void makeString()
{
  sprintf(buffer, "/cgi-bin/tof.cgi?s,%u;%u;%0.6f;%0.6f;%i;%0.1f;%0.2f", file-
    name, num, epochTime, GPSSlat, GPSSlong, numSat, w_temperature, BatVolt);
}
```



Her bygges innholdet av bufferet opp som en tekststreng som består av måleverdier skilt med semikolon. *Det kan bemerkes at dersom vi planlegger å lese datafila med Python så kan det være greit å bruke komma istedet for semikolon.* Vi skal se nærmere på dette i avsnitt 3.7.2.

### ***Funksjon for utskrift av databufferet til monitoren***

```
void printDataString()
{
  #ifdef DEBUG
  Serial.println(buffer);
  Serial.println();
  #endif
}
```

Vi legger merke til utskriften kun gjøres dersom DEBUG er definert. Utskriften blir lik den som skrives til datafila på serveren.

### ***Funksjon for tilkobling til NB-IoT-nettet (4G GSM)<sup>23</sup>***

```
void connectToGPRS()
{
  while (!connected)
  {
    if ((nbAccess.begin("", true, true) == NB_READY) && (gprs.attachGPRS() == GPRS_READY))
    {
      connected = true;

      #ifdef DEBUG
      Serial.println("Vellykket tilkobling til GPRS");
      #endif

      #ifdef LED
      flash(1);
      #endif
    }
    else
    {
      #ifdef DEBUG
      Serial.println("Mislykket tilkobling til GPRS");
      #endif

      #ifdef LED
      flash(5);
      #endif

      break;
    }
  }
}
```

---

23.<https://github.com/arduino-libraries/MKRNB/blob/master/docs/api.md>



```
}
```

Essensen i denne funksjonen kan gjengis slik:

```
void connectToGPRS()
{
  while (!connected)
  {
    if ((nbAccess.begin("", true, true) == NB_READY) && (gprs.attachGPRS() == GPRS_READY))
    {
      connected = true;
    }
  }
}
```

resten er bare meldinger som forteller om det har lyktes å koble opp eller ei.

`nbAccess.begin("", true, true)` ber om tilgang til nettverket ved hjelp av SIM-kortets PIN-kode. Om slikt ikke brukes som hos oss, kan første argument stå åpent. Andre argument (`true`) handler om synkron (`true`) eller asynkron (`false`) modus, mens siste argument angir om radiomodemet skal restartes (`true`) eller ikke. Dersom den gis tilgang til nettet returnerer funksjonen `NB_READY` (alternativt kan en av følgende koder returneres: `GPRS_READY`, `TRANSPARENT_CONNECTED`, `CONNECTING`, `IDLE` og `ERROR`).

`gprs.attachGPRS()` ber om å bli koblet opp mot GPRS-nettverket. Oppkobling er gjennomført tilfredsstillende når funksjonen returnerer `GPRS_READY`.

Når begge disse er akseptert så settes variabelen `connected = true;` slik at man ved neste overføring av data, slipper å koble opp på nytt. Imidlertid kan det være nødvendig dersom mikrokontrolleren har ligget i dvale. Imidlertid kan det være nødvendig dersom mikrokontrolleren har ligget i dvale.

Dersom vi kjører med frakoblet USB-kabel og batteri, så vil meldingene til monitoren utebli. Vi har derfor laget en `flash()`-funksjon, hvor argumentet angir et antall blink.

```
#ifdef LED
  flash(1);
#endif
```

Ved at vi deklarerer konstanten `LED` så gir den innebygde LED'en på kortet fra seg ett eller flere blink som indikerer hvor i programmet den befinner seg, og om oppkoblingen var vellykket (1 blink) eller ikke (5 blink).

### ***Funksjon for tilkobling til serveren ved Inst. marin teknikk***

Denne funksjonen foretar oppkobling og overføring av data til serveren.

```
void connectToServer()
{
  if (client.connect(server, port))
  {
    client.print("GET "); // Gjør et HTTP request:
    client.print(buffer);
  }
}
```



```
client.println(" HTTP/1.1");
client.print("Host: ");
client.println(server);
client.println("Connection: close");
client.println();

#ifdef DEBUG
  Serial.print("Vellykket overføring til server av datasett nr.: ");
  Serial.println(num);
#endif

#ifdef LED
  flash(2);
#endif
}
else
{
  #ifdef DEBUG
    Serial.print("Mislykket overføring til server av datasett nr.: ");
    Serial.println(num);
  #endif

  #ifdef LED
    flash(10);
  #endif
}
}
```

Essensen i denne funksjonen kan også gjengis i kortform, resten er bare meldinger om hvordan leveransen av dataene til serveren har gått:

```
void connectToServer()
{
  if (client.connect(server, port))
  {
    client.print("GET "); // Gjør et HTTP request:
    client.print(buffer);
    client.println(" HTTP/1.1");
    client.print("Host: ");
    client.println(server);
    client.println("Connection: close");
    client.println();
  }
}
```

`client.connect(server, port)` kobler opp mot serveren (`server`) med port nr. (`port`). Begge argumentene er gitt innhold under deklarasjonen av variabler:

```
char server[] = "sensor.marin.ntnu.no"; // Navnet på serveren der data legges
char path[] = "/cgi-bin/tof.cgi?"; // Angir cgi-kode for plassering i file
```



Derneft sendes en HTTP forespørsel til klienten (serveren): GET <pathname> HTTP/1.1.  
Denne er delt opp i tre print-kommandoer:

```
client.print("GET ");
client.print(buffer);
client.println(" HTTP/1.1");
```

der `buffer` også inneholder stien.

```
client.print("Host: ");
client.println(server);
client.println("Connection: close");
client.println();
```

Så angis server-navnet før forbindelsen lukkes, og kommandorekka avsluttes med en tom linje.

### ***Funksjon for skriving til SD-kort***

I vårt eksempelprogram skrives bufferet til filen `DataFile.txt` på SD-kortet:

```
void sdPrint(){

    myFile = SD.open("DataFile.txt", FILE_WRITE);
    delay(1000);

    if (myFile)
    {
        myFile.println(buffer);
        myFile.close();
    }
}
```

Det er det samme bufferet som skrives til serveren.

### ***Funksjon for å gi lysblink***

Som nevnt foran vil det være vanskelig å få skriftlige meldinger når mikrokontrolleren er fra-koblet PC'en. Funksjonen `flash()` gir et antall lysblink i mikrokontrollerens innebygde lysdiode for å signalisere at den passerer ulike punkter i programmet. Når funksjonen kalles, vil man i argumentet kunne angi et antall blink.

```
void flash(int number)
{
    delay(1000);
    for(int i=0; i<number; i++)
    {
        digitalWrite(LED_BUILTIN, HIGH);
        delay(100);
        digitalWrite(LED_BUILTIN, LOW);
        delay(100);
    }
    delay(1000);
}
```



```
}
```

Vi har valgt å gi *ett* blink ved oppnådd kontakt med NB IoT-nettverket, *to* når det oppnås kontakt med serveren og *tre* når mikrokontrolleren legges i dvale. Det angis 5 og 10 blink ved mislykket oppkobling mot henholdsvis NB IoT-nettverket og serveren.

### ***Funksjon som legger mikrokontrolleren i dvale***

Funksjonen `GoToSleep()` legger mikrokontrolleren i dvale for en viss tid (`SleepTime`). Man må regne med at radio forbindelsen brytes ved slik nedkobling.

```
void GoToSleep() // Legg mikrokontrolleren og GPS'en i dvale
{
    #ifdef DEBUG
        Serial.println("Går i dvale");
    #endif

    #ifdef LED
        flash(3);
    #endif

    GPS.standby();
    LowPower.deepSleep(SleepTime);
}
```

De to siste kommandoene `GSP.standby();` og `LowPower.deepSleep(SleepTime);` slår av GPS'en og legger mikrokontrolleren i dvale.

### ***Funksjon for retur fra dvale***

```
void dummy()
{
    GPS.wakeup();
    NVIC_SystemReset(); // Resett programmet
}
```

Idet mikrokontrolleren vekkes fra dvale, havner den inn i funksjonen `void dummy()`, der de første tiltakene etter oppvåkning kan gjøres. Her har vi valgt å vekke GPS'en (`GPS.wakeup();`) og resette programmet (`NVIC_SystemReset(); // Resett programmet`)<sup>24</sup>.

### ***Funksjon for siste aksjon før "vakhunden" resetter programmet***

```
void myshutdown()
{
    #ifdef DEBUG
```

---

24.[https://devzone.nordicsemi.com/f/nordic-q-a/77006/what-does-nvic\\_systemreset-actually-do](https://devzone.nordicsemi.com/f/nordic-q-a/77006/what-does-nvic_systemreset-actually-do)





```
Serial.println("Resetter MKR");
#endif
}
```

Her nøyer vi oss med å varsle ("Resetter MKR") om at mikrokontrolleren resettes av den interne "vakhunden".

### 3.7.2 Bygg opp tekststreng for overføring av en måleserie

Vi skal nå bygge opp tekststrengen som beskriver formatet til en rad i datafilen som skal legges på serveren. Vi tar utgangspunkt tekststrengen i vårt eksempel:

```
sprintf(buffer, "/cgi-bin/tof.cgi?%s,%u,%u,% .6f,% .6f,%i,% .1f,% .2f", file-
name, num, epochTime, GPSSlat, GPSSlong, numSat, w_temperature, BatVolt);
```

La oss ta for oss hvert av leddene i eksempelet over:

`sprintf()` står for "string print format" og lager en tekststreng, satt sammen av tekst og tallverdier omgjort til tekst.

`buffer`, som er det første elementet i argumentet, angir variabelen der den endelige tekststrengen skal plasseres. Denne er tidligere deklartert som et buffer på inntil 128 karakterer (Byte), men som bør gjøres så kort som mulig. Selve tekststrengen er plassert i hermetegn: " " .

`/tof.cgi?` er et lite skript (program) som web-serveren bruker for å styre data på serveren ("cgi" står for *Common Gateway Interface*). I vårt tilfelle vil skriptet ta tak i det neste argumentet etter "?", som i dette eksempelet er "%s, ". Der %-tegnet kalles *formatspesifikasjonen* ("format specifier") og "s" *formatkarakteren* ("format character"). Lengre bak på linja ligger variabelen som skal anvende formatspesifikasjonen, i dette tilfellet strengen `filnavn`. Alternativt kunne vi ha skrevet filnavnet rett inn i tekststrengen.

Stedene i tekststrengen der vi ønsker å plassere tekst eller måleverdier, angis derfor med formatspesifikasjoner av ulike slag og som harmonerer med hver av variablene vi ønsker å legge inn. Skal vi legge til et desimaltall, så angis dette som f.eks. ".nf". Dvs. at ".6f" betyr at desimaltallet skal angis med 6 desimaler. "f" angir at det er snakk om et flyttall, "u" et heltall uten fortegn, "i" et heltall med fortegn, "d" et desimaltall og "s" at det handler om en tekststreng som vist under.

```
%s,%u,%u,% .6f,% .6f,%i,% .1f,% .2f
```

Etter hermetegnet kommer listen over variabler som skal settes inn i tekst-strengen på de angitte plassene. I vårt tilfelle er dette:

```
filename, num, epochTime, GPSSlat, GPSSlong, numSat, w_temperature, BatVolt
```

Man kan eventuelt sette inn rene tall (67.5, 5.6), eller tekst, men formatspesifikasjonen må stemme med typen til variabelen.

Vi har her separert leddene med ",". Grunnen er at det påstås at ved bruk av Python så vil lesingen bli enklere dersom vi bruker komma framfor f.eks. semikolon. Etter den første formatspesifikasjonen %s, skal det uansett være komma, siden %s på dette stedet indikerer filnavnet til fila der vi ønsker å legge dataene.



Deretter legges dataene fortløpende inn i tekststrengen i henhold til angitt format. En slik tekststreng utgjør en rad i fila. Siden dette blir en tekst-fil, bør vi føye til .txt. Dermed vil den bli lettere å åpne med en text-editor f.eks. notepad++.

Den ferdige tekststrengen legges i strengvariabelen: *buffer*. Deretter sendes innholdet av *buffer* til serveren med følgende kommando:

```
client.print(buffer);
```

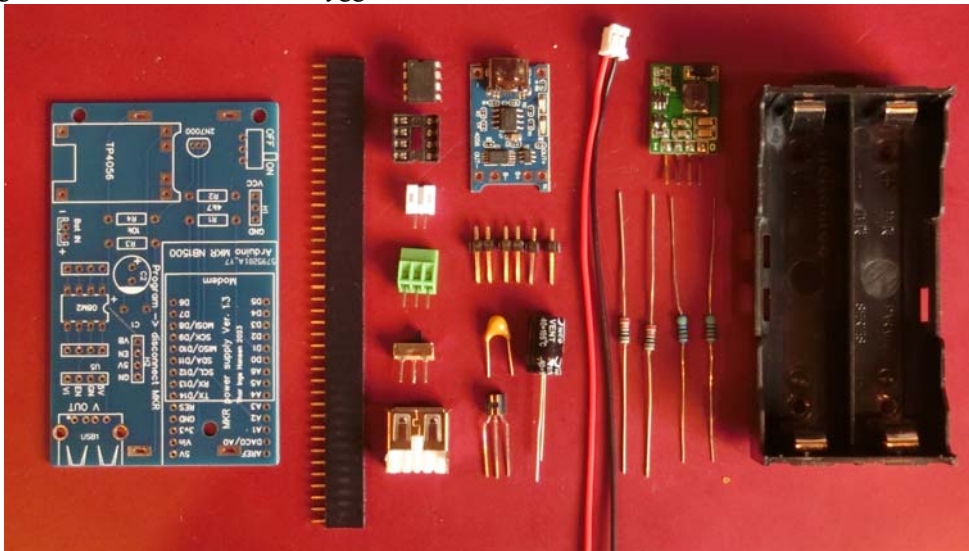
Det er selvfølgelig ingen ting i veien for å bygge opp en lengre streng med flere variabler, en må bare passe på at man ikke går ut over den maksimale bufferlengden. 255 kan være en slik grense, men bør sjekkes<sup>25</sup>.

Med tanke på at dataene skal kunne leses enkelt ved hjelp av Excel eller et Python-script er det lurt å droppe tekst mellom hver verdi. Strengen blir ikke så lett å lese, men egner seg godt for å ta inn i Excel eller Python.

### 3.8 Montering av PLSC MKR-kortet

I dette avsnittet skal vi beskrive montering og oppkobling av PLSC MKR-kortet. Kortet besørger lading av batteriet fra USB-C inngang, inneholder den eksterne “vakhunden” som holder rede på timingen for målingene, sørger for å slå av kretsene mellom hver måleperiode og hever spenningen fra batterispenningen på 3,7 V til 5,2 volt.

Figuren under viser innholdet i byggesettet:



Følgende verktøy kreves for monteringen:

- Loddebolt 30W med liten spiss

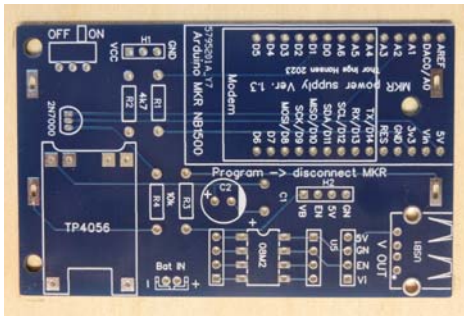
25.<https://www.sciencedirect.com/topics/computer-science/maximum-packet-size>



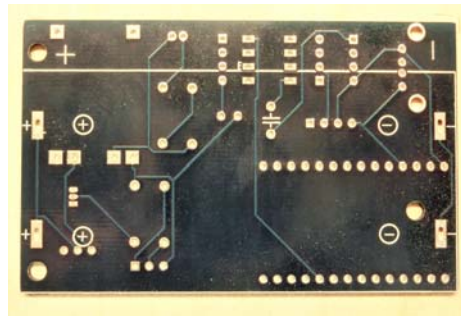
- Loddetinn
- Sideavbiter

## 1. Kretskortet

Kretskortet har en komponentside der de aller fleste komponentene skal plasseres (unntatt batteriholderen), og en loddesside der komponentene skal loddess til banene på kortet.



Komponentside



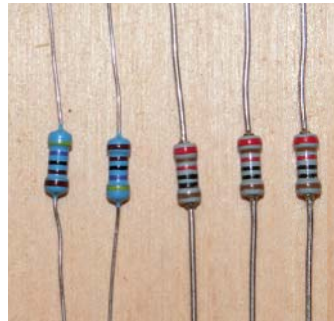
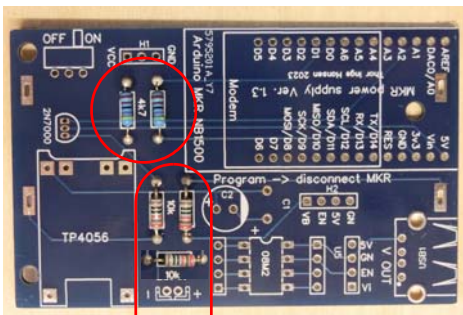
Loddesside

## 2. Motstandene

Motstandene er av to verdier:

R1 og R2 = 4,7k $\Omega$  (Gul, Fiolett, Rød, Gull (5%)) eller (Gul, Fiolett, Sort, Brun, Brun (1%))

R3, R4 og R5 = 10k $\Omega$  (Brun, Sort, Oransje, Gull (5%)) eller (Brun, Sort, Sort, Rød, Brun (1%)) Lodd på loddessida og klipp av ledninger.





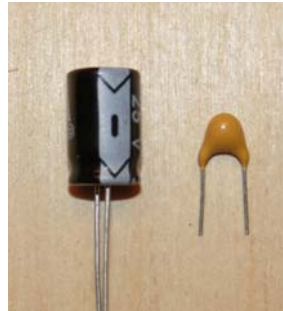
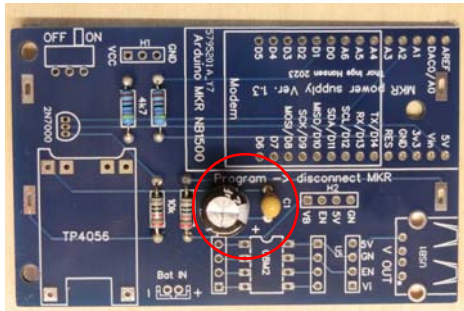
### 3. Kondensatorene

Kondensatorene er av to verdier:

C1 = 100nF (104)

C2 = 100µF (Husk at pinne merket – skal plassers i hull motsatt av +.

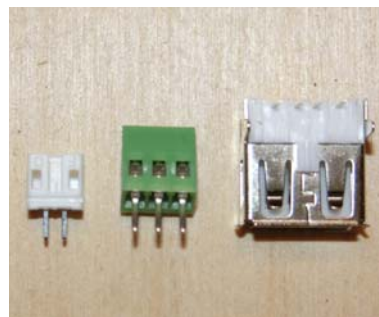
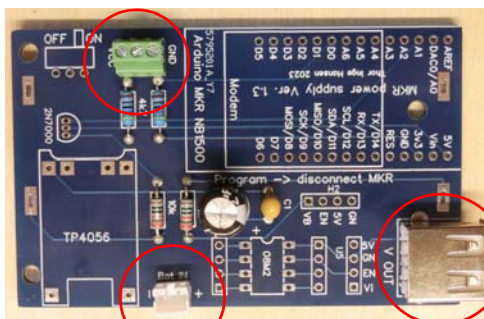
Lodd på loddessida og klipp av ledninger.



### 4. Batterikontakt, Koblingsplint og USB-A kontakt

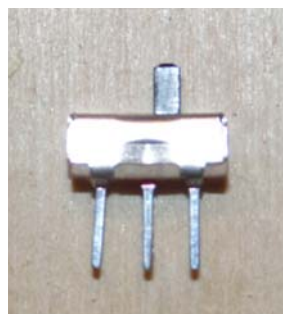
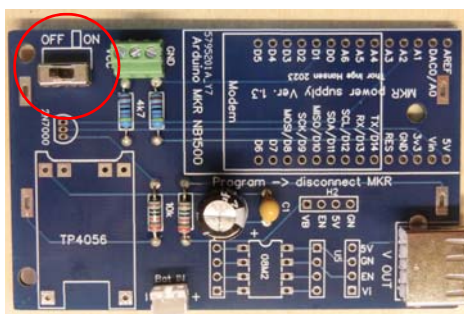
Tenk gjennom hvilken vei det er hensiktsmessig å sette koblingsplinten

Lodd på loddessida og klipp av ledninger.



### 5. Glidebryter

Plasser glidebryteren og lodd på loddessida

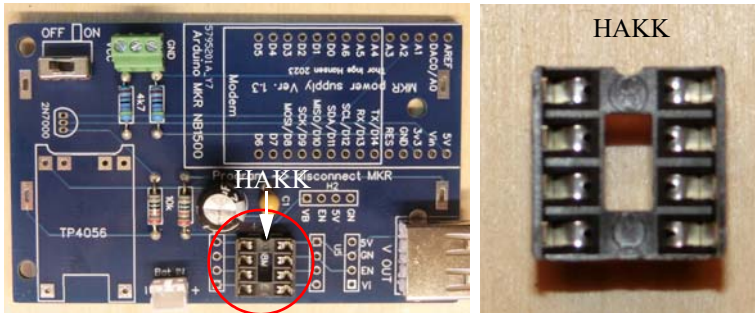






## 6. 8-pins sokkel

Husk å plasser hakket på rett side. Det skal vise hvilken vei 08M2+ mikrokontrolleren skal stå. 8pin IC-sokkelen har så tynne bein at de er lettere å brette flate på baksiden før lodding. Lodd på loddessida. Det er unødvendig å klippe ledningene på loddessida.



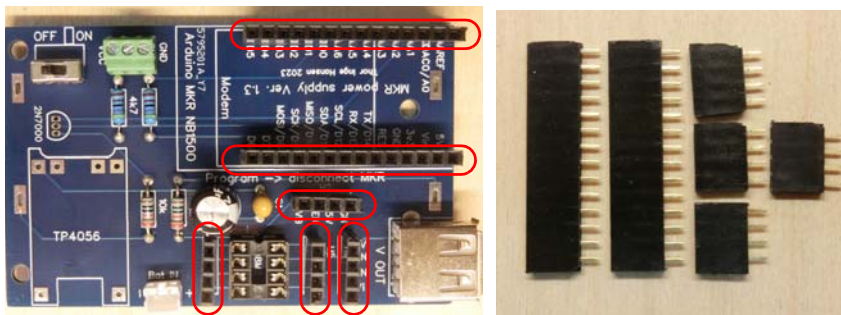
## 7. Hylsekontakter

Del opp hylsekontakten i seks remser med følgende lengder:

2 x 14 pins (til MKR NB 1500)

4 x 4 pins (til booster, til 3 målepunkter á 4 pins)

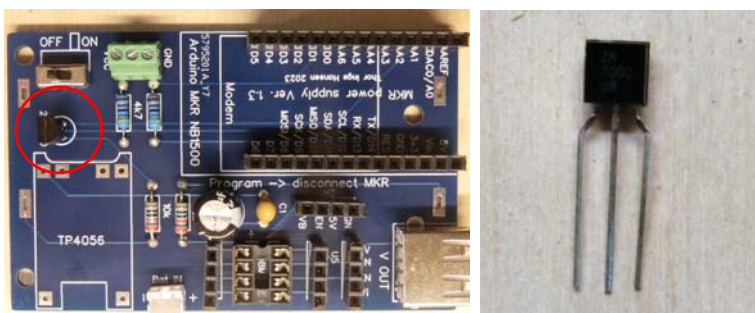
Lodd på loddessida og klipp av ledninger.



## 8. MOSFET transistor

Husk flat side rett vei. Transistoren må gjerne stå 5 – 10 mm over kortet.

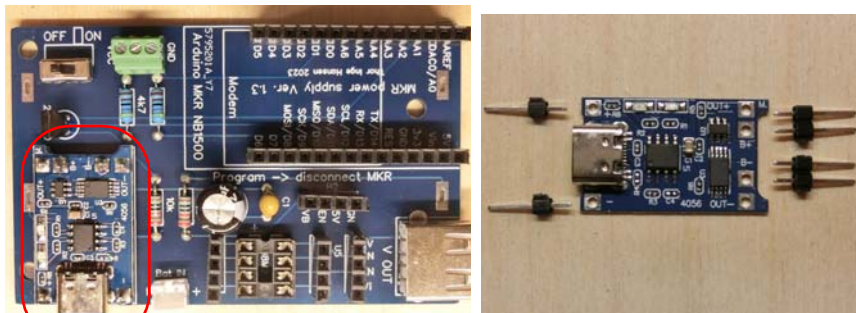
Lodd på loddessida og klipp av ledninger.





## 9. Ladekrets

Monter stiftlister på ladekrets og plasser på kortet.  
Lodd på loddessida og klipp av ledninger.



## 10. Batteriholder

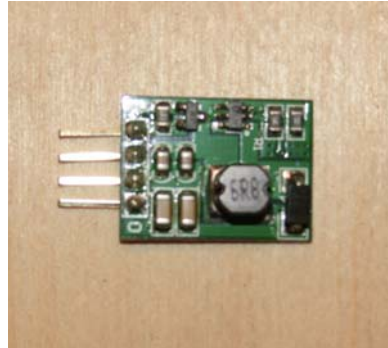
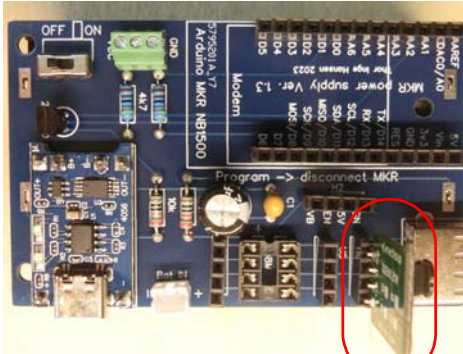
Snu kortet og plasser batteriholderen. Merk at batteriholderen har + og -, det samme har kretskortet. Pass på at + og - kommer på rett plass. Lodd loddeørene på komponentsida. **NB!** Det anbefales imidlertid å koble opp 18650-holderen via batterikontakten slik at brettet kan funksjonstestes med denne som tilførsel før batteriholderen loddes på baksiden. Dette gjør at det er lettere å fikse loddefeil før batteriholderen monteres





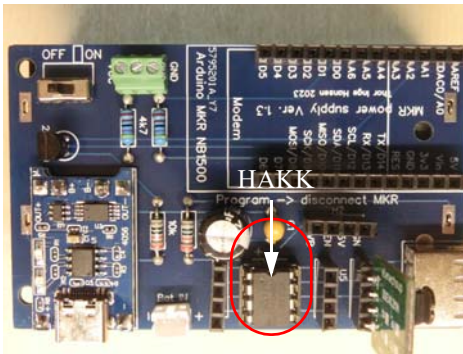
### 11. Booster

Plugg boosteren inn i hylselista som vist på bildet under. Pass på at den står riktig vei. Sørg for at Vi og Vo kommer på rett plass.



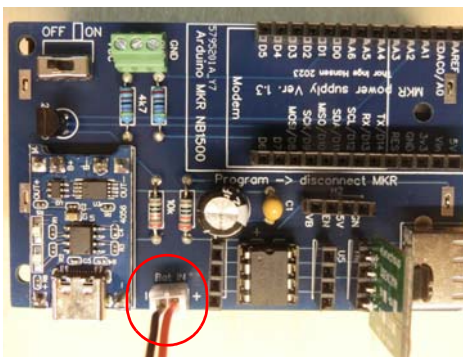
### 12. Sett i "Vakthund" 08M2+

Mikrokontrolleren som utgjør den eksterne "vakthunden" er en IC med 8 pinner (08M2+).



### 13. Sett i batteripluggen om det er ønskelig med ekstern spenningskilde.

Pass på at batteripluggen har rød ledning på + siden.







## 4 Teknologien

Kapitlet er ment som en ressurs som utdyper sentrale sider ved den valgte teknologien. Den omtaler også bruk av micro:bit (Wappsto:bit), Raspberry Pi (Pico) og Arduino (MKR NB 1500) som alternative teknologier som kan vurderes brukt. Vårt valg falt på Arduino som er utdypet i avsnitt 4.4 side 92.

### 4.1 Internet of Things

La oss først ganske kort se på hva som ligger i begrepet “Internet of Things” (IoT).

#### 4.1.1 Definisjon og nytteverdi<sup>26</sup>

*Internet of Things er et system av sammenkoblede mekaniske og digitale maskiner, objekter, dyr eller mennesker som er utstyrt med unike identifikatorer og har evnen til å overføre data over et nettverk. Interaksjonen mellom de ulike objektene er uavhengig av menneske-til-menneske eller menneske-til-datamaskin interaksjon.*

Umiddelbart kan dette virke ganske skremmende siden en kan få inntrykk av at utveksling av data skjer uten vår viten og vilje. Vi legger merke til at sammenkoblingen ikke bare gjelder gjenstander, men at også mennesker og dyr betraktes som “gjenstander” i denne sammenhengen. Et eksempel på det siste er:

*Innen helsevesenet kan IoT tilby mange fordelaktige tjenester som f.eks. overvåking og analysering av pasientdata. Dette kan være tilfelle på et sykehus eller en intensivavdeling for tidlig å bli klar over kritiske situasjoner. Eller for å holde oversikt over lagerbeholdninger når det gjelder medisiner og medisinsk utstyr.*

Et annet eksempel knyttet til jordbruk kan være:

*Overvåking av dyrka mark ved å måle og overføre data om lysforhold, temperatur, luftfuktighet og fuktigheten i jorda. IoT gir dermed mulighet til automatisk oppstart av vanningsanlegg, evt. regulering av lys og temperatur i drivhus.*

I slike sammenhenger ser en klart en gevinst mht. optimalisering av jordbruket.

I bynære områder er anvendelsesområdene mange:

*I forbindelse med “Smart city” kan man se for seg en rekke anvendelser som f.eks. smart gatebelysning og smart trafikkovervåking som gjør trafikken mer smidig, innføre tiltak som sparer energi og miljø, og forbedre sanitære forhold.*

---

26.<https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT>



På et mer personlig plan kan en tenke seg:

*Smarte boliger (“Smart house”) hvor sensorer styrer lys og temperatur i rom på bakgrunn av bruksmønster. Eller styrer luftkondisjoneringsanlegget på bakgrunn av hvor mange som befinner seg i et rom og således trenger utskifting av lufta og styring av oppvarming eller kjøling. På denne måten kan man redusere energibruken på en intelligent måte.*

*Noen av oss disponerer kombinerte kopi- og utskriftsmaskiner som automatisk varsler leverandøren om at det begynner å bli tomt for toner eller papir og sender ut nye forsyninger etter behov. Lignende maskiner er dessuten koblet sammen i nettverket slik at uansett hvilken maskin jeg oppsøker så gjenkjennes jeg slik at jeg kan få skrevet ut min tekst på den nærmeste maskinen.*



Med fornuftig bruk kan IoT bidra til en bærekraftig utvikling, samtidig som man må være seg bevisst at overdreven bruk like lett kan medføre det motsatte.

En litt mer spesiell anvendelse kan man oppnå ved å montere sensorer på kroppen og i klærne slik at de kan påvirke livsmønsteret vårt:

*Bærebare enheter med sensorer og programvare kan samle og analysere og overføre data, til installasjoner i omgivelsene som gjør livet lettere og mer komfortabelt. Slike bærbare enheter kan også redusere responstiden ved behov for akutt hjelp, ved f.eks. at den trengendes lokalisering overføres automatisk til redningsmannskapet, for å gi dem den optimale kjøreveien til den hjelpetrengende. Det samme gjelder for brannvesenet under utrykning.*

Vi legger merke til at flere av de nevnte tjenestene alt finnes i dag.

#### **4.1.2 Litt historie**

Som så mange andre ting så oppsto tanken om automatisk fjernstyring og overvåking tidlig. Allerede på 1970-tallet snakket man om “*embedded internet and pervasive computing*”, hvilket kanskje kan oversettes som *Integrert nettilkoblet databehandling*.

Den første internettapplikasjonen var f.eks. en Cola-automat ved Carnegie Mellon University tidlig på 80-tallet, hvor programmere ved universitetet kunne sjekke om det var kalde drikker tilgjengelig i automaten, slik at det var verdt turen for å hente seg drikke.



Det var imidlertid **Kevin Ashton (1968 –)**, en av grunnleggeren av Auto-ID senteret ved Massachusetts Institute of Technology (MIT), som først brukte begrepet Internet of Things. Han brukte begrepet i en presentasjon han gjorde for Procter & Gamble i 1999. Presentasjonen hadde til hensikt å rette søkelyset mot RFID (Radio Frequency IDentification), en teknologi vi i dag er godt kjent med gjennom f.eks. adgangs- og betalingskort. Den gang kalte Ashton forelesningen sin “Internet of Things”.



På omtrent samme tid skrev professor **Neil Gershfeld (1959 –)** ved MIT boka: *When Things start to Think*. Selv om han ikke brukte begrepet IoT så uttrykte han en klar visjon om at vi er på vei mot det vi i dag omtaler som IoT.

Siden den gang har trådløs teknologi, mikroelektromekaniske systemer og Internett nærmet seg hverandre og etter hvert blitt en fungerende enhet.

#### 4.1.3 Fordeler og ulemper med IoT

Generelt kan man si at IoT gir større muligheter til å overvåke prosessene i et foretak og på den måten øke produktiviteten hos de ansatte. Dessuten kan IoT gjøre kundeservicen bedre og mer effektiv og på den måten spare tid og penger. Fordelene avhenger imidlertid av i hvilken grad man er i stand til å dra nytte av den innhentete informasjon i foretakets driftmodeller slik at man kan ta bedre beslutninger og på den måten skape større overskudd. De mest åpenbare bedriftsmessige fordelene ser en kanskje innen produksjon, transport og lagerhold, men også innen effektivisering av jordbruk (jfr. tidligere eksempel), hushold og hjemmeautomatisering (f.eks. energiøkonomisering) og i forbindelse med “Smart cities” ser en fordeler mht. redusert forsøpling og energisparing. Andre fordeler er i større grad knyttet til bedriftspesifikke forhold og kan relateres til følgende forhold:

- Gir mulighet til å innhente informasjon om gjenstander, når som helst og fra hvor som helst slik at en f.eks. tidlig kan påvise feil og slitasje og dermed redusere større skader med påfølgende driftsstans.
- Forbedrer informasjonsflyt mellom ulike gjenstander som er tilkoblet nettverket.
- Gir betydelige innsparinger som et resultat av at informasjonen kan overføres automatisk via nettverket.
- Automatisk håndtering av informasjon uten menneskelig inngripen øker kvaliteten på prosesser både innen næringslivet og samfunnet forøvrig.

Men det finnes også ulemper:

- Ettersom antallet tilkoblede enheter øker og stadig mer informasjon deles over nettverket, blir også faren for at utenforstående får urettmessig adgang til gradert informasjon større.
- Utfordringen med å håndtere store mengder data på en fornuftig måte vil øke.



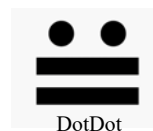
- Feil som oppstår i et slikt komplekst system vil lett ødelegge data fra samtlige noder siden alt henger sammen.

Siden det foreløpig ikke finnes noen enhetlig internasjonal standard på dette feltet så oppstår det lett problemer når ulike systemer skal utveksle data.

#### 4.1.4 Internasjonal standard for IoT

Det finnes en rekke kandidater til å bli en internasjonal standard på markedet, men foreløpig har ingen tatt skrittet fullt ut. Her er kanskje de vanligste og mest aktuelle:

- **iPv6** – “Low-Power Wireless Personal Area Networks” (6LoWPAN) er en åpen standard introdusert av *Internet Engineering Task Force* (IETF). Denne standarden tillater en hver laveffekts radio å kommunisere via Internett. iPv6 inkluderer og bygger på protokollen definert av 804.15.4. I denne familien finner vi også *Bluetooth Low Energy* (BLE) og *Z-Wave* som brukes innen hjemmeautomatisering.
- **ZigBee** er et laveffekts trådløst nettverk med lav datarate oftest brukt i industrielle nettverk og er basert på *Institute of Electrical and Electronics Engineers* (IEEE) 802.15.4 standarden. ZigBee-sammenslutningen har utviklet et felles programspråk *DotDot*<sup>27</sup> for IoT som gjør det mulig for “smarte objekter” å kommunisere trygt og forståelig med hverandre på alle nettverk.
- **LiteOS** er et Linux-basert operativsystem for trådløse sensornettverk. LiteOS støtter Smarte telefoner og klokker, wearables<sup>28</sup>, men brukes også i forbindelse med Smarte hjem og kjøretøyer koblet opp mot Internet.
- **OneM2M** er en kommunikasjonsstandard beregnet til å kommunisere mellom ulike maskiner f.eks. i en produksjonskjede.
- **DDS** – *Data distribution Service* ble utviklet av *the Object Management Group* (OMG) og er en standard for kommunikasjon mellom maskiner i en produksjonslinje. Standarden er skreddersydd for kommunikasjon med høy kvalitet i “real time”, og lar seg lett skalere opp fra små til store systemer.
- **LoRaWAN** – *Long Range Wide Area Network* er en protokoll for Wide Area Network – WAN, utviklet for å støtte f.eks. *Smart cities*, med millioner av enheter som sender på lave effekter.



## 4.2 Valg av teknologi

I dette avsnittet skal vi se på tre utvalgte teknologier knyttet til *Micro:bit* og programmering i blokkode og *Python, Raspberry Pi Pico* som bl.a. kan programmeres i microPython, og *Arduino* som programmeres i C++. Uansett hvilken teknologi vi velger så er radiosambandet viktig for vår bruk. Også her re det flere alternativer til ulik bruk.

---

<sup>27</sup>.<https://zigbeealliance.org/solution/dotdot/>

<sup>28</sup>.Wearables er elektronikk som festes nær kroppen, gjerne på huden, og er elektronikk som samler inn data, analyserer og sender videre informasjon om f.eks. vitale data om kroppens tilstand eller om omgivelsene.



### 4.2.1 Dataoverføringskanaler

Sentralt for de fleste IoT-løsninger er at datakommunikasjonen skjer trådløst fra sensornoden og til brukeren.

Fra norsk Wikipedia<sup>29</sup> leser vi:

***Bluetooth** (også kalt blåtann) er en radiooverføringsprotokoll som benyttes for å sende og motta data trådløst mellom enheter i et personlig datanett sammensatt av mobiltelefon, data-maskin, og/eller andre enheter som støtter protokollen. Bluetooth ble først utviklet av Ericsson, og senere formalisert av Bluetooth Special Interest Group (SIG), som ble presentert den 20. mai 1999. SIG var sammensatt av Sony Ericsson, IBM, Intel, Nokia og Toshiba. Bluetooth opererer i frekvensbåndet 2,402 til 2,480 GHz. Rekkevidden til bluetooth er i praksis på rundt 10-20 meter, avhengig av støymåling. Overføringskapasiteten er på maksimalt 2 Mbit/s.*

Bluetooth er altså en kortdistanse radiooverføring benyttet til personlige nettverk. Normalt brukes ikke protokollen i forbindelse med IoT, men kan evt. kommunisere lokalt ved sensornoden, evt. mellom nærliggende sensornoder.

Fra norsk Wikipedia<sup>30</sup> leser vi:

***Wi-Fi** er en kommunikasjonsteknologi for trådløst lokalt datanett. Gjennom Wi-Fi kan elektroniske enheter overføre data trådløst over et datanettverk ved hjelp av radiobølger, avgrenset til 10–100 meters rekkevidde. Datamaskiner, smarttelefoner og andre trådløse enheter kan koble seg til det trådløse nettet for å få tilgang til Internett. Den benytter radiofrekvenser i det lisensfrie 2,4 GHz-båndet (S-båndet).*

Wi-Fi blir mye brukt i forbindelse med IoT, men da gjerne innen en husstand, bedrift, skole eller et større område. Overføringshastigheten kan være flere 100 Mbit/sek. Wi-Fi 6E som ble etablert i 2020 kan levere datahastigheter opp til 9,6 Gbit/sek. Det normale er langt lavere.

Fra engelsk Wikipedia<sup>31</sup> leser vi:

***LoRa** (from "long range") is a physical proprietary radio communication technique. It is based on spread spectrum modulation techniques derived from chirp spread spectrum (CSS) technology. It was developed by Cycleo, a company of Grenoble, France, later acquired by Semtech.*

***LoRaWAN** (Wide Area Network) defines the communication protocol and system architecture. LoRaWAN is an official ITU-T Y.4480 standard of the International Telecommunication Union (ITU). The continued development of the LoRaWAN protocol is managed by the open, non-profit LoRa Alliance, of which SemTech is a founding member.*

*Together, **LoRa** and **LoRaWAN** define a Low Power, Wide Area (LPWA) networking protocol designed to wirelessly connect battery operated devices to The Internet in regional, national or global networks, and targets key Internet of Things (IoT) requirements such as bi-direc-*

---

29.<https://no.wikipedia.org/wiki/Bluetooth>

30.<https://no.wikipedia.org/wiki/Wi-Fi>

31.<https://en.wikipedia.org/wiki/LoRa>



*tional communication, end-to-end security, mobility and localization services. The low power, low bit rate, and IoT use distinguish this type of network from a wireless WAN that is designed to connect users or businesses, and carry more data, using more power. The LoRaWAN data rate ranges from 0.3 kbit/s to 50 kbit/s per channel.*

Som vi ser så brukes dette også i forbindelse med IoT for lavhastighets datakommunikasjon for et mindre område. LoRa kan ha typisk rekkevidde fra 1 – 10km, avhengig av hvilken effekt man bruker. I Norge er typisk øvre grense effekt på 100 mW i andre lang kan den være opp til 1W.

Fra Telenor<sup>32</sup> leser vi:

***NB-IoT**, Narrowband IoT er en IoT-teknologi basert på 4G-nettet som fungerer praktisk talt hvor som helst i landet. Den egner seg godt der hvor "tilkoblede ting" er lokalisert i avsidesliggende eller vanskelig tilgjengelige områder. Med lange avstander fra den neste mobile basestasjonen, eller i skjermede områder dypt inne i bygninger eller underjordiske områder. NB-IoT passer for løsninger som har behov for å overføre små datamengder, er skreddersydd for sensorer som bare skal sende data av og til, behov for lang levetid for batteri og har behov for høy dekningsgrad i gravgrendte strøk. Et godt valg ved bruk på logistikk av fraktgods, containere, transportmidler, jordbruk, smarte byer, smarte hjem og sporing av mennesker og dyr. Telenor har en befolkningsdekning på 99,9% for NB-IoT.*

***LTE-M** er også basert på 4G-nettet, og er en IoT-teknologi som ligner på NB-IoT. LTE-M gir imidlertid noe høyere hastigheter og er beregnet på overføring av større datamengder enn NB-IoT. LTE-M har forbedret batterilevetid og dekning, samt etterhvert mulighet for tale over nettverket. Et godt valg ved bruk av mer aktive sensorer som f.eks. sporing av eiendeler, smartklokker, autonome kjøretøyer, smartmålinger, medisinsk utstyr og hjemmesikkerhetsapplikasjoner.*

NB IoT er meget aktuell der man har behov for god dekning over hele landet, også innendørs. Men kan greie seg med å overføre små datamengder fra enheter med batteridrift.

### 4.3 Valg av microcontroller-teknologi

Det finnes et rikt tilfang av utviklingskort for NB IoT, her skal vi ta for oss noen få

#### 4.3.1 Micro:bit – Wappsto:bit fra det danske firmaet Seluxit<sup>33</sup>

Wappsto:bit er et ekspansjonskort for micro:bit spesielt utviklet for undervisningsformål. Både micro:bit og Wappsto:bit tilføres 5,0V via USB-pluggen på Wappsto:bit-kortet og ikke USB-pluggen på micro:bit-kortet. Programmene må imidlertid legges på micro:bit-kortet gjennom USB-pluggen på micro:biten, som så benytter ressursene på Wappsto:bit-kortet. Alle Wappsto:bit-kortene, det er fire varianter, har dessuten en stiftlist som gjør det mulig å komme til alle portene som micro:biten tilbyr. Alle sensorer som kan brukes sammen med micro:biten og gå på 3,3V spenningsforsyning, kan brukes, men det er bare et fåtall som det finnes biblioteker til for micro:bit.

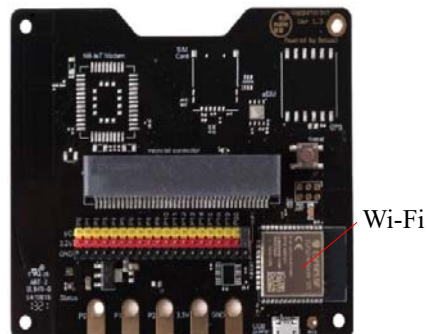
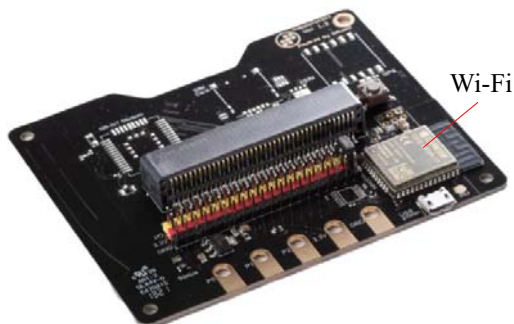
32. <https://www.telenor.no/bedrift/iot/abonnement/oversikt/>

33. Disse kretsene er i salg hos bl.a. RS-online.

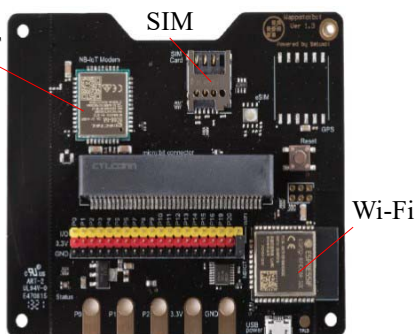
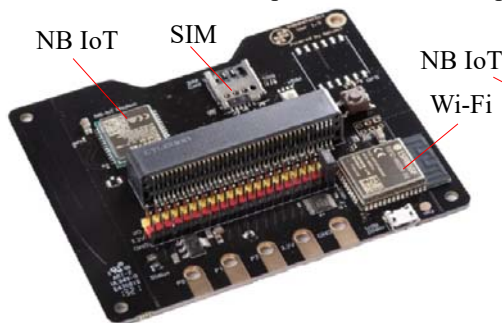


Produktet leveres i fire utgaver:

- *Weppsto:bit Basic*<sup>34</sup>  
har en ESP32 som kan bidra med betydelig regnekraft, lagerkapasitet og Wi-Fi om det er mulig å ta i bruk dennes egenskaper. Via Wi-Fi kan Weppsto:bit kobles opp til skyen via det lokale Wi-Fi-nettet.



- *Weppsto:bit IoT*<sup>35</sup>  
I tillegg til en ESP32, er kortet utstyrt med et NB IoT-modem for kommunikasjon i 4G-båndet. For å kunne kommunisere på NB IoT så må Wappsto:bit utstyres med et SIM-kort.



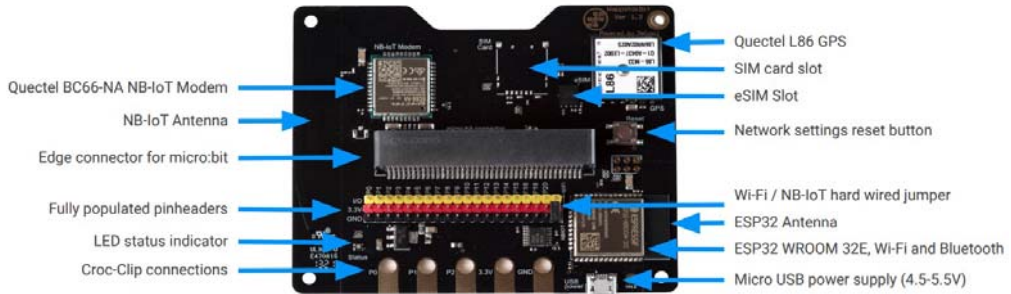
34. <https://no.rs-online.com/web/p/bbc-micro-bit-add-ons/2251593> Pris: NOK 364,85 + MVA

35. <https://no.rs-online.com/web/p/bbc-micro-bit-add-ons/2251594> - Pris: NOK 612,49 + MVA





- *Weppsto:bit IoT+*<sup>36</sup>  
I tillegg til Wi-Fi og NB IoT, så har denne varianten også en GPS.



Alle kortene er dessuten utstyrt med en kantkontakt der man kan koble til en standard micro:bit, versjon v1 eller v2. En stiftlist gir tilgang til samtlige porter supplert med +3,3V og jord (GND). I tillegg kan man koble til med banastikker eller krokodilleklemmer til P0, P1 og P2 i tillegg til GND og 3.3V.

- *Weppsto:bit Go*<sup>37</sup>  
Kortet kombinerer micro:bit og ESP32 slik at det er mulig å opprette kontakt med Wi-Fi og på den måten med sky-tjenesten og smarttelefon eller nettbrett. Micro:biten er integrert i kortet

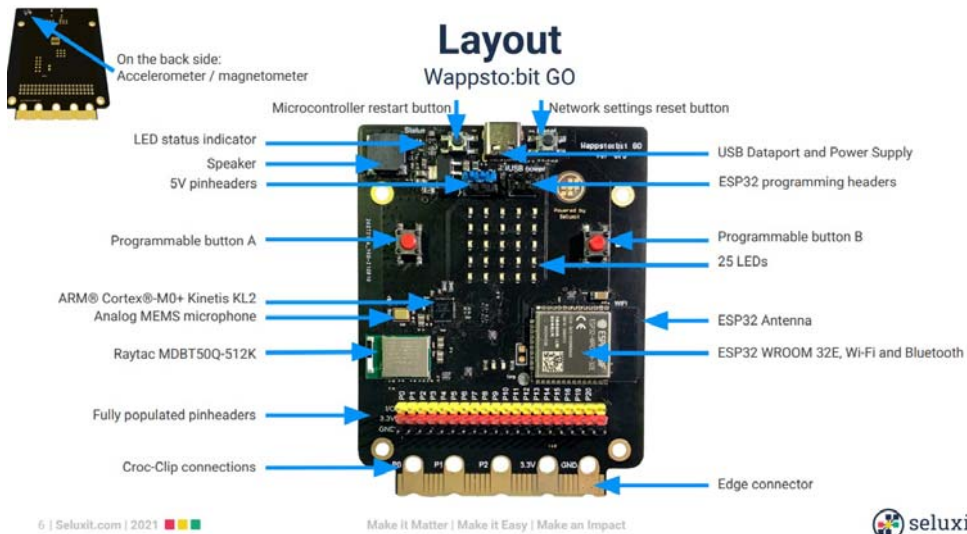
---

36. <https://no.rs-online.com/web/p/bbc-micro-bit-add-ons/2251595> - Pris: NOK 729,70 + MVA

37. <https://bit.wappsto.com/wp-content/uploads/sites/3/2021/09/Wappsto-bit-GO-Datasheet-September-2021-2.pdf>



som også har kantkontakt ala micro:bit og stiftlist slik at alle portene kan nå ved jumpere. Kortet passer og derfor godt sammen med f.eks. Kitronik Invention kit med koblingsbrett for eksperimentering.



Siden kortet inneholder alle egenskapene til micro:bit v2, så har kortet innebygget en rekke sensorer som temperatur, lyd- og lyssensor, magnetometer (compass), akselerometer i tillegg til to programmerbare knapper, A og B.

### Programmeringsspråk

Man kan velge å programmere i blokk-kode om man ønsker det, evt. kan man programmere i Java eller Python. Denne teknologiske løsningen egner seg derfor godt for nybegynnere som kan lære seg struktur ved bruk av blokkode for så å øke mulighetene ved å gå over til Python og tekstbasert programmering når de er modne for det.

### Skytjeneste

Alle disse kortene kan kobles til Seluxits sin skytjeneste som tilbys både for PC, smarttelefon eller nettbrett, både for IOS og Android. Dashboard'et kan enten være et standard board eller bygges opp etter egne behov. Nett-tilkoblingen kan enten gjøres via Wi-Fi eller NB IoT

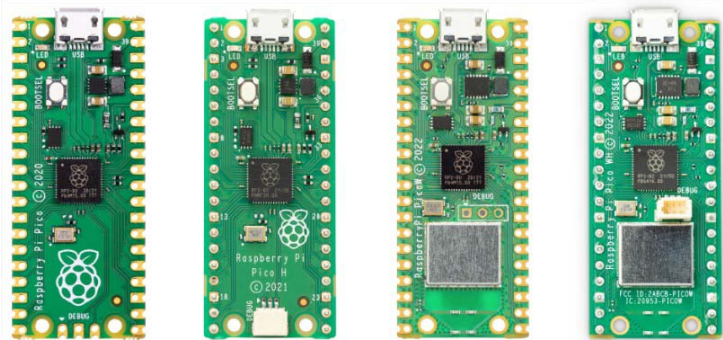
For mer informasjon om oppkobling se: *Miljøovervåking med havbøye, Arduino MKR NB 1500 – Data til server ved Inst. for marin teknikk* [9]. Eller gå rett til nettsiden til firmaet Seluxit: <https://bit.wappsto.com/>



### 4.3.2 Raspberry Pi Pico + SIM7080G<sup>38</sup>

#### Raspberry Pi Pico

Raspberry Pi er mest kjent for sine små kraftige datamaskinlignende kort som kan betjene både tastatur, mus og skjerm om slike kobles til. På den måten blir Raspberry Pi en selvstendig datamaskin. I denne sammenheng vil vi beskrive deres mikrokontroller-løsning som går under navnet Raspberry Pi Pico med shield-kort, eller *hatt*'er som det kalles i denne sammenheng.



Raspberry Pi Pico

Pico H

Pico W

Pico WH

Raspberry Pi Pico er et lavkost (< kr. 50,00 + MVA<sup>39</sup>) og relativt kraftig mikrokontroller-kort bygget opp omkring mikrokontrollerchipsen RP2040 som har en dobbelkjerne Arm Cortex M0+ prosessor med følgende sentrale parametere:

- 264kB SRAM, og 2MB flash memory
- USB 1.1
- Low-power sleep og dvale modus
- 26 GPIO porter
- 2 × SPI, 2 × I<sup>2</sup>C, 2 × UART, 3 × 12-bit ADC, 16 × PWM kanaler
- Nøyaktig klokke og timer på brikken
- Temperatursensor
- Akselerert flytende komma bibliotek på brikken
- 8 × Programmerbare I/O (PIO) tilstandsmaskiner for å støtte ekstern kommunikasjon

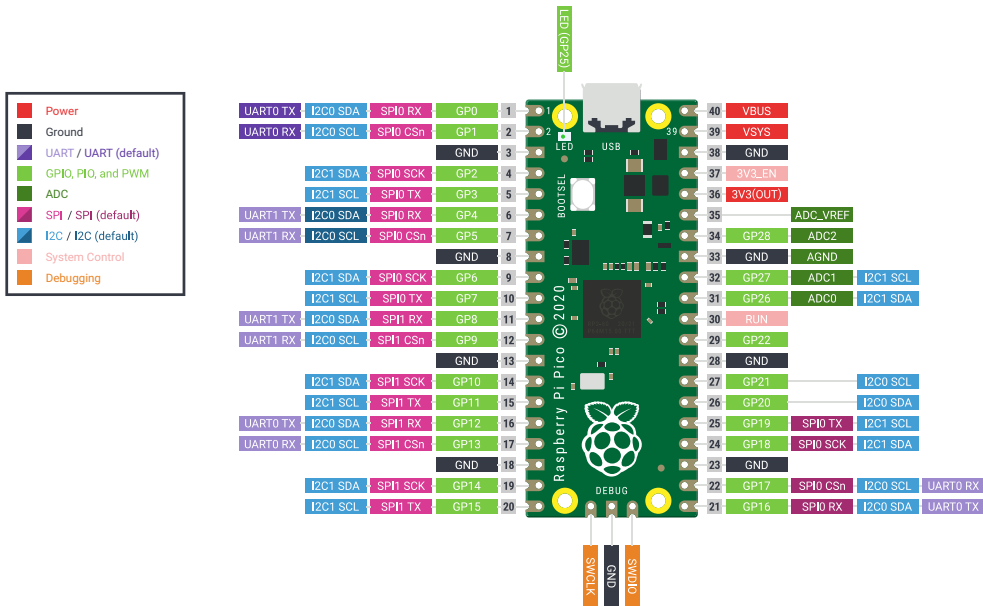
Pico leveres normalt uten stiftlist eller hylsekontakter, men har hull for slik montasje. Normalt monteres denne med "borgmur"-lodding direkte på kortet. Pico H leveres normalt med stiftlist.

38. <https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html>

39. <https://no.rs-online.com/web/p/raspberry-pi/2122162>



Figuren under viser pinningen til både Pico og Pico H.



Raspberry Pi Pico W og Pico WH er i tillegg utstyrt med et 2,4GHz Wi-Fi radiomodem (802.11n). Også disse variantene leveres for “borgmur”-loding (Pico W) for direkte montering på kortet eller med stiftlist (Pico WH).

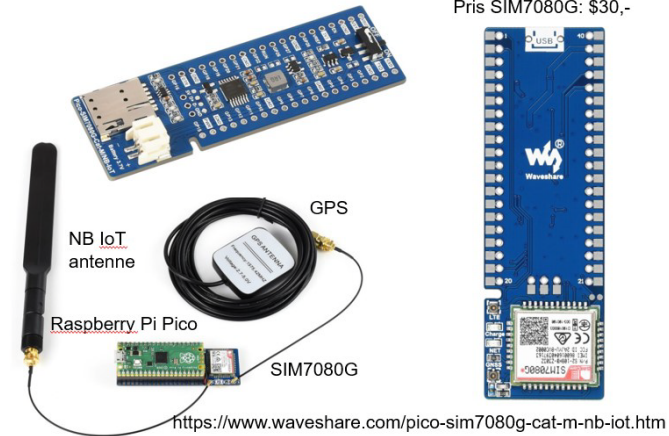
### SIM7080G NB-IoT / Cat-M(eMTC) / GNSS<sup>40</sup>

Dette er en påbygningsmodul til Raspberry Pi Pico som tilfører Pico NB IoT og GNSS. Pris: ca. \$ 30. Kortet kan også leveres uten GPS.

Pico-SIM7080G-Cat-M/NB-IoT er en NB-IoT (Narrow-Band-Internet of Things), Cat-M (aka eMTC, forbedret Machine Type Communication) og GNSS (Global Navigation Satellite System) modul designet for Raspberry Pi Pico. Den støtter flere NB-IoT-frekvensbåndene, kan styres via serielle AT-

#### Shield-kort med NB IoT, GPS – SIM7080G

Pris SIM7080G: \$30,-



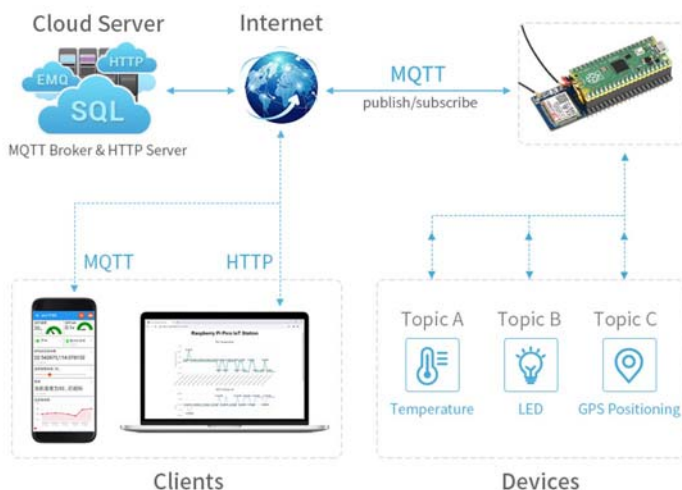
40. <https://www.waveshare.com/pico-sim7080g-cat-m-nb-iot.htm>



kommandoer, og støtter kommunikasjonsprotokoller som HTTP / MQTT / LWM2M / COAP, o.l. På grunn av fordelene med lav forsinkelse, lav effekt, lave kostnader, og bred dekning, egner denne kretsen seg for IoT-applikasjoner som intelligente instrumenter, sporing, fjernovervåking, o.l.

- Standard Raspberry Pi Pico tilkobling, støtter Raspberry Pi Pico serien av kort
- UART kommunikasjon, styres med AT-kommandoer
- Kommunikasjonsprotokollen støtter: TCP/UDP/HTTP/HTTPS/TLS/DTLS/PING/LWM2M/COAP/MQTT...
- GNSS lokalisering støtter: GPS, GLONASS, BeiDou, og Galileo
- 2x LED indikatorer, for overvåking av modulens driftsstatus
- Innebygd Nano SIM-kortholder, støtter *kun 1,8V SIM-kort* (3V SIM-kort er ikke tilgjengelig)
- Leveres med utviklingsressurser og instruksjonsbok med MicroPython eksempler
- Har en 3.7 – 4,2V Li-Po batteritilkobling og innebygget ladekrets
- Kretsen slår av Li-Po batteriet når USB kobles inn og starter lading
- Strømforbruk i Idle mode: 10mA
- Strømforbruk i sleep mode: 1,2mA
- Strømforbruk i PSM (Power Save Mode): 3,2µA
- Logisk nivå 3,3V

SIM7080G gjør det mulig å koble Pico opp mot en sky-tjeneste som vist på figuren til høyre.



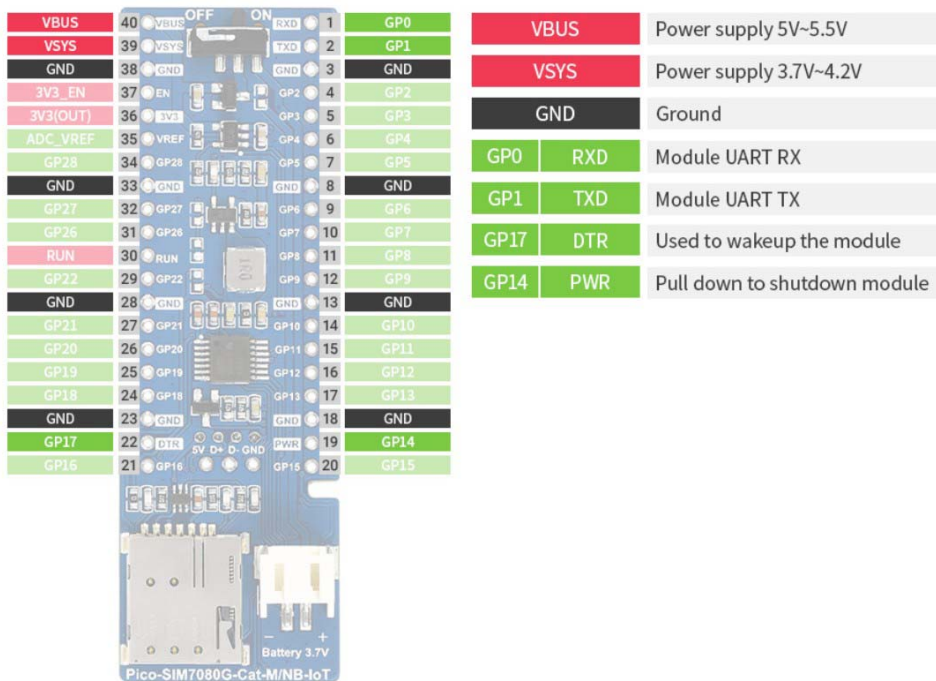
Ved å koble SIM7080G til Internett, vil Raspberry Pi Pico kunne sende data til en Cloud server ved hjelp av MQTT-protokoll. Dette gjør det mulig for brukere å overvåke Pico i sanntid via en smarttelefon APP eller en nettside.

Det finnes også en variant uten GPS som går under betegnelsen SIM7020<sup>41</sup>.

41. <https://docs.ineltek.com/docs/get-started-with-sim7020/>



Figuren under viser pinningen til SIM7080G.



## Programspråk

MicroPython er en slanket versjon av programmeringsspråket Python 3 som inkluderer et lite delsett av Python-standardbiblioteket og er en Python-versjon som er optimalisert for å kjøre på mikrokontrollere og i begrensede miljøer. MicroPython har som mål å være så kompatibel med vanlig Python som mulig, slik at man enkelt kan overføre kode fra skrivebordet til en mikrokontroller.

Thonny<sup>42</sup> er et gratis utviklingsprogram for PC som ble laget av en uavhengig utvikler som har samme navn (siste versjon i skrivende stund er v.4.0.2). Det er et integrert utviklingsmiljø med åpen kildekode (IDE) som kan brukes til å lage forskjellige applikasjoner ved hjelp av programmeringsspråket Python.

Akkurat som Microsoft Visual Studio eller NetBeans IDE, gjør Thonny det lettere for programmerere å kode, da det allerede kommer med de viktigste verktøyene, bibliotekene og det som ellers trengs for å komme i gang. Denne spesielle IDEen ble laget for å fokusere på Python og for å imøtekomme nybegynnere som ønsker å lære å kode og lage programmer i Python for mikrokontrollere.

42.<https://thonny.en.softonic.com/>





**Kommentar:** Fordelen med denne sett fra et skoleperspektiv er hovedsakelig at den kan programmeres i Python som begynner å bli utbredt i mange skoler. Prisen er dessuten lavere enn for f.eks. Arduino MKR NB 1500.

### 4.3.3 Arduino MKR NB 1500<sup>43</sup>

Arduino er en velkjent familie av mikrokontrollere også ved mange skoler, både ungdomsskole og videregående skole. Det er også en Teknologi som er mye brukt ved Skolelaboratoriet ved NTNU. Vi har derfor valgt å se nærmere på MKR NB 1500 som tilbyr NB IoT kommunikasjon.

Kortets hovedprosessor er en laveffekt Arm Cortex-M0® 32-bit SAMD21, som i de andre kortene i Arduino MKR-familien. Smalbåndstilkoblingen utføres med en modul fra u-blox, SARA-R410M-02B, et brikkesett med lav effekt som opererer i de forskjellige båndene i NB IoT og LTE-mobilområdet. I tillegg sikres sikker kommunikasjon gjennom kryptobrikken Microchip® ECC508 og tilkobling for ekstern antenne.



Kortet er designet for global bruk, og gir tilkobling på LTEs Cat M1 / NB1-bånd 1, 2, 3, 4, 5, 8, 12, 13, 18, 19, 20, 25, 26, 28.

USB-porten kan levere strøm (5V) til brettet. Den har også en inngang for tilkobling av et 3,7V Li-Po-batteri og en Li-Po-ladekrets som gjør at brettet, når det kjøres med ekstern 5V-kilde (USB), også lader Li-Po-batteriet. Bytte fra den en kilde til den andre gjøres automatisk ved tilkobling.

#### Noen sentrale parametere:

- Batteridrift med Li-Po 3.7V, 1500mAh Minimum
- Arbeidspenning er 3,3V
- 8 digitale I/O pinner og 7 analog input porter (ADC 8/10/12 bit)
- 13 porter som kan pulsbreddemoduleres (PWM) (0 .. 8, 10, 12, 18 / A3, 19 / A4)
- 1 x UART, 1 x SPI, 1 x I<sup>2</sup>C
- 1 x analog output porter (DAC 10 bit)
- 8 x eksterne interrupt (0, 1, 4, 5, 6, 7, 8, 16 / A1, 17 / A2)
- DC strøm pr. I/O port: 7mA
- Flash Memory: 256kB (internt)

---

43.





- SRAM: 32 kB
- EEPROM: nei
- Klokkehastighet: 32.768 kHz (RTC), 48 MHz
- LED\_BUILTIN port 6
- Effektklasse (radio) LTE Cat M1 / NB1: Klasse 3 (23 dBm)
- Data Rate (LTE M1 Half-Duplex): UL 375 kbps / DL 300 kbps
- Data Rate (LTE NB1 Full-Duplex): UL 62.5 kbps / DL 27.2 kbps
- Effektforbruk (LTE M1): min 100 mA / max 190 mA
- Effektforbruk (LTE NB1): min. 60 mA / max. 140 mA

For nærmere beskrivelse og pinning se avsnitt 4.4.1 på side 92.

### Shieldkort

Det finnes et stort utvalg av shield-kort som kan plugges direkte i MKR NB 1500. Her kan nevnes: MKR GSM, MKR ENV (Miljø-kort), stort og lite prototypkort med og uten SD-kortleser, memory-kort for å øke lagerkapasitet, med flere. For nærmere beskrivelse se avsnitt 4.5 på side 95

### Programmering

Kortet programmeres med Arduino C++ med en egen IDE (Editoringsverktøy). Se avsnitt 3.1 på side 49.

#### 4.3.4 Vårt valg av teknologi – En begrunnelse

Årsaken til at Arduino MKR NB 1500 ble valgt, har hovedsakelig vært at vi ved Skolelaboratoriet kjenner Arduino og programmeringsspråket C++. Dessuten finnes det et meget rikholdig utvalg av sensorer og biblioteker tilpasset Arduino ikke minst for mange sensorer. Utviklingskortet har dessuten vært lett å få tak i, hittil også med relativt kort leveringstid. MKR-serien har også et rikholdig utvalg av shield-kort som lett lar seg koble sammen uten lodding. En mulighet som er attraktiv i forbindelse med kurs. Når det er sagt, så er kortet dyrere en mange andre alternativer<sup>44</sup> og vi har hatt problemer med å starte radiomodemet når vi forsyner kortet med Li-Po-batterier på 3,7V. Vi har derfor valgt å bruke en spennings-booster som løfter spenningen til 5V. Dette er grundig behandlet senere i dette dokumentet.

En micro:bit variant som Wappsto:bit (ved firmaet Seluxit) er interessant fordi mange skoler har brukt micro:bit i undervisningen. Bruk av Makecode og blokkode har dessuten lav terskel og egner seg godt som introduksjon til programmering. Dessuten kan man gå over til tekstbasert programmering ved at programmeringsverktøyet tillater veksling mellom blokkode, MicroPython og Java. Dette er spesielt attraktivt siden Python er blitt et foretrukket språk i skolen. Ulempen er at utvalget av biblioteker foreløpig begrenser utvalget av sensorer og aktuatorer som kan anvendes

---

44.ELFA Distrelec: kr. 853,19 + MVA, RS-online: kr. 894,10 + MVA, Mouser: kr. 809,83 + MVA



sammen med kortet. Kortet er heller ikke forberedt for pluggin kort (shield) slik som Arduino og Raspberry Pi Pico er. Det er imidlertid en fordel at Seluxit tilbyr en skytjeneste for presentasjon av data. Prisen er også noe lavere enn Arduino MKR NB 1500.

Raspberry Pi Pico er også et godt alternativ, kanskje først og fremst fordi det er en billig løsning. Det er imidlertid en fordel av mikrokontrolleren kan programmeres i microPython. Utvalget av biblioteker sensorer er imidlertid rikelig.

## 4.4 Arduino

Vi har tidligere studert ESP32 som er utstyrt med Bluetooth og Wi-Fi for oppkobling mot lokale nettverk og andre Bluetooth-enheter. MKR NB 1500 bruker det mobile datanettet 4G, som har en langt større dekningsgrad enn Bluetooth og Wi-Fi. Ved hjelp av MKR NB 1500 kan man via dette nettet samle inn data fra nær sagt hvor som helst i Norge, om det skulle være i en myr i utmarka eller langs kysten. I dette kapittelet skal vi se nærmere på anbefalt hårdvare med tilhørende shield-kort som er egnet til vårt formål. Vi har konsentrert oss om MKR NB 1500 med noen aktuelle shield-kort, som også ble anbefalt av Telenor som en av flere teknologiske løsninger.

### 4.4.1 MKR NB 1500

MKR NB 1500 er en representant for en hel familie av mikrokontrollerkort, både fra Arduino og andre leverandører. Mikrokontrolleren har en lav-effekts Arm® Cortex®-M0 32-bit SAMD21 (48 MHz), som kanskje ikke sier oss så mye. Vi legger merke til at kjernen er en Arm-prosessor som er en annen enn de vi vanligvis finner på Arduino utviklingskortene som UNO og MEGA2560 o.l. som er levert av Atmel, eller nå Microchip. For å kunne gjennomføre en sikker overføring av data, krypteres de ved hjelp av krypteringskretsen, ECC508 som også leveres av Microchip. Kommunikasjonsenheten er en SARA-R410M-02B levert av firmaet u-blox<sup>45</sup>. Kortet har en RTC-klokke (Real Time Clock). Lagerkapasiteten er på 256KByte Flash, 32KByte SRAM.



**Kommunikasjonsbånd:** Kretsen er ment for global bruk og kan kommunisere via NB-IoT og LTE-M (CAT-M1).

---

45. U-blox er et sveitsisk firma som så dagens lys i 1997 og har sitt utspring fra ZTH i Zyrich. I starten spesialiserte de seg på å utvikle kretser for lokalisering ved hjelp av GPS og leverte i 2000 GPS-kretser til den første mobiltelefonen med GPS. I 2009 begynte de leveransen av kretser for datakommunikasjon via GSM. I 2014 ble også Bluetooth kommunikasjon inkludert i porteføljen. I 2016 leverte de moduler for lokalisering med GNSS med en nøyaktighet på under meteren. I 2019 lanserte de SARA-R% serien som er spesialisert for IoT og som også tilbyr kryptert overføring. (<https://www.u-blox.com/en/history-0>)



**Power supply**<sup>46</sup>: I tillegg til at den får tilført spenning via USB-kontakten, har den en egen kontakt hvor man kan koble til et Li-Po batteri (typ. 3.7V). Når man bruker USB-kabelen for strømtilførsel, vil batteriet lades. Selve kortet arbeider på 3,3 V hvilket betyr at spenningen på I/O-portene ikke må overskride denne spenningen.

Dersom man ønsker å bruke et Li-Po batteri kan man koble til et batteri på 3,7 V (min. 1500 mAh) med JST-kontakt (samme som micro:bit). Kortet trekker fra 100 til 190 mA når det opererer i LTE M1-modus og fra 60 til 140 mA i LTE NB1-modus.

Batteriet vil lades dersom kretsen har tilkoblet både Li-Po-batteri og USB-kontakt. Ladestrømmen er begrenset til typisk 350mA og ladingen avsluttes etter 4 timer, som betyr at et batteri på 1400 mAh er det som maksimalt kan fullades i hver ladesesjon. Det hindrer ikke at batterier med større kapasitet kan brukes, men oppladingen vil måtte skje over lengre tid og på annen måte.

Det rapporteres om at bruk av 3,7V Li-Po-batteri kan gjøre kretsen ustabil. Dette kan skyldes at spenningen kan bli noe marginal etter som batteriet lades ut. I slike tilfeller kan det være lurt å vurdere bruk av DC-DC-konverterer for å løfte spenningen fra 3,7 V til 5V, som er en løsning også vi har valgt (se avsnitt 4.11 side 129).

$V_{in}$  er en inngang som kan brukes for å gi kortet spenning. Normalt bør denne spenningen være 5V, men kan maksimalt gå opp til 6 V. Det anbefales imidlertid ikke å gå særlig høyere enn 5V.

+5V utgangen er ment å brukes til eksterne kretser som trenger 5V og er ikke ment å brukes for å tilføre kretsen driftspenning. Unntaksvis kan vi likevel gjøre det dersom kilden er godt regulert og sikret mot uønskede spenning variasjoner.

Bruker man USB'en eller  $V_{in}$  som strømkilde vil en LED på kortet lyse (strømtrekk typisk 10 mA). Bruker man batterikontakten (JST) vil denne LED'en være slukket. Det kan derfor være aktuelt å lodde fra denne LED'en dersom man ønsker å gi kortet spenning via USB-pluggen eller fra annen batterikilde via  $V_{in}$ .

**Inn- og utganger:** Kortet har 8 unike digitale I/O-porter, hvor det interne LED-lyset er koblet til port 6. I tillegg har kortet 7 analoge innganger, som også kan benyttes som digitale I/O-porter etter behov. AD-konverteren knyttet til de analoge inngangene kan settes til 8, 10 eller 12 bit. I alt kan 13 av portene pulsbreddemoduleres, dette gjelder portene D0 – D8, D10, D12, A3 og A4. Dessuten kan følgende 10 av portene kobles til interrupt: D0, D1, D4 – D9, A1 og A2. Hver utgang kan belastes med inntil 7mA.

**AREF** er en spenningsinngang hvor det er mulig å sette en ekstern referansespenning til AD-konverteren. Normalt vil den være 3,3V for MKR-serien, men den kan settes til spenninger fra 0V og opp til maks 5V. Dersom man ønsker å bruke ekstern referansespenning må dette angis i programmet med følgende kommando `analogReference(EXTERNAL)`;<sup>47</sup>

**RESET (P24):** Denne er normalt høy så lenge ingen ting er tilkoblet. Dersom den dras lav vil mikrokontrolleren resettes og starte programmet på nytt.

---

46.<https://core-electronics.com.au/arduino-mkr-nb-1500-44916.html>

47.<https://www.arduino.cc/reference/en/language/functions/analog-io/analogreference/>



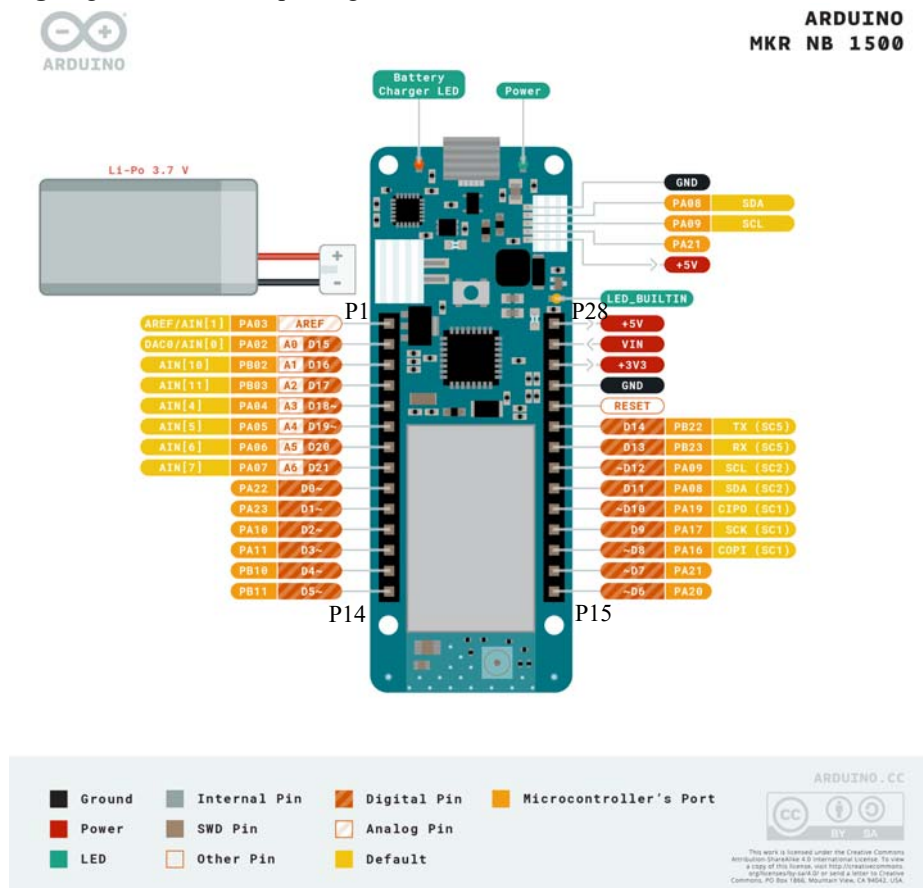
**Frekvensbånd og datarate**<sup>48</sup>: Kortet kan operere i følgende LTE-bånd: 1, 2, 3, 4, 5, 8, 12, 13, 18, 19, 20, 25, 26, 28 hvilket innebærer en lang rekke frekvensbånd spredt ut over i GHz området. Hvert bånd har en båndbredde på fra 10 – 80 MHz og inneholder mange kanaler. Det opereres med separate frekvenser for up- og down-link (til og fra basestasjonen).

Kortet kan operere i halv eller full dupleks. *Halv dupleks* betyr at det først sendes for så å motta, eller omvendt, mens *Full dupleks* betyr at det sendes og mottas samtidig.

Typisk datarate for Full dupleks er for up-link 62.5 kbps og down-link 27.2 kbps, mens for halv dupleks up-link 375 kbps og down-link 300 kbps.

**Kommunikasjon**: Kortet har også UART (Tx/Rx), en I<sup>2</sup>C, og en SPI-buss.

**Pinning**: Figuren under viser pinningen til MKR NB 1500.



Det er dessuten en holder for mikro SIM-kort på undersiden av kortet.

<sup>48</sup>[https://www.sqimway.com/lte\\_band.php](https://www.sqimway.com/lte_band.php)



**Shield-kort:** Det finnes en rekke Shield-kort som kan plugges ned i hovedkortet (se eget avsnitt 4.5 side 95).

## 4.5 Arduino shield-kort

Dette er kort som kan monteres på “ryggen” av MKR NB 1500 eller kobles til via kabel. Figuren under viser hva som finnes av slike kort i skrivende stund.



La oss se på et aktuelt utvalg av disse.

Flere av disse kortene benytter den felles I<sup>2</sup>C-buss for å kommunisere med mikrokontrolleren. Her er en oversikt over brukte I<sup>2</sup>C-adresser for GPS- og ENV-kortene:

- 66 (0x42) – MKR GPS
- 92 (0x5C) – MKR ENV – Lyssensor LPS22HB
- 95 (0x5F) – MKR ENV – Luftfuktighet HTS221
- 96 (0x60) – MKR NB 1500 – MKR WiFi-enheten (internt)
- 105 (0x6B) – MKR NB 1500 – MKR WiFi-enheten (internt)

### Oversikt over brukte pinner på MKR NB 1500

Tabellen under viser hvordan pinnene på MKR NB 1500 er organisert når shield-kortene MKR ENV og MKR GPS plugges inn på ryggen av MKR NB 1500. Ikke alle er i bruk i vår anvendelse.

Tabell 1: Shield-kortenes bruk av pinner

Port #	Port funksjon	Shield	Port #	Port funksjon	Shield
1	AREF		15	D6~	ENV DRDY Temp, Hum
2	A0	Analog sensor	16	D7~	ENV DRDY, Air pressure
3	A1	Analog sensor	17	D8~	MOSI - SD-kort
4	A2	ENV LYS	18	D9~	SCK - SD-kort
5	A3	Batterinivå	19	D10~	MISO - SD-kort

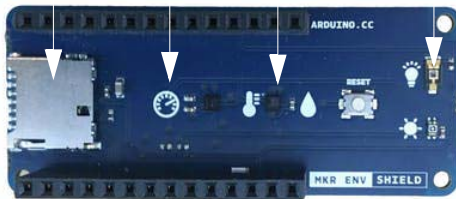


Port #	Port funksjon	Shield	Port #	Port funksjon	Shield
6	A4		20	D11~	ENV SDA I <sup>2</sup> C GPS SDA I <sup>2</sup> C
7	A5		21	D12~	ENV SCL I <sup>2</sup> C GPS SDA I <sup>2</sup> C
8	A6		22	D13~	Rx (GPS)
9	D0~	pinWDRreset	23	D14~	Tx (GPS)
10	D1~	DS18B20 VannTemp	24	RESET	
11	D2~	Digital sensor	25	GND	
12	D3~	Digital sensor	26	+3,3V	
13	D4~	ENV SD CS	27	Vin	
14	D5~		28	+5V	

### 4.5.1 MKR ENV Shield

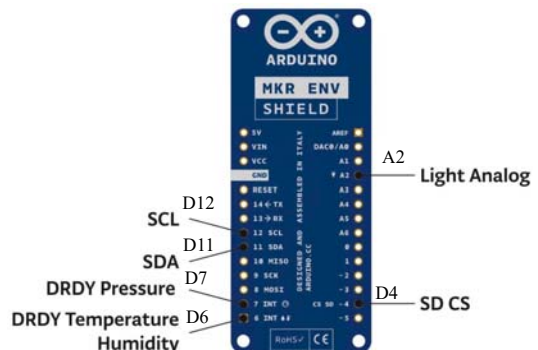
Kortet inneholder temperatur-, luftfuktighet-, lufttrykk, lyssensor og sensor for måling av UV-stråling (kun eldre kort). I tillegg har kortet en SD-kortleser for lagring av data.

SD-kortleser Trykk Temp/Fukt Lys



### Tilkoblinger<sup>49</sup>

Kortet bruker både I<sup>2</sup>C, SPI og en av de analoge inngangene. I<sup>2</sup>C brukes for å kommunisere med lufttrykk, temperatur, luftfuktighet og UV-stråling (som fantes på de opprinnelige kortene). I<sup>2</sup>C kobles til port D12 (SCL) og D11 (SDA). Lysmåleren er koblet til den analoge inngangen A2 og måler lysintensiteten i Lux. SD-kortleseren er koblet til SPI grensesnittet hvor D8 (MOSI), D9 (SCK) og D10 (MISO) står for



49. <https://community.element14.com/products/arduino/b/blog/posts/arduino-mkr-wifi-1010-mkr-env-shield>



kommunikasjonen. I tillegg benyttes D4 SD CS (SD-kort Chip Select) for å aktivisere SD-terminalen. Videre er port D6 koblet til HTS221 (temperatur og luftfuktighet) DataReaDY, og D7 er koblet til LPS22HB (lufttrykk) DataReaDY som vist på figuren over til høyre.

### Lufttrykk (LPS22HP – I<sup>2</sup>C adresse 0x5C)<sup>50</sup>

Sensoren har et måleområde fra 260 – 1260 hPa (mBar) med en oppløsning 24 bit. I tillegg måler den temperaturen med 16 bits oppløsning, temperaturmålingen brukes bl.a. til å kompensere for temperaturavhengighet hos lufttrykksmåleren. Dataraten (ODR) kan settes fra 1 til 75 sps (samples pr. sec.).

Kretsen kan kommunisere med omverdenen via I<sup>2</sup>C og SPI og kan arbeide med spenninger fra 1,7 – 3,6 V. Strømforbruket er så lavt som 3  $\mu$ A.

### Luftfuktighet og temperatur (HTS221 – I<sup>2</sup>C adresse 0x5F)<sup>51</sup>

Sensoren har et måleområde fra 0 – 100 % relativ fuktighet (RH) med en oppløsning på 16 bit, og en nøyaktighet på  $\pm 3,5\%$  i området fra 20 – 80 % RH. I tillegg måler den temperaturen med 16 bits oppløsning med et måleområde fra  $-40^{\circ}\text{C}$  til  $120^{\circ}\text{C}$ , og en nøyaktighet på  $\pm 0,5^{\circ}\text{C}$ , fra  $15^{\circ}\text{C}$  til  $+40^{\circ}\text{C}$ , temperaturmålingen brukes også til å beregne relativ fuktighet. Dataraten (ODR) kan settes fra 1 til 12,5 målinger pr. sekund.

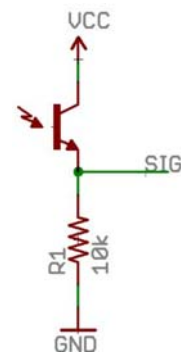
Kretsen kan kommunisere med omverdenen via I<sup>2</sup>C og SPI.

Også denne kretsen kan arbeide med spenninger fra 1,7 – 3,6 V og strømforbruket er så lavt som 1  $\mu$ A ved målerate på 1 Hz.

### Lyssensor (TEMPT6000 – A2)<sup>52</sup>

Sensoren er en fototransistor med følsomhetsvinkel på  $\pm 60^{\circ}$  og med en følsomhetstopp på 570 nm. Båndbredden strekker seg fra 440 – 800 nm. Sensoren måler lysintensiteten som treffer sensoren og som angis i Lux. Sensoren består av en fototransistor og en seriemotstand<sup>53</sup>. Strømmen i transistoren bestemmes av lysintensiteten. Sensoren leverer en spenning som er proporsjonal med lysintensiteten og kan leses av en av de analoge inngangene. Et typisk koblingsskjema er vist til høyre.

Strømforbruket varierer med lysstyrken der typisk strøm ved 1000 Lux er 0,5 mA. Ved sterkt lys kan strømmen bli vesentlig høyere.



50. <https://www.st.com/resource/en/datasheet/dm00140895.pdf>

51. <https://www.st.com/resource/en/datasheet/hts221.pdf>

52. <https://www.st.com/resource/en/datasheet/hts221.pdf>

53. <https://learn.sparkfun.com/tutorials/temt6000-ambient-light-sensor-hookup-guide/all>



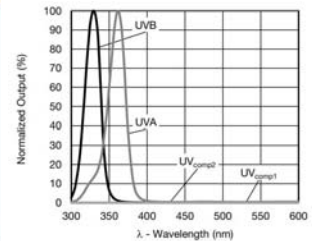


## UV-sensor (VEML6075) (Finnes bare på eldre utgaver av kortet)

Denne sensoren måler UV-stråling av typen A og B. Den siste kan også levere UV-indeksen som beregnes av MKR ENV biblioteket. Sensoren er utstyrt med I<sup>2</sup>C-buss for å overføre sensordataene. Tabellen under gir en indikasjon på faregraden for de ulike verdiene<sup>54</sup>, mens grafen til høyre viser normalisert spektralrespons for UVA (ca. 330 nm) og UVB (ca. 360 nm)<sup>55</sup>.

UV INDEX	UV INDEX	UV INDEX	UV INDEX	UV INDEX	UV INDEX	UV INDEX	UV INDEX	UV INDEX	UV INDEX	UV INDEX
1	2	3	4	5	6	7	8	9	10	11+
Low (1,2)		Moderate (3,4,5)			High (6,7)		Very high (8,9,10)		Extreme (11+)	
Green PMS 375		Yellow PMS 102			Orange PMS 151		Red PMS 032		Purple PMS 265	

Table 4: Presenting the UV: International colour codes!



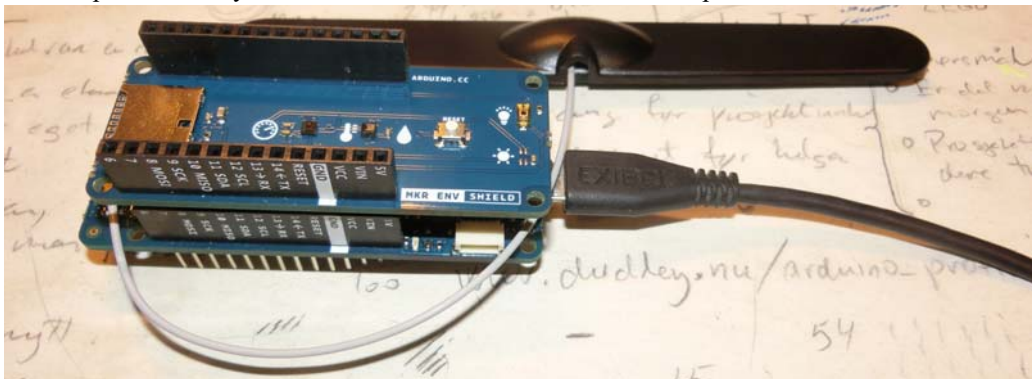
Kretsen kan operere med spenninger fra 1,7 – 3,6 V.

## Energiforbruk

Det har foreløpig ikke vært mulig å bestemme det totale strømforbruket til kortet og om det er mulig å legge sensorene i “stand by”. Sensorene for temperatur, luftfuktighet og lufttrykk trekker svært lite strøm, i størrelsesorden noen få  $\mu$ A. Lyssensorens strømforbruk varierer med lysstyrken og kan komme opp i flere mA. SD-kortleseren har en chip select som gjør at den kan legges i “stand by”.

## Montering av MKR ENV

MKR ENV kan monteres på toppen av mikrokontroller-kortet. PASS PÅ å sette kortet rette vei. Teksten på siden av hylsekontaktene viser hvilken vei kortet skal plasseres.



54. [https://www.who.int/news-room/questions-and-answers/item/radiation-the-ultraviolet-\(uv\)-index](https://www.who.int/news-room/questions-and-answers/item/radiation-the-ultraviolet-(uv)-index)

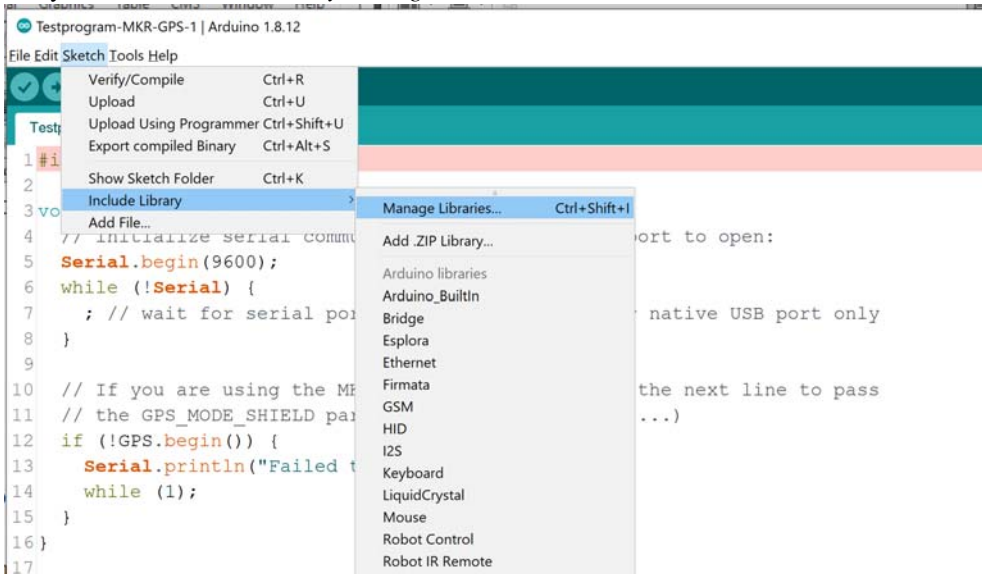
55. <http://www.farnell.com/datasheets/2245219.pdf>



## Programmering – Testprogram for MKR ENV (Testprogram-MKR-ENV-1.ino)

### 1. Installasjon av bibliotek MKR ENV

For å kunne bruke MKR ENV-kortet må vi installere et bibliotek. For å gjøre det går vi inn i menyen *Sketch/Inklude Library/Manage Libraries*



Vi får da opp følgende vindu hvor vi skriver inn *Arduino\_MKRENV* i søkefeltet.

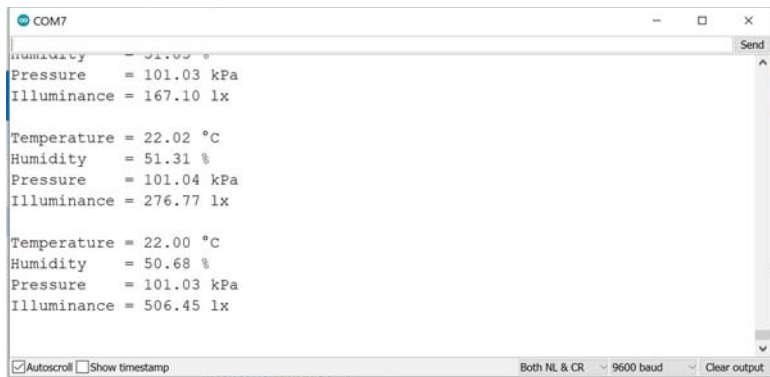


Vi velger å installere dette biblioteket som vist på figuren over.



## 2. Kjør testprogrammet

Last opp testprogrammet, se avsnitt I.1 på side 292, og kjør programmet. Pass på å velge riktig port og åpne monitorvinduet. Legg merke til at vi har kommentert bort koden som omhandler UVA



```
COM7
humidity = 51.03 %
Pressure = 101.03 kPa
Illuminance = 167.10 lx

Temperature = 22.02 °C
Humidity = 51.31 %
Pressure = 101.04 kPa
Illuminance = 276.77 lx

Temperature = 22.00 °C
Humidity = 50.68 %
Pressure = 101.03 kPa
Illuminance = 506.45 lx
```

og UVB siden vårt kort mangler denne sensoren. Om alt har gått som planlagt, skal du nå se måleverdier for de aktuelle parametrene som vist på figuren over.

Legg merke til at programmet viser temperatur, luftfuktighet, lufttrykk og lysintensitet.

## 3. Studer oppbyggingen av testprogrammet

Det er lurt å ta en titt på hvordan testprogrammet er bygget opp. Hovedprogrammet vårt inkluderer deler av dette testprogrammet.

### *Inkluder biblioteket*

Vi inkluderer biblioteket ved å sette følgende kommando øverst i programfila:

```
#include <Arduino_MKRENV.h>
```

### *Initialisering av ENV-kortet*

For å initialisere og sette opp ENV-kortet bruker vi følgende kommandoer

```
if (!ENV.begin()) {
  Serial.println("Failed to initialize MKR ENV shield!");
  while (1);
}
```

Initialiseringen gjøres av kommandoen `ENV.begin()`. Denne funksjonen returnerer "1" dersom alt er gått bra. Returnerer den "0" gir den feilmeldingen "Failed to initialize MKR ENV shield!" og stopper programmet (`while (1);`).

Initialiseringen gjøres i `setup()`-funksjonen.

### *Avlesning av måledata*

Følgende måleverdier kan leses av fra ENV-kortet:

```
float temperature = ENV.readTemperature();
float humidity    = ENV.readHumidity();
float pressure    = ENV.readPressure();
float illuminance = ENV.readIlluminance();
```



Vi legger merke til at variablene som holder de avleste verdiene deklarerer og tilordnes avleste verdier på samme linje.

### ***Skriving til SD-kort***<sup>56</sup>

Vi skal se hvordan vi kan skrive til SD-kortterminalen:

#### **1. Inkluder bibliotek**

Vi antar at vi henter måledata fra MKR ENV kortet som vi ønsker å skrive til SD-kortet. Vi må da inkludere biblioteket ved hjelp av header-filene:

```
#include <Arduino_MKRENV.h>
#include <SPI.h>
#include <SD.h>
```

Disse legger vi helt i starten av programmet

#### **2. “Chip select” SD-kort**

SD-kortleseren kan slås av og på via programmet. Dette kan man gjøre ved å legge linjen SD\_CS\_PIN “lav” (av) eller “høy” (på). Vi velger å lage oss en konstant som vi tilordner den porten vi har valgt å koble til denne inngangen på SD-kort holderen, fabrikanten har valgt å bruke port D4.

```
const int SD_CS_PIN = 4;
```

Denne legger vi før void setup()-funksjonen

#### **3. Deklarer variabler**

Deretter deklarerer vi tre variabler som holder de tre måleverdiene vi ønsker å lese av og logge på kortet:

```
float temperature = 0;
float humidity = 0;
float pressure = 0;
```

Disse deklarerer før void setup()-funksjonen.

#### **4. Deklarer et objekt for filbehandling**

Vi deklarerer et objekt dataFile av typen File

```
File dataFile;
```

Denne deklarerer før void setup()-funksjonen.

#### **5. Initialisering**

I void setup()-funksjonen initialiseres SPI-bussen og skriving til SD-kortet:

```
SPI.begin();
delay(100);
```

og deretter SD-kortterminalen

---

<sup>56</sup><https://docs.arduino.cc/retired/getting-started-guides/IoT%20Prime%20-%20Experiment%2003>



```
if(!SD.begin(SD_CS_PIN))
{
  Serial.println("Failed to initialize SD card!");
  while (1);
}
```

I tillegg har vi lagt inn et varsel dersom det oppstår problemer med å initialisere SD-kort-terminalen. Vi legger merke til at porten som sørger for “chip select”, inngår i initialiseringen av SD-kortterminalen. Oppstår det en feil vil programmet stoppe (`while (1)`).

## 6. Åpne filen og gi den et navn

Vi ønsker å opprette en fil med et gitt filnavn og skrive en overskrift. Siden overskriften kun skal skrives en gang, velger vi å legge denne kommandoen i void `setup()`-funksjonen.

```
dataFile = SD.open("log-0001.csv", FILE_WRITE);
delay(1000);
```

I denne kommandoen gir vi fila navnet `log-0001.csv`, hvor `csv` står for “comma-separated values” som er lett å hente opp i f.eks. Excel eller ved hjelp av et Python-skript. `FILE_WRITE` betyr at filen er både skrivbar og lesbar. Legg også merke til at vi legger inn en pause etter at den er gitt et navn. ***Her er det viktig å huske på at vi ikke kan bruke lengre filnavn enn 8 karakterer.***

## 7. Lukk filen

Deretter ønsker vi å lukke filen med følgende kommando:

```
dataFile.close();
delay(100);
```

Legg merke til at vi har lagt inn et lite `delay` etter at kommandoen er utført.

## 8. Åpne fila for datalogging

I void `loop()`-funksjonen ønsker vi å åpne fila for å skrive inn sensorverdiene. Da åpner vi fila på nytt med følgende kommando:

```
dataFile = SD.open("log-0001.csv", FILE_WRITE);
delay(1000);
```

Dernest leser vi av sensorverdiene:

```
temperature = ENV.readTemperature();
humidity = ENV.readHumidity();
pressure = ENV.readPressure();
```

For så å skrive dem inn i fila på denne måten:

```
dataFile.print(millis());
dataFile.print(",");
dataFile.print(temperature);
dataFile.print(",");
dataFile.print(humidity);
dataFile.print(",");
dataFile.println(pressure);
```



Legg merke til at vi legger inn komma mellom hver variabel uten noe mellomrom, dessuten har vi brukt `.println` i siste linje for å få programmet til å skifte linje mellom hver måling. Det er også lurt å legge inn en tidsangivelse i første kolonne, dermed er det mulig å lage plott av målingene som funksjon av tiden.

## 9. Lukk filen

Deretter lukker vi fila med følgende kommando:

```
dataFile.close();
```

Et testprogram ligger i vedlegg I.2 side 293.

For ytterligere studie av kommandoer for bruk i forbindelse med SD-kort, se følgende:

- `SD.begin(<chip select pin>)` – Initialiserer SD-leseren og setter hvilken port “chip select” skal kobles til: <https://www.arduino.cc/en/Reference/SDbegin>
- `SD.exists(<filenavn>)` – Sjekker om den aktuelle filen er på SD-kortet <https://www.arduino.cc/en/Reference/SDexists>
- `SD.mkdir(a/b/c)` – Lager mapper og undermapper <https://www.arduino.cc/en/Reference/SDmkdir>
- `SD.open(<filenavn>, <mode>)` – Åpner filen med navnet <filenavn>, for skriving og lesing `mode = FILE_WRITE` eller `mode = FILE_READ`, kun for lesing. <https://www.arduino.cc/en/Reference/SDopen>
- `SD.remove(<filenavn>)` – Fjern filen fra SD-kortet med navnet <filenavn> <https://www.arduino.cc/en/Reference/SDremove>
- `SD.rmdir(<filenavn>)` – Slett mappe med aktuelle navn. <https://www.arduino.cc/en/Reference/SDrmdir>

### 4.5.2 MKR GPS Shield

MKR GPS er et shield-kort som enten kan plugges ned i “ryggen” på MKR NB 1500 eller tilkobles via en kabel med Eslov-kontakt. Mottakeren baserer seg på modulen SAM M8 Q Ublox Global Navigation Satellite System (GNSS) og kan håndtere signaler fra GPS (Amerikanske), GLONASS (Russiske), og Galileo (Europeiske). Selve GPS-mottaker-enhetene kommuniserer via I<sup>2</sup>C. Brukes kabelen kommuniserer GPS-kortet med hovedkortet via I<sup>2</sup>C-buss, men plugges det inn i “ryggen” på MKR NB 1500, benyttes UART (Rx/Tx)<sup>57</sup>.

***Vi anbefaler å bruke kabelen og Eslov-kontakten da det er I<sup>2</sup>C-bussen som brukes i programvaren i denne anvendelsen.***

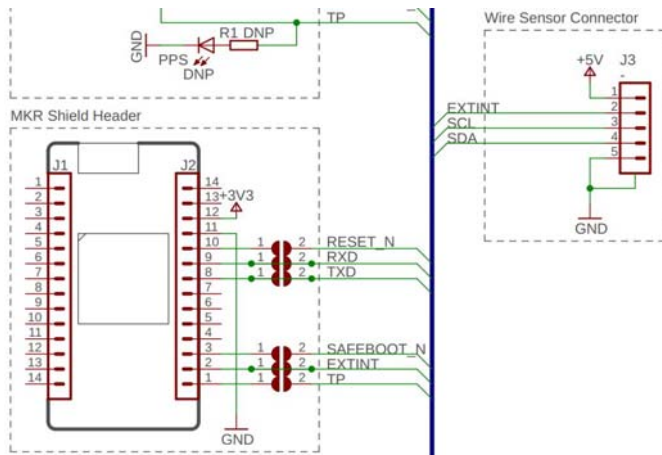


<sup>57</sup><https://www.farnell.com/datasheets/3317027.pdf>

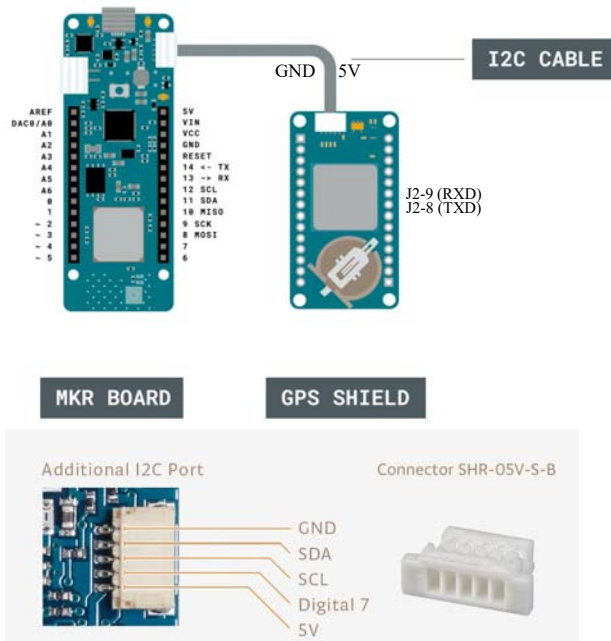


Kortet har også et backup batteri (CR1216) for å “huske” satellitt-konfigurasjoner som programmet trenger for raskt å låse seg til satellittene. Erfaringer viser at låsehastigheten kan økes betydelig ved bruk av batteri, se under. Kortet arbeider på 3,3 V, og kan under spesielle tilfeller trekke opp til 40 mA.

Figuren under viser den delen av koblingskjemaet<sup>58</sup> som omhandler headere og Eslov-kontakten. Her ser vi at de to kommuniserer på henholdsvis UART (Rx/Tx) og I<sup>2</sup>C.



Figuren under viser hvordan MKR GPS kan kobles til MKR NB 1500 med kabel.



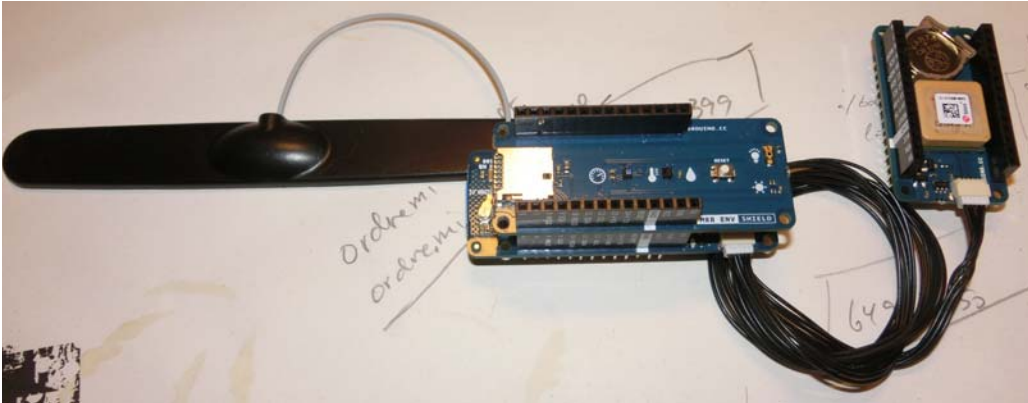
58. [https://content.arduino.cc/assets/MKRGPSShield\\_V4.0\\_sch.pdf](https://content.arduino.cc/assets/MKRGPSShield_V4.0_sch.pdf)





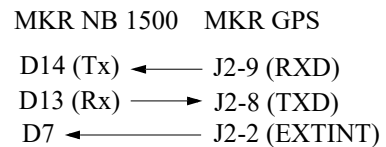
## Montering av MKR GPS

MKR GPS monteres med kabel til kontrollerkortet. Pass på å presse støpselet helt inn i kontakten. På bildet under er MKR GPS-kortet alt montert ved hjelp av kabel.



### I<sup>2</sup>C eller UART (Rx/Tx)

MKR GPS kan enten monteres med kabel til I<sup>2</sup>C-kontakten på mikrokontrollerkortet som vist over eller den kan ev. plugges inn i ryggen til mikrokontrollerkortet (“piggyback”). Ved bruk av piggyback anvendes Rx og Tx portene for kommunikasjon. Disse er angitt som D13 (Rx) og D14 (Tx) hos mikrokontrolleren og J2-9 (RXD) og J2-8 (TXD) hos MKR GPS. Figuren over til høyre viser hvordan de to skal forbindes. I tillegg skal den ha med signalet EXTINT som sannsynligvis er et interrupt-signal fra MKR GPS.



Dersom vi skal bruke GPS som piggyback og anvende UART’en for overføring av data, må vi initialisere GPS-kortet med følgende argument<sup>59</sup>:

```
GPS.begin(GPS_MODE_SHIELD);
```

Default-verdien dersom vi ikke spesifiserer modus i initialiseringen er:

```
GPS.begin(GPS_MODE_I2C);
```

NB! Vi legger også merke til at EXTINT er lagt til samme pinne på mikrokontrolleren som *DRDY Pressure* fra MKR ENV-kortet. Dvs. det ligger an til en konflikt. Dette kan være årsaken til feilen i trykkmålingen vi har registrert, se vedlegg C.6 side 231.

---

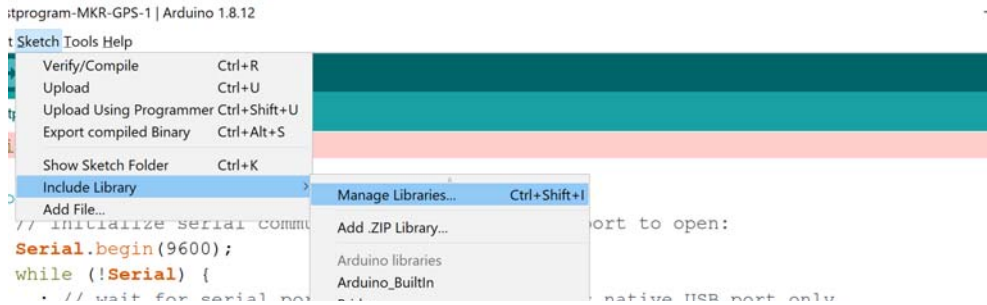
<sup>59</sup>[https://www.arduino.cc/reference/en/libraries/arduino\\_mkrgps/begin/](https://www.arduino.cc/reference/en/libraries/arduino_mkrgps/begin/)



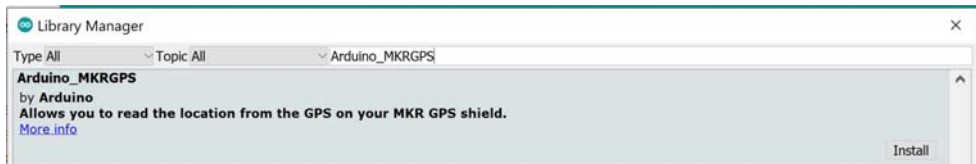
## Programmering – Testprogram for MKR GPS (Testprogram-MKR-GPS-2B.ino)

### 1. Installasjon av bibliotek MKR GPS

For å kunne bruke MKR GPS shield-kortet må vi installere et bibliotek. For å gjøre det går vi inn i menyen *Sketch/Inkludér Bibliotek/Manage Libraries*



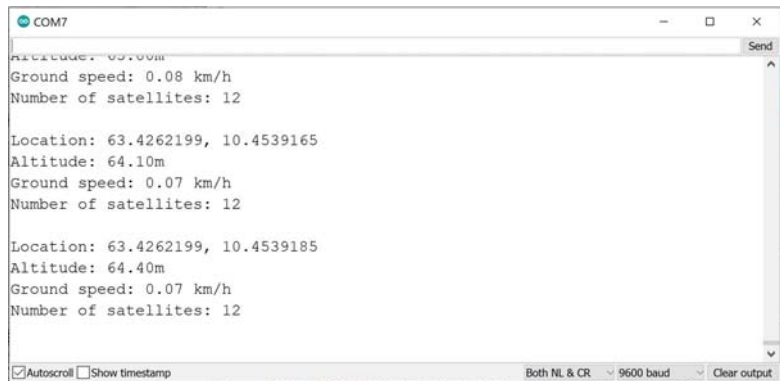
Vi får da opp følgende vindu hvor vi skriver inn *Arduino\_MKRGPS* i søkefeltet.



Vi får opp forslaget vist på figuren over og velger å installere dette biblioteket.

### 2. Kjør testprogrammet

Last opp testprogrammet, se avsnitt I.3 på side 296, og kjør programmet. Pass på å velge riktig port og åpne monitorvinduet. Om alt har gått som planlagt skal du nå se din egen posisjon som vist på figuren over.



Det kan imidlertid ta lang tid før mottakeren får låst seg til tilstrekkelig mange satellitter. Tiden det tar avhenger av plasseringen av mottakeren og i hvilken grad det er firsikt til himmelen.

Legg merke til at programmet viser lengde- og breddegrad og høyde over havet, i tillegg til farten og antall satellitter som den “ser”.



### 3. Studer oppbyggingen av testprogrammet

Det er lurt å ta en titt på hvordan testprogrammet er bygget opp, når vi senere skal se hvordan elementer fra programmet er inkludert i hovedprogrammet vårt.

#### *Inkluder biblioteket*

Vi inkluderer biblioteket ved å sette følgende kommando øverst i programfila:

```
#include <Arduino_MKRGPS.h>
```

#### *Initialisering av GPS-enheten*

For å initialisere og sette opp GPS-enheten bruker vi følgende kommandoer

```
if (!GPS.begin()) {  
    Serial.println("Failed to initialize GPS!");  
    while (1);  
}
```

Initialiseringen gjøres av kommandoen `GPS.begin()`. Denne funksjonen returnerer "1" dersom alt er gått bra. Returnerer den "0" gir den feilmeldingen "Failed to initialize GPS!" og stopper programmet (`while (1);`).

Initialiseringen gjøres i `setup()`-funksjonen.

#### *Sjekk om det er gyldige verdier*

Vi er nå klare for å lese av verdier fra GPS-enheten. Dette gjøres via I<sup>2</sup>C-bussen som er tatt ut gjennom kontakten på siden av MKR NB 1500 kortet. Kommandoen:

```
while(!GPS.available()) {}
```

sjekker om det er mottatt nye gyldige posisjonsdata fra GPS-enheten. Kommandoen `GPS.available()` sjekker om nye posisjonsdata er tilgjengelig. Den returnerer "1" dersom det er tilfelle og "0" om det ikke er tilfelle. Returneres "0" går den inn i `while`-loopen og sjekker på nytt og på nytt til det er gyldige data tilgjengelig, da går den videre i programmet.

Her bør en være klar over at under normale forhold så vil den kunne sjekke flere 1000 ganger før nye data er tilgjengelig 2500 til 4000 er ganske vanlig.

#### *Avlesning av posisjonsdata*

En rekke kommandoer er tilgjengelige fra GPS-enheten for å ta ut data. Her er noen av spesiell interesse:

```
float latitude = GPS.latitude();  
float longitude = GPS.longitude();  
float altitude = GPS.altitude();  
float speed = GPS.speed();  
int satellites = GPS.satellites();  
long Epoch_time = GPS.getTime();
```

Vi legger merke til at variablene som holder de avleste verdiene deklarerer og tilordnes avleste verdier på samme linje.



## GPS biblioteket

GPS-biblioteket inneholder en rekke kommandoer som er nyttige når vi skal hente ut data fra GPS-mottakeren. Det finnes en oversikt over disse her: [https://www.arduino.cc/reference/en/libraries/arduino\\_mkrgps/](https://www.arduino.cc/reference/en/libraries/arduino_mkrgps/)

**Epoch time:** Her vil vi nøye oss med å nevne beregnet Epoketid eller Epoch time.

GPS-mottakeren leverer såkalt Epoch tid (Epoketid) som er tiden målt i sekunder siden midnatt søndag 6. januar 1980. Denne tidsangivelsen tar ikke hensyn til “skuddsekunder” slik f.eks. UTC-tiden gjør. Pr. mars 2019 lå GPS-tiden 18 sekunder foran UTC-tid.

Ønsker man å regne om fra GPS-tid til UTC-tid finnes det kalkulatorer for denne omregningen, se <https://www.labsat.co.uk/index.php/en/gps-time-calculator>. Det finnes også funksjoner i Python som gjør denne omregningen, se avsnitt avsnitt 5.4.4 på side 148.

Den løpende epoketiden siden 1980 finnes på følgende nettside: <https://www.epoch101.com/seconds-in-1980>. Epoketiden oppgis i sekunder siden 00:00 søndag 6.1.1980 og er pr. 08.07.22 kl. 11:25 ca. 1341739518 sekunder. Se <https://timetoolsltd.com/gps/what-is-gps-time/>.

Det finnes også en “Unix time epoch” som referer til en oppstarttid den 01 Jan 1970. *Avlesninger gjort med MKR GPS synes å tyde på at det er denne som leses av med kommandoen:*

```
Epoch_time = GPS.getTime();
```

Et enkelt overslag bekrefter dette. Torsdag 17. nov. 2022 ble epoketiden avlest til 1 668 725 207<sup>60</sup>. Siden det er ca. 31 556 926 sekunder i et år, så får vi at den avleste verdien gir ca. 52,88 år hvilket skulle føre oss tilbake til starten av 1970.

### Sett MKR GPS i “stand by”<sup>61</sup>

Det er viktig å kunne bruke så lite strøm som mulig, også for MKR GPS-kortet. Dette kan gjøres ved å be kortet gå i “stand by” mellom målingene for så å vekkes opp når posisjonen skal bestemmes. Dette gjøres på følgende måte:

#### 1. Sett MKR GPS i “stand by”

Følgende kommando vil sette MKR GPS i “stand by”

```
GPS.standby();
```

Den legges inn i programmet der man ønsker å sette den i “stand by”.

#### 2. Vekk MKR GPS fra “stand by”

Følgende kommando vil vekke MKR GPS fra “stand by”

```
GPS.wakeup();
```

Den legges inn i programmet der man ønsker å vekke kretsen fra “standby”.

---

60. Påpekt og avlest av Thor Inge Hansen, Skien videregående skole

61. <https://www.arduino.cc/en/Reference/ArduinoMKRGPStandby>



### 3. Vent for nye GPS data

Derne kan det ta noe tid for GPS-kretsen har låst seg til et sett med satellitter og er istand til å levere posisjonsdata. dette er som, nevnt avhengig av om det benyttes backup-batteri og hvor lenge den ble liggende i dvale.

```
while (!GPS.available());
```

Denne kommandoen vil stoppe programmet til data er klare fra GPS-mottakeren. Likevel anbefales denne da det kan ta en del tid å komme opp med rett posisjon. Bruker man en “vakt-hund”, bør denne settes på tilstrekkelig lang tid eller resettes med jevne mellomrom.

Normalt trekker MKR GSM 30 mA, og i “stand by” trekker den 70 – 80  $\mu$ A.

Kald start krever en maks oppstart tid på 27 s, varmstart er fra 1 – 2 sek. Målinger viser en akvisisjonstid på 3 – 4 sek etter 10 sek. stand by, og 5 – 10 sek. etter 60 sek. stand by.

### Hvordan unngå at manglende GPS låsing blokkerer programmet

Når man setter spenning på GPS-mottakeren kan den bruke mange minutter på å låse seg til et tilstrekkelig antall satellitter (min. 4 stk.). Er man i tillegg inne i et hus, evt. langt fra vindu, så kan det ta enda lengre tid, evt. at låsing mislykkes.

Problemet løses best ved at man forflytter seg ut eller til et vindu der man har utsikt til en tilstrekkelig stor del av himmelen. Dersom man først har låst til et nødvendig antall satellitter kan man flytte seg inn i rommet og beholde kontakten med satellittene.

Normalt vil mobiltelefoner og annet GPS-utstyr hente inn baneinformasjon via Internet, dermed kan låsing skje raskere. Dersom man ikke har slik tilgang, som er tilfellet for våre micro-kontrollere, så tar det lengre tid og vi må hente ned de nødvendige dataene fra GPS-satellittene. Dersom vi har backup batteri i GPS'en vår, så kan slike data lagres selv om strømmen slås av. Dermed er det mulig å oppnå raskere låsing, selv ved “kaldstart” og manglende Internet, som nevnt.

For å hindre at programmet stopper opp for å vente på låsing, så kan vi i stedet for en while-loop legge inn en for-loop med et begrenset antall tester for tilgjengelige data. Dermed vil programmet komme videre selv om låsing ikke skulle være oppnådd og posisjonsdataene uteblir. Alternativt kan man bruk en while-loop som sjekker for time out, dvs at man har satt en øvre tid for hvor lenge GPS-mottakeren skal lete etter satellitter.

Vi velger å bytte ut while-loopen med følgende for-loop slik at vi får full kontroll over hvor mange runder vi går i loopen:

```
int maxNoChecks = 3000;
int noChecks = 0;

for (int i = 0; i < maxNoChecks; i++)
{
  // Check if there is new GPS data available
  if (GPS.available())
  {
    // If there is, read GPS values
    float latitude = GPS.latitude();
    float longitude = GPS.longitude();
    float altitude = GPS.altitude();
```



```
float speed      = GPS.speed();
int satellites   = GPS.satellites();

// Print GPS values
Serial.print("Location: ");
Serial.print(latitude, 7);
Serial.print(", ");
Serial.println(longitude, 7);

Serial.print("Altitude: ");
Serial.print(altitude);
Serial.println("m");

Serial.print("Ground speed: ");
Serial.print(speed);
Serial.println(" km/h");

Serial.print("Number of satellites: ");
Serial.println(satellites);

delay(1000);
break;
}
noChecks++;
}

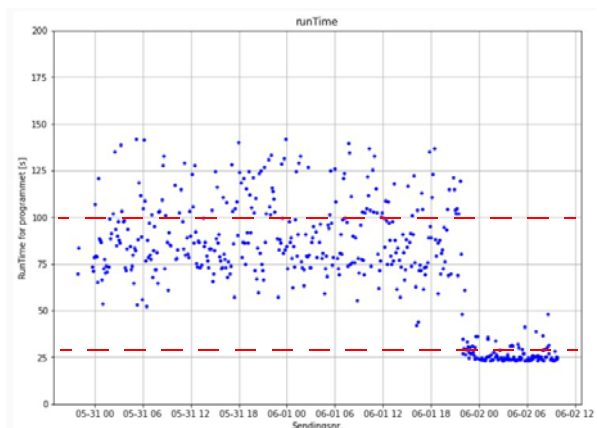
Serial.print("Number of checks: ");
Serial.println(noChecks);
Serial.println();
```

Selv med 3000 runder så vil den i starten ikke oppnå kontakt. Ved plassering i vindu så trenger den 2000 – 3000 runder, og under åpen himmel, enda færre. Programkoden skriver også ut antall runder slik at en kan holde øye med hvor krevende det er å oppnå tilstrekkelig låsing.

### Låsehastighet og bruk av batteri

Som sagt skal batteriet skal gjøre det lettere for GPS-mottakeren å reetablere kontakt med satellittene etter at mottakeren er vekket opp av dvale eller slås på. Figuren til høyre viser hvordan låsetiden ble betydelig forbedret etter at et tomt batteri ble byttet med et nytt.

Vi ser at total på-tid gikk fra ca. 100 sekunder med stor spredning til ca. 30 sek. med relativt lav spredning.





### 4.5.3 MKR MEM Shield

Kortet inneholder en lagringskrets, Flash memory 2MB (W25Q16), og en SD-kortterminal for lagring av data til SD-kort. I tillegg har det et større område for oppkobling av egne kretser.

Programmering av SD-kortet skjer på samme måte som beskrevet i 4.5.1 side 96.

Både flash memory og SD-kortet er koblet til SPI-bussen. Vi bruker “chip select” for å velge mellom disse.

Tabellen under viser hvilke pinner som er brukt for å kommunisere med moderkortet MKR NB 1500.



Pin number	Usage	Notes
4	SD CS	No INT
5	FLASH CS	
8	MOSI	Reserved
9	SCK	Reserved
10	MISO	Reserved

Vi legger spesielt merke til at D4 og D5 er Chip Select for henholdsvis SD-kortterminalen (D4) og flash memory kretsen (D5).

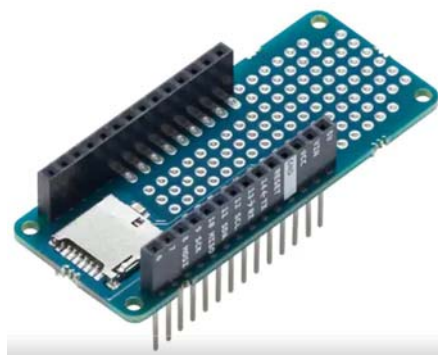
### 4.5.4 MKR SD Proto Shield

Også dette kortet inneholder en SD-kortterminal for lagring av data til SD-kort og et område for oppkobling av egne kretser. Programmering av SD-kortet skjer på samme måte som beskrevet i 4.5.1 side 96. SD-kort terminalen er koblet til SPI-bussen. Vi bruker “chip select” (port D4 = høy) for å velge SD-kort terminalen.

For en kort omtale av kortet, se <https://docs.arduino.cc/hardware/mkr-sd-proto-shield>.

Kortet krever at man installerer SD biblioteket. Se avsnitt 3.6.1 side 57. For mer informasjon og nedlasting av siste versjon av biblioteket se: <https://www.arduino.cc/reference/en/libraries/sd/>

Nettsiden inneholder også oversikt over kommandoer for betjening av filene. Filnavnene kan ha inntil 8 karakterer pluss tre etter punkt. F.eks. DataFile.txt. Filer med lengre navn vil ikke kunne åpnes av programmet.

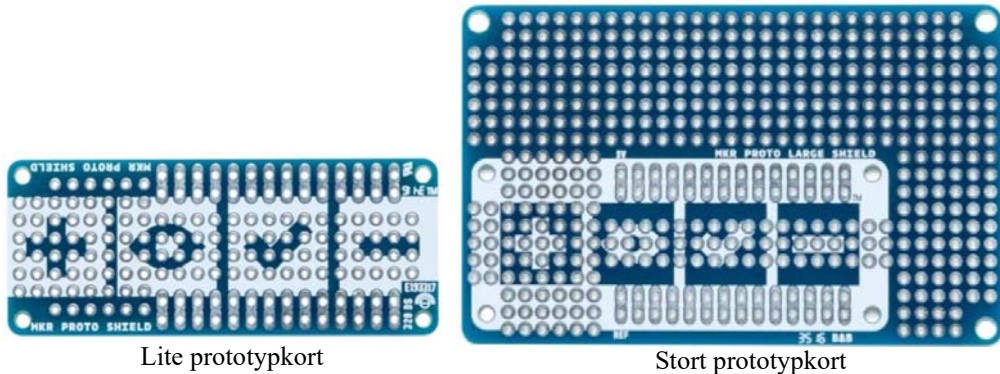






#### 4.5.5 Prototyp-kort

Det finnes både et stort og et lite prototypkort som vist på figuren under.



Begge kortene leveres med løse hylsekontakter. Det lille med gjennomgående lange bein for stabling. Det stor med hylsekontakter med korte bein. Slik det store kortet leveres så er det beregnet på å kunne plasseres nederst i stabelen.

Dersom man f.eks. ønsker å bruke koblingsplinter er det å foretrekke å bruke et stort prototypkort slik at disse kan plasseres på utsiden av den øvrige stabelen. Dersom de skal plasseres inne på det lille prototypkortet vil de lette komme i konflikt med korte som er plassert lenger opp i stabelen. Se avsnitt 6.3.3 på side 167 for oppkobling av temperatursensor for måling av vanntemperatur.

#### 4.6 Energiøkonomisering og bruk av dvale

Ved måling over lengre tid er det viktig å spare på energien. Dette kan vi bl.a. gjøre med å legge kretsene i dvale for så å vekke dem opp når det skal utføres en måling som skal overføres til serveren eller skyen.

For å kunne legge kretsen MKR NB 1500 (og andre i samme serie) i “sleep” og “deep sleep” benytter vi biblioteket<sup>62</sup>:

```
#include "ArduinoLowPower.h"
```

Dette inneholder en rekke kommandoer som er nyttig å vite om:

- **LowPower.idle(millisseconds);**<sup>63</sup>

Dette legger kretsen i lett søvn slik at den raskt kan vekkes. “millisseconds” angir hvor lenge den skal ligge i lett søvn før den vekkes. Dersom “millisseconds” sløfyes vil den sove til den vekkes. Funksjonen kalles i programmet der man ønsker at dvale skal inntreffe.

---

62. <https://www.arduino.cc/en/Reference/ArduinoLowPower>

63. <https://www.arduino.cc/en/Reference/LowPowerIdle>

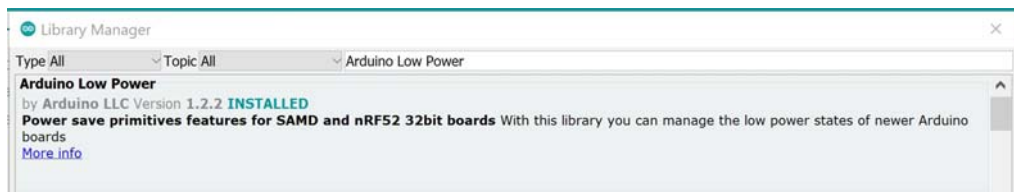


- **LowPower.sleep(millisecods);**<sup>64</sup>  
Denne kommandoen legger kretsen i dypere søvn med noe lenger oppvåkningstid, og med lavere strømforbruk. ”millisecods” angir hvor lenge den skal ligge i lett søvn før den vekkes. Dersom ”millisecods” sløyfes, vil den sove til den vekkes. Funksjonen kalles i programmet der man ønsker at dvale skal inntreffe.
- **LowPower.deepSleep(millisecods)**<sup>65</sup>;  
Alt unntatt RTC er lagt i dvale. Denne modusen har da den lengste oppvåkningstiden og det laveste effektforbruket. ”millisecods” i argumentet angir hvor lenge den skal ligge i dyp søvn før den vekkes. Dersom ”millisecods” sløyfes vil den sove til den vekkes. Funksjonen kalles der man ønsker at dyp dvale skal inntreffe.
- **LowPower.attachInterruptWakeup(pin, callback, mode);**  
Denne kommandoen gjør det mulig å vekke kretsen ved hjelp av en port, ”pin”. Ved oppvekking hopper programmet til ”callback” funksjonen. ”mode” angir om porten ”pin” skal reagere på ”FALLING”, ”RISING” eller på begge, ”CHANGE”. Funksjonen kalles i void setup()-funksjonen.

Vær klar over at Serial.print-kommandoene ikke virker etter oppvåkning hos Arduino IDE versjon 1<sup>66</sup>. Dette synes å være endret ved versjon 2 av IDE'en.

## Installasjon av Low Power biblioteket

Gå til Sketch/Inklude Library/Manage Libraries/ og skriv *Arduino Low Power* i søkefeltet:



Etter en stund får du bl.a. opp forslaget ”Arduino Low Power” som vist på figuren over. Installer biblioteket.

### 4.6.1 Vekking etter en bestemt tid

Dersom vi ønsker å vekke kretsen etter en gitt tid, så kan vi bygge opp programmet slik:

#### 1. Inkluder biblioteket

... som tillater ”sleep mode”

```
#include "ArduinoLowPower.h"
```

Denne legges øverst i programmet

---

64.<https://www.arduino.cc/en/Reference/LowPowerSleep>

65.<https://www.arduino.cc/reference/en/libraries/arduino-low-power/lowpower.deepsleep/>

66.<https://github.com/arduino-libraries/ArduinoLowPower/issues/7>



## 2. Deklarer variabler

Dersom man ønsker å bruke variabler som skal ta vare på verdien mens kontrolleren er i interrupt-funksjonen, så er det lurt å legge disse på sikre plasser (“volatile”). Her er det variabelen `repetitions` som legges på sikker plass:

```
volatile int repetitions = 1;
```

Denne deklarerer som en global variabel.

## 3. Initier sleep

I dette tilfellet trenger vi ikke å initiere dvalefunksjonen. Dersom du likevel ønsker å lede programmet til en oppvåkingsfunksjon, så kan vi legge inn følgende kommando i `void setup()`-funksjonen:

```
LowPower.attachInterruptWakeup(RTC_ALARM_WAKEUP, dummy, CHANGE);
```

Når `RTC_ALARM_WAKEUP` vekker kretsen, så vil den gå til funksjonen `dummy()`; og utføre funksjonen. Dette er imidlertid valgfritt.

## 4. Legg i søvn

Så har vi kommandoen som legger kretsen i dvale, enten idle, `sleep` eller `deepSleep` for en på forhånd bestemt tidsperiode, her “`sleepTime`”.

Denne legges gjerne i `void loop()`-funksjonen eller i en funksjon man selv har skrevet.

```
LowPower.idle(sleepTime);
```

eller

```
LowPower.sleep(sleepTime);
```

eller:

```
LowPower.deepSleep(sleepTime);
```

Alt etter hvor dyp søvn vi ønsker å legge den i, dvs. hvor mye energi vi ønsker å spare.

## 5. Callback()-funksjonen

Dette er funksjonen ditt programmet går når det vekkes opp. I dette eksempelet er det valgfritt om man ønsker å bruke denne.

```
void dummy()  
{  
    // Utføres dersom man har behov.  
}
```

Funksjonen legges gjerne til slutt i programmet, men kan i prinsippet legges hvor som helst, men utenfor andre funksjoner.

## 6. Testprogram for vekking pga. timeout

Det ligger et testprogram i vedlegg H.2 side 290 som demonstrerer dette.

### 4.6.2 Vekking med ekstern trigger koblet til en port

Dersom vi ønsker å vekke den med en ekstern trigger koblet til en port på mikrokontrolleren, f.eks. en ekstern RTC-klokke, kan programmet bygges opp slik:



### 1. Inkluder biblioteket

... som tillater å legge kretsen i dvale:

```
#include "ArduinoLowPower.h"
```

Denne legges øverst i programmet

### 2. Deklarer variabler

Dersom man ønsker å bruke variabler som skal ta vare på verdiene mens kontrolleren er i interrupt-funksjonen så er det lurt å legge dem på en sikker plass ("volatile"). Her er det variabelen `repetitions` som legges på sikker plass:

```
volatile int repetitions = 1;
```

Denne deklarerer som en global variabel

### 3. Initier sleep

Så må vi fortelle programmet hvilken port ("pin") som skal vekke kretsen, hvor den skal gå når den vekkes (her "`repetitionsIncrease`") og hvilken endring i signalet den skal vekkes på (f.eks. "`CHANGE`").

```
LowPower.attachInterruptWakeup(pin, repetitionsIncrease, CHANGE);
```

Denne legges i void `setup()`-funksjonen.

### 4. Legg i søvn

Så har vi kommandoen som legger kretsen i søvn, enten `idle`, `sleep` eller `deepSleep`. Denne legges gjerne i void `loop()`-funksjonen eller i en egendefinert funksjon.

```
LowPower.idle();
```

eller

```
LowPower.sleep();
```

eller:

```
LowPower.deepSleep();
```

Alt etter hvor dyp søvn vi ønsker å legge den i, dvs. hvor mye energi vi ønsker å spare.

### 5. Callback() -funksjonen

Dette er funksjonen der programmet går etter at det er vekket opp. I dette eksempelet teller man kun opp en variabel, derav navnet "`void repetitionsIncrease()`", se eksempelet for nærmere forklaring.

```
void repetitionsIncrease()
{
    repetitions++;
}
```

Funksjonen legges gjerne til slutt i programmet, men kan i prinsippet legges hvor som helst.

### 6. Testprogram for vekking via en port

Det ligger et testprogram i vedlegg H.1 side 289 som demonstrerer dette.



### 4.6.3 Strømforbruk i dvaletilstand

Eksempelprogrammet i vedlegg J.1 side 303 legger kretsen i dvale og tar den ut av dvale etter 5 sekunder. På denne måten får man en veksling mellom 11,7 mA i dvale og 23,1 mA under operasjon. Dvs. ikke særlig lavt strømforbruk i dvale. Dette skyldes primært at den grønne power LED'en lyser.

**Power LED og dvale:** Dersom man tilfører kortet energi gjennom USB eller  $V_{in}$  så legger man merke til at den grønne Power LED'en lyser, hvilket ikke er særlig hensiktsmessig dersom man ønsker å spare energi. Bruker man imidlertid batteriinngangen vil lysdioden slukke<sup>67</sup>.

**Batteriplugg:** Batteripluggen på kortet er av typen hankjønn 2 pin JST PH type. Pluggen på batteriet må dermed være av hunnkjønn 2 pin JST PHR2 type. Ser man rett inn på pluggen på kortet er den positive polen til venstre, og GND (jord) til høyre<sup>68</sup>.

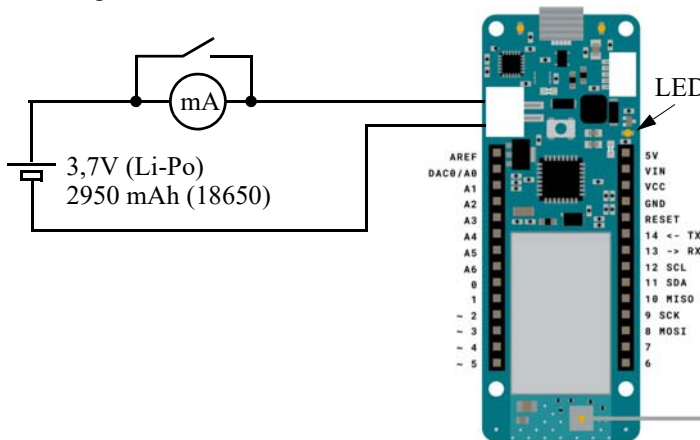
**Valg av batteri:** Det anbefales å bruke 3,7V Li-Po batterier med en kapasitet på 700 mAh eller mer. Bruker man mindre batterier så vil de ikke tåle strømtrekket over tid og vil kunne gå varme. Velger man et batteri på mer enn 1400 mAh så vil ladetiden med den interne laderen bli lang. Lading av batteriet skjer automatisk dersom man kobler til USB-kontakten. Dette stiller seg annerledes ved bruk av ekstern lader.

$V_{in}$ : Dersom man ønsker å tilføre strøm gjennom  $V_{in}$  så vil den optimale spenningen være 5V som ikke under noen omstendigheter må overskride 6V.

La oss se litt på strømforbruket.

#### Målt strømforbruk under ulike forhold

Vi tar utgangspunkt i følgende testkrets:



For å kunne kjøre testprogrammet må biblioteket *Arduino Low Power* installeres.

67. <https://core-electronics.com.au/arduino-mkr-nb-1500-44916.html>

68. <https://core-electronics.com.au/arduino-mkr-nb-1500-44916.html>



Vi lager et enkelt program som slår av og på lysdioden og legger kretsen i dvale. Vi får da følgende strømtrekk:

- Kretsen er våken og LED tent – 21,0 mA, ingen sending
- Kretsen er våken og LED slukket – 16,5 mA, ingen sending
- Kretsen er i deep sleep – 3,7 mA
- Kretsen er i sleep – 3,3 mA

Vi kobler til ENV-kortet og måler strømforbruket da:

- Kretsen er våken, ENV-kortet tilkoblet og LED slukket – 17,1 mA, ingen sending
- Kretsen er i deep sleep – 3,1 mA

Vi kobler til ENV- og GPS-kortet og måler strømforbruket da:

- Kretsen er våken, ENV- og GPS-kortet tilkoblet og LED slukket – 17,4 mA, ingen sending  
GPS-kortet er initialisert
- Kretsen er i deep sleep – 3,1 mA

Vi ser at strømtrekket i dyp søvn (deep sleep) er ca. 3,0 mA.

For ytterligere å senke strømforbruket se: <https://forum.arduino.cc/t/mkr-nb1500-high-power-consumption/597574>

#### **4.6.4 Bruk av ekstern lav energi timer**

Senere i dette kapittelet skal vi se på en løsning som slår strømmen helt av til alle kretser unntatt en klokkekrets som holder rede på måletidspunktene. Denne har vist seg så stabil at vi har realisert løsningen på et kretskort og brukt den i langtidstester. Se avsnitt 4.11 side 129.

#### **4.6.5 Opplasting av programmer til mikrokontroller som er i deep sleep**

Følgende er viktig:

- Når man skal laste opp et nytt program til en mikrokontroller som legges i deepSleep så er sjansen for at man ikke får tilgang til porten stor. For å få lastet opp programmet, må man gi Resett på kortet to raske trykk. Dermed vil booteren være konstant tilgjengelig og man kan laste opp programmet. Mikrokontrolleren vil imidlertid ha skiftet port ved denne operasjonen slik at man må velge port på nytt ved å gå til Tools på menylinja og velge Port.
- Når en MKR NB 1500 lagges i deepSleep slås nesten alle kretser av unntatt RTC (Real Time Clock) som skal holde orden på klokka, i tillegg til noen porter som kan brukes til å vekke kretsen. Ved vekking vil programmet derfor starte på begynnelsen hvilket betyr at setup-rutinen kjøres hver gang, og alle initialiseringer kjøres etter oppstart.



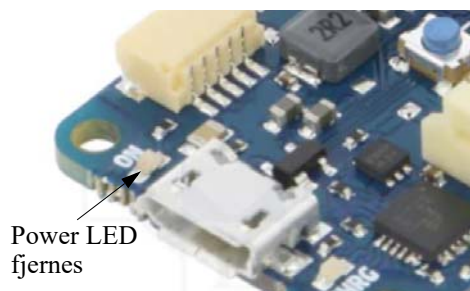
## 4.7 Energikilder

### 4.7.1 Bruk av power bank

Power banker kommer i ulike størrelser til en overkommelig pris. Her har vi kun valgt et eksempel som vist på figuren til høyre: Powerbank 20Ah 18W PD fra ANSMANN<sup>69</sup> (67.2 x 136.8 x 24.8 mm) levert av RS-online for en pris på ca. kr. 900,- (inkl. MVA). Med et gjennomsnittlig strømforbruk på 20 mA vil denne i teorien kunne holde 1000 timer eller ca. 40 døgn. Batteriet kan lades med en 5 eller 12V's kilde og leverer strømmen via USB-A eller USB-C fra inntil tre porter.



Tilførsel strøm via USB-inngangen hos mikrokontrollerkortet er særdeles enkelt, men medfører at power LED'en på kortet lyser og vil trekke strøm. Denne kan evt. fjernes (loddes ut). Andre tilkoblede enheter, som f.eks. MKR ENV sensor kortet og MKR GPS trekker lite strøm, men kan også legges i stand by av programmet når de ikke brukes. Backup batteriet til MKR GPS vil gjøre at oppstart av GPS'en går vesentlig raskere.



#### **NB! – Slår seg av**

*Et problem med flere Power banker er at de slår seg av etter noen få sekunder dersom man ikke trekker nok strøm fra dem, grensen ligger på 50 – 80 mA. Grunnen er at de skal slå seg av når telefonen er fulladet. Dette er et alvorlig problem i vårt tilfelle, da vår krets trekker lite strøm i forhold til en mobiltelefon som står på lading. Å legge inn en ekstra last er ikke noen god løsning da det vil tappe batteriet unødig raskt. En må derfor sørge for å velge en Power bank som ikke har denne funksjonen, hvilket kan være ganske krevende å finne nå til dags.*

### 4.7.2 Bruk av dvale og Li-Po batteri

Denne løsningen ligner mye på den foran (avsnitt 4.7.1), men batteriet byttes ut med et Li-Po batteri (3,7 V) med en passende batteriplugg slik at det kan kobles rett inn i pluggen på MKR NB 1500.

Dette har to fordeler:

1. Man slipper å fjerne Power LED'en som ikke lyser når batteripluggen brukes.
2. Li-Po batteriet lades når man kobler USB-kabelen til mikrokontroller-kortet.

---

69. <https://no.rs-online.com/web/p/power-banks/2019784>





Ulempen er at utvalget slike batterier med en passende plugg er begrenset, dessuten kan de være vanskelig å få tak i. Utvalget blir større dersom man kjøper plugger separat. Vi har foreslått dette batteriet på 1800 mAh (RS PRO, 3.7 V, 53.5 x 35 x 10.4 mm, et Lithium Polymer ladbart batteri<sup>70</sup>), til en pris av ca. kr. 167,- (inkl. MVA).

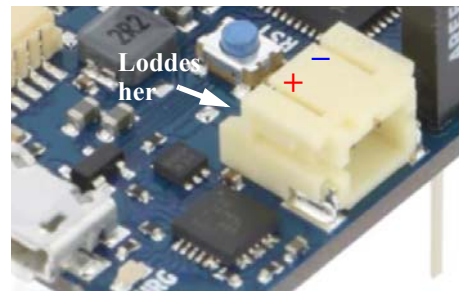
Pluggen er den samme som brukes til Micro:bit og er av typen 2 pin JST PHR2.

Man kan evt. lodde seg rett på kontakten til mikrokontrolleren. Merk at + er til venstre og – til høyre sett rett inn i pluggen, se figuren til høyre.

Ellers legges kretsene i dvale evt. stand by, når det ikke måles eller overføres data.

### Den vanskelige forsendelsen

Det viser seg imidlertid at det kan være problematisk å få tilsendt Li-Po batterier over en viss størrelse, pga. at flyselskaper o.a. er redde for varmgang og eksplosjon. De fleste har vel merket at kabinpersonalet på fly ber passasjerene om å være spesielt oppmerksomme på mobiltelefoner som blir varme eller begynner å ryke.



### 4.7.3 Bruk av Li-Po batterier av typen 18650

Batterier av typen 18650 er ladbare batterier på 3,6 – 4,2V, og er noe større enn vanlige AA-batterier (diameter 18 mm og lengde 65 mm – derav betegnelsen) og krever derfor egne batteriholdere. Batteriene er relativt lette å få tak i, f.eks. hos Biltema og har en kapasitet på opptil 3000 mAh til en pris på ca. 100,- kr. pr. stk. Biltema forhandler også ladere for denne typen batterier som selges for en billig penge.



Batteri  
kr. 100,-



Batterilader  
kr. 70,-

70.<https://no.rs-online.com/web/p/speciality-size-rechargeable-batteries/1449405>



## NB! – Ulik størrelse

En skulle tro at 18650 Li-Po batterier har en standard størrelse. Dette er imidlertid ikke tilfelle. Noen slike batterier har påmontert en beskyttelse som hindrer overlading, og for mye utlading og begrenser kortslutningsstrømmen. Dette går under betegnelsen PCB-protection (PCB – Protection Circuit Board).



RS-online Stock No.: 880-1558

ULEMPEN er imidlertid at batteriet bygger ca. 4 mm i lengden slik at det passer dårlig i standard batteriholdere for 18650-batterier. Biltemas batteri har ikke dette tillegget og passer derfor godt i holderen, men er ikke beskyttet mot høye kortslutningsstrømmer og total utlading, som kan ha uheldige konsekvenser. Batteriet fra RS-online er derimot av denne typen og har problemer med å passe inn i en standard batteriholder.

Batterier til 18650-batterier finnes i ulike utforminger. Vi ønsker å parallellkoble flere celler for å øke kapasiteten, men beholde spenningen på 3,7V. ELFA Diastrelec leverer holdere for en og to batterier. Parallellkoblingen må man selv utføre. Hos Aliexpress<sup>71</sup> får man holdere av alle varianter opp til fire batterier også for disse må man koble sammen batteriene selv.



Som nevnt er kontaktene av samme type som brukes til Micro:bit. Vi har valgt å kjøpe en pakke plugger med kabel fra Banggood<sup>72</sup> for en billig penge. Aliexpress tilbyr det samme, se komponent-listene i vedlegg A side 222. Her må man imidlertid være klar over at det er ingen standard for hva som skal være + og – i kontakten. Det er derfor viktig å forsikre seg om at rød + og sort – er plassert på rett side. Om det skulle være byttet om, kan man ved å vippe opp en liten plastfjær, trekke ut kontaktene og sette dem inn på nytt slik at det blir rett polaritet.

71. <https://www.aliexpress.com/item/1005001861065786.html>

72. [https://www.banggood.com/Excellway-100Pcs-Mini-Micro-JST-2\\_0-PH-2Pin-Connector-Plug-With-120mm-Wires-Cables-p-1147298.html?cur\\_warehouse=CN&rmmms=search](https://www.banggood.com/Excellway-100Pcs-Mini-Micro-JST-2_0-PH-2Pin-Connector-Plug-With-120mm-Wires-Cables-p-1147298.html?cur_warehouse=CN&rmmms=search)



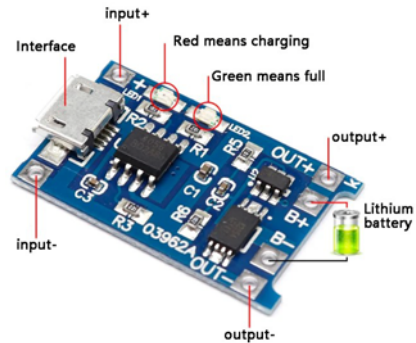
En enkel og billigere måte å skaffe seg batterier på er ved å kjøpe brukte sykkelbatterier på FINN og demontere disse. På innsiden finner man batterier av typen 18650<sup>73</sup>.

## 4.8 Ladere

Dersom vi bruker Li-Po batterier så har vi også behov for lading. Her finnes det mange løsninger. La oss se på to alternativer. En enkel lader med strømtilførsel fra USB og en mer avansert krets fra Adafruit som kombinerer lading fra USB-plugg med lading fra et lite solcellepanel.

### 4.8.1 Enkel og billig USB-lader

Dette er en enkel og billig USB-C lader som finnes i flere varianter som er bygget opp omkring TP4056 kontrolleren. Dokumentasjon finnes på <http://www.tp4056.com/d/tp4056.pdf>. Laderen har flere grunnleggende sikkerhetsfunksjoner som f.eks. kortslutningsvern, som er nyttig for vår anvendelse. Laderen finnes også uten separat tilkobling for batteri og forbrukskretser (output  $\pm$ ). Disse er uten kortslutningsvern og anbefales ikke til elevprosjekter. Det finnes også mulighet for tilførsel av ladespenning på loddbare punkter (input  $\pm$ ). Dette kan være nyttig dersom laderen ønsket brukt sammen med et solcellepanel for energifangst<sup>74</sup>. Selv om lading fra solcellepanel skjer på denne måten, er det ikke sikkert at oppkoblingen gir optimal utnyttelse av solcellepanelet.



Tabellen under viser noen sentrale parametere

Ladespenning inn	4,5 – 5,5 V
Ladespenning ut	4,2 V
Maksimal ladestrøm	1000 mA
Batteribeskyttelse	2,5 V
Kortslutningsvern	3,0 A
Mål	28 x 17 mm

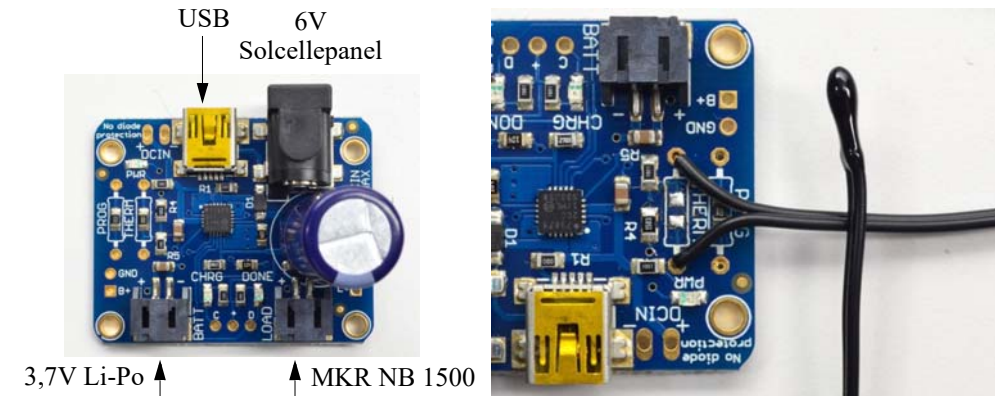
73. Tips gitt av Thor Inge Hansen ved Skien videregående skole

74. Denne laderen er foreslått av Thor Inge Hansen ved Skien videregående skole. Pris ca. 18,00 Aliexpress



#### 4.8.2 Lading fra USB og/eller solceller

Adafruit har utviklet en fleksibel lader for lading av 3,7 V Li-Po batterier som kan kombinere ordinær lading ved hjelp av Power bank eller USB, og solceller<sup>75</sup>.



Siden det kan være skadelig for Li-Po batterier å bli oppvarmet, kan man koble en NTC-temperatursensor til kortet som overvåker temperaturen til batteriet, bilde over til høyre.

Overflatemotstanden på 10 k $\Omega$  som er montert på kortet der det står TERM, fjernes og erstattes med en NTC-motstand på 10 k $\Omega$  som vist på figuren over til høyre. Sensoren limes til batteriet. Dersom temperaturen overskrider en terskelverdi så vil laderen avslutte ladingen av batteriet.

Adafruit anbefaler følgende solcellepanel<sup>76</sup>, men det burde også gå an å lage et eget panel.



6V – 2W  
330 mA  
\$ 29,0  
3.5mm x 1.1mm DC jack  
Vanntett  
En adapter gjør det mulig å koble panelet rett inn på laderen.



75. [https://www.elfadistelec.no/Web/Downloads/\\_t/ds/Adafruit\\_390\\_eng\\_tds.pdf](https://www.elfadistelec.no/Web/Downloads/_t/ds/Adafruit_390_eng_tds.pdf)

76. <https://www.adafruit.com/product/200>



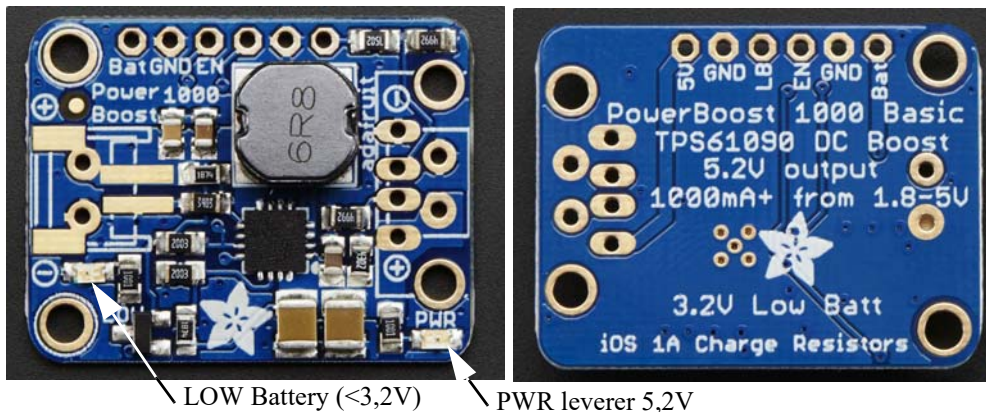
## 4.9 Bruk av power booster

Det viser seg at det i enkelte tilfeller er nødvendig å forsyne mikrokontrolleren og tilleggskort med 5V spenningskilde enten via Vin eller +5V<sup>77</sup>. Bruker vi standard Li-Po-batterier på 3,7 V vil vi ha behov for en DC-DC-konverter som løfter spenningen til 5V, slike kalles også *boostere*.

### 4.9.1 Adafruit Powerboost 1000 Basic<sup>78</sup>

Adafruit Powerboost 1000 Basic er en DC/DC-konverter som kan hente en DC-spenning fra et batteri, gjerne et Li-Po batteri med spenning på 3,7V og løfter denne opp til en spenning på ca. 5,2V. Dette kan være hensiktsmessig for å kunne bruke standard Li-Po batterier til lading av f.eks. mobiltelefoner eller iPad. Li-Po batterer har høy kapasitet og kan levere store strømmer.

Figuren under viser over- og undersiden av Powerboostern.



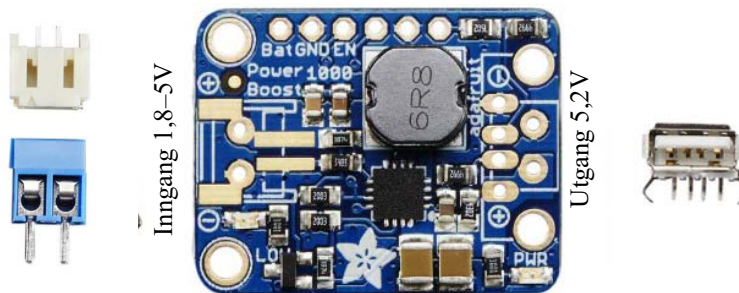
77. Vi har erfart at radiomodemet til MKR NB 1500 har problemer med å starte når kretsen forsyne med et 3,7V batteri. For å unngå problemet har vi valgt å “booste” spenningen opp til 5,2 V og kjøre den inn på USB-inngangen eller også +5V inngangen, noe vi kan gjøre dersom vi er trygg på at spenningen er godt regulert. Se avsnitt 4.11 side 129.

78. <https://learn.adafruit.com/adafruit-powerboost-1000-basic/pinouts>





Spenningen på inngangen kan være fra 1,8 V til 5 V og utgangsspenningen er ca. 5,2 V. Kretsen leveres bl.a. av ELFA<sup>79</sup> og inneholder da løse kontakter slik at man selv kan velge hvilke som passer best til sin applikasjon. En JST-kontakt for Li-Po batteri, USB-A hann-kontakt for lading av mobiltelefoner og annet og en topolt koblingsplint.



### Power-pinner

Tilkobling av energikilden – en inngang kalt BAT og en utgang kalt 5V:

- **BAT** er inngangen der batteriet tilkobles med en spenning fra 1,8 – 5V. Innen dette spenningsområdet vil valg av høyere batterispenning gjøre det mulig å trekke større strømmer ut av boosterens som dermed vil bli mer effektiv. Gjør ledningene kortest mulig for å unngå tap, gjerne under 3 – 4 cm.
- **GND** er jord. Jord på inn- og utgang er ikke isolert, men sammenhengende.

### Kontroll-pinner

Kretsen har to kontroll-pinner:

- **EN** – Dette er “enable” porten som normalt ligger høyt. Ved å dra denne porten lav vil utgangen bli helt skilt fra inngangen. Det er imidlertid ikke angitt hvor mye kretsen trekker av strøm når den er “disabled”.
- **LBO** – Dette er “Low Battery Out” som er en utgang som går lav når batteriet er under 3,2V og kan brukes som en indikasjon på at batteriet bør lades. En rød diode er også knyttet til denne porten. I tillegg lyser lysdioden markert med **LOW**, rødt.
- **5V** – Dette er den oppkonverterte spenningen på ca. 5,2V. Spenningen kan falle til under 5V dersom utgangen belastes med mer enn 1A. En grønn LED er tilknyttet utgangen og lyser grønt når utgangen leverer strøm.

---

79. <https://www.elfadistelec.no/en/powerboost-1000-basic-adafruit-2030/p/30129192>

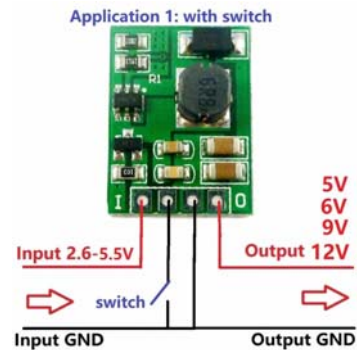


## 4.9.2 Enkel og billig booster fra Aliexpress

En enkel og billig booster kan fås kjøpt fra Aliexpress til en pris av noen få kroner. Kretsen er produsert av firmaet Ele-techsup og bærer betegnelsen DDEN12MA/B. Boosteren kan leveres for ulike utspenninger (5, 6, 9 og 12V). Vi velger 5V til vårt bruk. Målinger viser at den leverer stabilt 5,19 V ubelastet. Boosteren har en enable inngang som gjør det mulig å slå av utspenningen ved å legge denne inngangen høy. Spenningen på inngangen kan være fra 2,6 – 5,5 V.

Tabellen under viser noen nøkkelparametere:

Uinn	Uut	Strøm avslått	Maks strøm	Max effektivitet	Dim
2,6 – 4,5V	5,19V	5uA	600mA	93%	20 x 15 x 6 mm



## 4.10 Bruk av “vakt hund” – Watchdog

En “vakt hund” (watchdog) er en funksjon som enten er en egen krets utenfor eller en intern teller i mikrokontrolleren som passer på at mikrokontrolleren og programmet går som normalt. Dersom programmet stopper opp, evt. går inn i en sløyfe og ikke kommer videre, skal “vakt hunden” hjelpe mikrokontrolleren ut av knipen.

Dette kan gjøres på flere måter, men en måte kan være at en timer hos “vakt hunden” starter når strømmen slås på. Hver gang programmet går en normal runde i loopen, resettes timeren ved hjelp av en puls fra mikrokontrolleren. Dersom programmet stopper opp så vil denne resettpulsen utebli og timeren går til enden og resetter mikrokontrolleren slik at programmet starter på nytt.

Her skal vi først se nærmere på en intern “vakt hund” som benytter en teller i mikrokontrolleren som er uavhengig av programmet.

### 4.10.1 Programvare basert “vakt hund”<sup>80</sup>

Prosessoren har også mulighet til å definere en “vakt hund” som baserer seg på en uavhengig intern timer. Telleren kan settes opp slik at den gir en soft reset til programmet dersom den ikke “mates” med en klarering (reset) før tiden er rent ut. Den baserer seg på et bibliotek som kan lastes ned fra:

<https://github.com/javos65/WDTZero/archive/refs/heads/master.zip>

og installeres fra en zip-file på vanlig måte. Selv om biblioteket ikke primært er skrevet for MKR NB 1500, så er det flere som rapporterer at den fungerer også for MKR NB 1500.

---

80. Informasjon om WDTZero er hentet fra av Thor Inge Hansen ved Skien vgs og er dokumentert i [https://content.arduino.cc/assets/mkr-microchip\\_samd21\\_family\\_full\\_datasheet-ds40001882d.pdf](https://content.arduino.cc/assets/mkr-microchip_samd21_family_full_datasheet-ds40001882d.pdf) side 237 kap. 18.





Biblioteket hentes inn med følgende kode:

```
#include <WDTZero.h>
```

Det defineres en instans for setup()-funksjonen

```
WDTZero MyWatchDoggy;
```

Dernest tilordner man “vakthunden” en tidsfrist for resetting, for at den ikke å gå til time out. Om resetting ikke skjer innen fristen, så vil “vakthunden” restarte programmet:

```
MyWatchDoggy.setup(WDT_SOFTCYCLE1M);
```

som legges i setup()-funksjonen. Konstanten WDT\_SOFTCYCLE1M setter tidsfristen til ca. 1 minutt (1M – 64sek). Lista under som finnes i header-fila i biblioteket, inneholder en rekke slike konstanter for ulike tidsintervaller:

WDT_OFF	WDT off
WDT_HARDCYCLE62m	62.5ms
WDT_HARDCYCLE250m	250msek
WDT_HARDCYCLE1S	1 sek
WDT_HARDCYCLE2S	2 sek
WDT_HARDCYCLE4S	4 sek
WDT_HARDCYCLE8S	8 sek
WDT_HARDCYCLE16	16 sek
WDT_SOFTCYCLE8S	8 sek
WDT_SOFTCYCLE16S	16 sek
WDT_SOFTCYCLE32S	32 sek
WDT_SOFTCYCLE1M	64 sek (ca. 1 minutt)
WDT_SOFTCYCLE2M	128 sek (ca. 2 minutter)
WDT_SOFTCYCLE4M	256 sek (ca. 4 minutter)
WDT_SOFTCYCLE8M	512 sek (ca. 8 minutter)
WDT_SOFTCYCLE16M	1024 sek (ca. 17 minutter)

Dessuten må man resette (klarere) “vakthunden” jevnlig med følgende kommando:

```
MyWatchDoggy.clear();
```

som legges inn i loop()-funksjonen.

#### 4.10.2 Bruk av PICAXE-08M2 som ekstern vakthund

En ekstern “vakthund” er helt frikoblet fra mikrokontrolleren og programmet og kan dermed brukes til å slå av kretsene fullstendig i tillegg til at den kan hjelpe programmet ut av en “hang up”. En enkel og billig løsning synes å være en liten 8 pins mikrokontroller som skreddersys for denne jobben. Av praktiske hensyn foreslås benyttet en PIC-prosessor av typen PICAXE-08M2<sup>81</sup>. Den er billig (2£), kan arbeide på 3,3V og trenger ingen tilleggskomponenter, dessuten trekker den lite

---

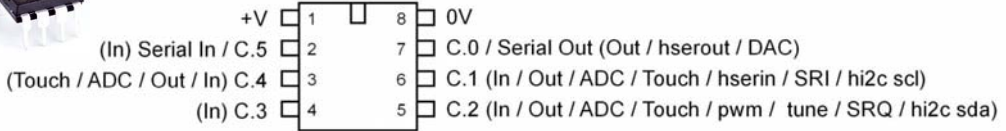
81. <https://picaxe.com/hardware/picaxe-chips/picaxe-08m2-microcontroller/>



strøm. En trenger imidlertid enkelt verktøy for å programmere kretsen<sup>82</sup>. Mikrokontrolleren og verktøyet kjøpes fra samme nettsted<sup>83</sup>, og programvaren kan lastes ned fra nett<sup>84</sup>.

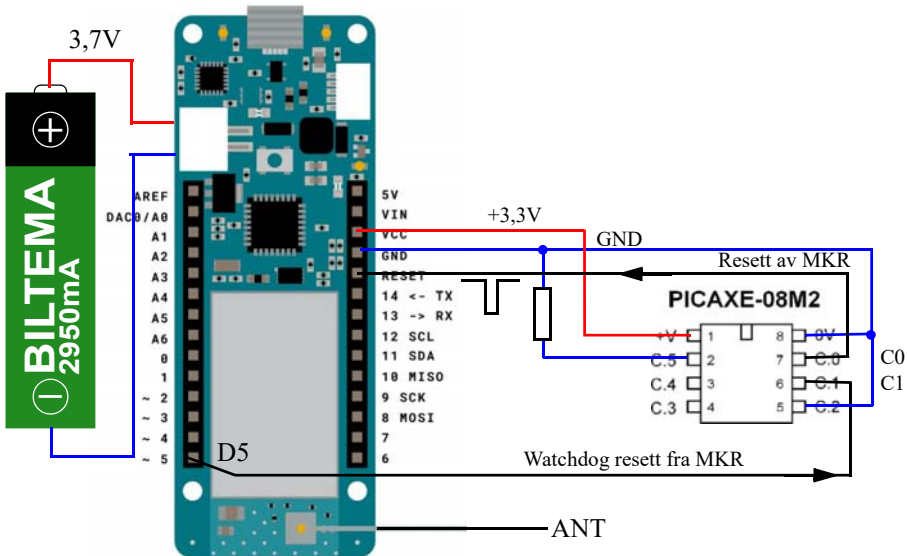


### PICAXE-08M2

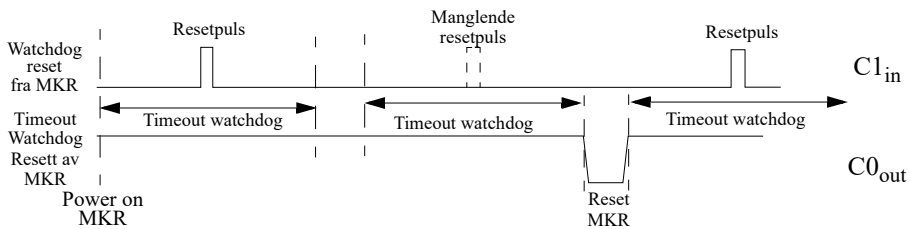


### Enkel oppkobling for test

Det kan være lurt å teste om "vakthunden" fungerer som forventet ved en relativt enkel oppkobling som vist på figuren under:



Et ønsket forløp av resettpulser vil kunne se ut som vist på figuren under:



82. Dette enkle verktøyet kan brukes for å programmere kretsen:

<https://picaxe.com/hardware/project-boards/picaxe-08-proto-board/>

83. <https://picaxe.com/hardware/picaxe-chips/picaxe-08m2-microcontroller/>

84. Last ned og installer programvare: <https://picaxe.com/software/> Velg: PICAXE Editor 6



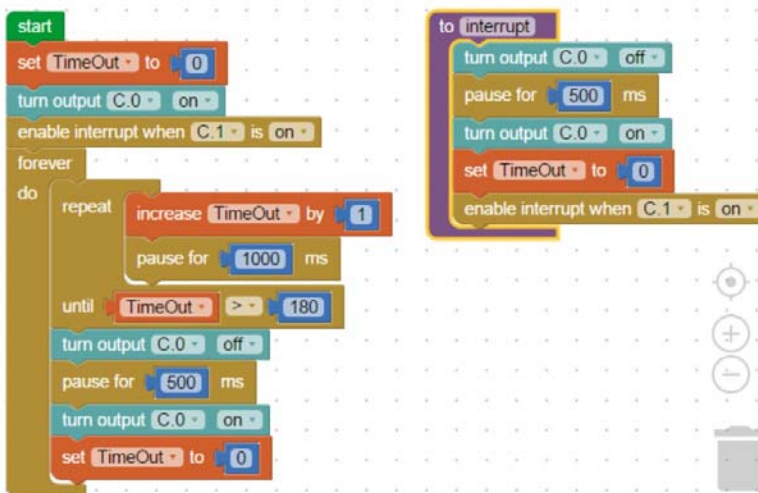
“Timeout watchdog” er tiden "vakhunden" venter før den går til aksjon og resetter MKR mikrokontrolleren. Normalt vil "vakhunden" bli resatt av resett-pulsen før timeout inntreffer. Det er viktig å huske på at første oppkobling til nettet av radiomodemet etter resett, kan ta noe lengre tid enn normalt, dette gjelder spesielt GPS-mottakeren som må låses til satellittene. Imidlertid burde dette gå raskt da GPS-kretsen har backupbatteri som husker parametrene for låsing. Fristen for time out må tilpasses målesekvensen som velges.

### Strømtrekk

Strømtrekket til mikrokontrolleren, PICAXE-08M2+, er målt til typisk 580 $\mu$ A.

#### 4.10.3 Programvare og uttesting av “vakhunden”

Figuren til under viser programmet som er lastet opp i “vakhunden”. C0 – C2 er porter på mikrokontrolleren:



- C0 – er i dette eksempelet, en utgang som resetter MKR-kortet og starter programmet på nytt. C0 er normalt høy, men går lav i 500ms ved time out for å resette MKR.
- C1 – er en inngang som “lytter” etter resett signalet fra MKR-kortet. Når dette kommer, settes TimeOut-variabelen til null (resettes), og “nedtelling” til time out starter på nytt.

Vi legger merke til vi har brukt interrupt for å registrere pulsen fra MKR, som skal resette TimeOut-variabelen. Dermed risikerer vi ikke å gå glipp av en resett-puls.

#### 4.10.4 Bruk av Sced Studio XIAO SAMD21

Dette er en litt dyrere mikrokontroller fra Sced Studio som er Arduino kompatibel og som trekker vesentlig mindre strøm enn PICAXE-08M2+. Det antydes et strømtrekk på 10 $\mu$ A med en timer med en klokkefrekvens på 32kHz<sup>85</sup> i sleep mode.



---

## 4.11 Datalogger med lader, power booster og ekstern “vakhund”<sup>86</sup>

Etter en del prøving og feiling kom vi fram til følgende komplette løsning som viste seg å fungere tilfredsstillende.

### 4.11.1 Forslag til systembeskrivelse

Designet har følgende egenskaper:

1. Håndtere brudd i overføringen til serveren og programmet som stopper opp ... ved bruk av den interne programvarebasert “vakhund”
2. Håndtere dvaletilstand med minimal bruk av energi ... ved å la en ekstern “vakhund” slå av spenningen på samtlige kretser unntatt laderen
3. Lagre data selv om det ikke oppnås forbindelse med serveren ... ved bruk av en lokal datalagringsenhet (SD-kort)
4. Redusere energiforbruket maksimalt ... ved bruke av ekstern mikrokontroller og en spenningsbooster som gjør det mulig å slå av strømmen totalt til mikrokontroller og sensorer.
5. Skape en stabil og robust konstruksjon ... ved oppkobling (lodding) på et skreddersydd kretskort

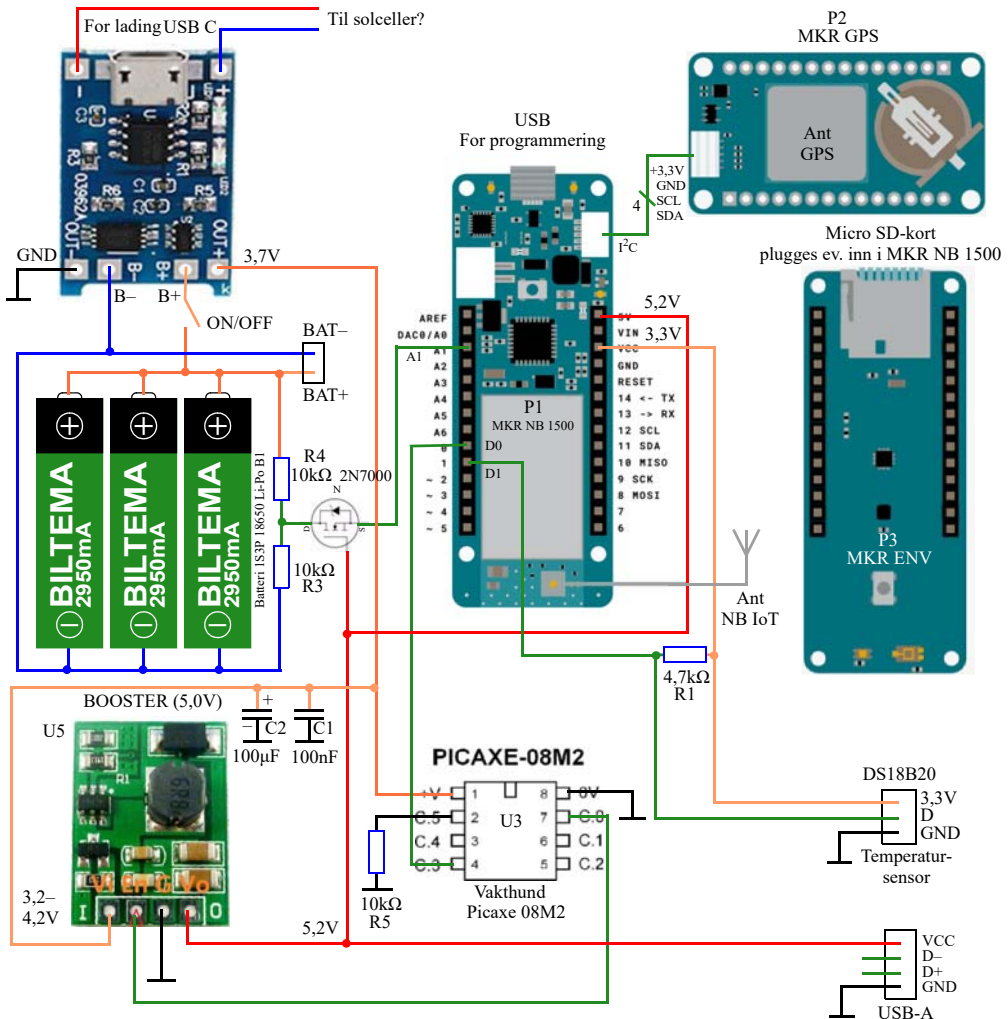
---

85. <https://no.mouser.com/new/seeed-studio/seeed-seeeduino-xiao-pre-soldered-board/#Bullet-2>

86. Denne løsningen er utviklet av Thor Inge Hansen ved Skien vgs i samarbeid med N.K. Rossing



Figuren under viser kretsskjema for dette forslaget til løsning:



#### 4.11.2 Virkemåte

**Mikrokontrolleren:** Kortet er forberedt for montering av en mikrokontroller av typen MKR NB 1500. Mikrokontrolleren plugges ned i et sett med hylsekontakter. Kortet forsynes via 5V-inngangen.

**Miljøkort:** Ulike shield-kort kan plugges inn i “ryggen” til MKR NB 1500. For eksempel kan “miljø-kortet” MKR ENV anvendes. Dette inkluderer måling av lufttrykk, lufttemperatur, luftfuktighet og lys. I tillegg har det en SD-kort terminal for lagring av data.



**GPS-kort:** MKR GPS er en GPS-mottaker som lett kan tilsluttes MKR NB 1500 og gi GPS data til måleserien. Vi har valgt å montere den ved siden av kortet med en kabeltilslutning (I<sup>2</sup>C-buss) til mikrokontrolleren.

**Spenningsforsyning:** Mange ulike batteriløsninger har vært prøvd. Den vi har valgt er å benytte Li-Po-batterier som leverer en nominell spenning på 3,7 V, løfte den opp til ca. 5.2 V ved hjelp av en spenningsbooster og tilføre den gjennom +5 V inngangen til mikrokontrolleren.

**Batteripakken:** I dette tilfellet består denne av tre Li-Po 18650 batterier koblet i parallell (1S3P). De er plassert i en batteriholder til dette formålet. Batteriene fra Biltema er uten strømbegrenser hvilket betyr at dersom de kortsluttes vil det kunne gå strømmer på opp mot 10A. Batterier med beskyttelse (PCB-protection) er fysisk litt lengre, og er noe man må ta hensyn til ved valg av batteriholder. Fulladede batterier gir en spenning på ca. 4,2 V.

**Lading:** Batteriene lades enten enkeltvis utenfor batteripakken i egne ladere ala de som selges fra Biltema (se avsnitt 4.7.3 side 119). Eller med laderen montert på kortet<sup>87</sup> (se 4.8.1 side 121). For at lading skal kunne skje på kortet må bryteren B1 være påslått. Dette betyr at også mikrokontrolleren med tilhørende kretser får spenning. Ved lading lyser en rød LED, ved ferdig lading lyser en grønn LED.

**Spenningsboosteren:** Denne løfter spenningen fra ca. 3,2 – 4,2 V til ca. 5,2 V. Batterispenningen kan variere fra fulladet (4,2 V) til nesten utladet (3,2 V). Boosteren vil sørge for at spenningen inn på mikrokontrolleren vil holde seg konstant. Av en noe uforklarlig grunn ser det ut til at radiomodemet på MKR NB 1500 ikke starter dersom vi tilfører spenning til Vin, men fungerer greit når den tilføres +5V. Dette medfører at noe av spenningsbeskyttelsen som sitter i MKR NB 1500 utelates og vil dermed måtte overtas av laderen og boosterens. Boosteren har en enable-inngang som slår av spenningen på utgangen dersom den legges lav.

**“Vakthunder”:** Det finnes to “vakthunder” i denne løsningen: En programaktivert “vakthund” (intern) som sørger for å hoppe ut av programmet om det låser seg, og en eksterne (PICAXE) “vakthund” som sørger for å slå av strømmen for å forberede restart eller for å legge kretsen i kontrollert i dvale (dvs. med avslått strøm) inntil neste måling skal gjøres. Dersom programmet “henger seg” vil den interne “vakthunden” sørge for å hoppe til en funksjon som sender et signal (aktivt høyt) til den eksterne “vakthunden” (via D0 på MKR NB 1500 til C3 på PICAXE) som sørger for å slå av strømmen. Ved neste oppstart vil kretsen bli totalt resatt.

**Programmet:** Programmet som er brukt i denne løsningen finnes i vedlegg K side 312. Hele den aktive programkoden ligger i setup()-funksjonen som kjøres når programmet starter opp et at strømmen slås på. På slutten av setup()-funksjonen sendes et signal til den eksterne “vakthunden” som slår av strømmen for så å slå den på igjen når ventetiden mellom målinger er omme.

---

87. Foreslått av Thor Inge Hansen fra Skien videregående skole



**Ekstern “vakt hund”:** Den eksterne “vakt hund” er den lille mikrokontrolleren (PICAXE 08M2+) som må programmeres separat. Denne trekker særdeles lite strøm (580  $\mu$ A) og kan derfor være på hele tiden. Den er koblet direkte til utgangen av laderen (batteriet) slik at den får strøm selv om boosterens slår av spenningen på mikrokontrolleren med tilhørende kretser.

Når måleprogrammet har gjort seg ferdig, sendes en puls til den eksterne “vakt hund” (C3 legges lav) om at nå kan strømmen slås av (C0 legges høy). Den slår da av strømmen via boosterens og starter en ventesløyfe som tilsvarer ventetiden mellom to målinger. Når så ventetiden er ute, slår den på boosterens (C0 legges lav) som gir spenning til mikrokontrolleren, som starter måleprogrammet.

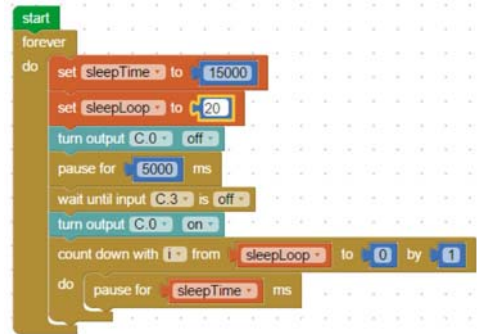
Dersom programmet “henger seg” vil den interne “vakt hund” sørge for at den eksterne “vakt hund” får beskjed og slår av spenningen og går inn i ventesløyfa. På denne måten vil man sannsynligvis miste en måling, men programmet blir resatt og man er klar til en ny måling “med blanke ark”.

Figuren til over høyre viser programmet hos “vakt hund” som er skrevet i blokkode. For nærmere beskrivelse av programmering av PICAXE kontrollere, se referanse [10].

**Måling av batterispenning:** Det er interessant å følge med batterispenningen etter som tiden går. Siden batterispenningen kan bli hele 4,2 V så vil den normalt ligge over arbeidsspenningen på mikrokontrolleren som er 3,3 V. For å kunne måle spenninger på over 3,3 V, halverer vi spenningen med en enkel spenningsdeler og kompenserer for det i programmet.

**MOSFET-bryter:** En MOSFET er plassert mellom batterispenningen og hindrer at batterispenningen blir liggende på inngangen (A1) når forsyningspenningen er avslått.

**Måling av temperatur:** En koblingsplint for tilkobling av temperatursensoren DS18B20 er inkludert på kortet. Sensoren leveres i flere utforminger også i en vanntett utgave som vi benytter her. DS18B20 overfører data på en enkelt tråd i tillegg til VCC og GND, se avsnitt 6.3.3 side 167 for nærmere beskrivelse.







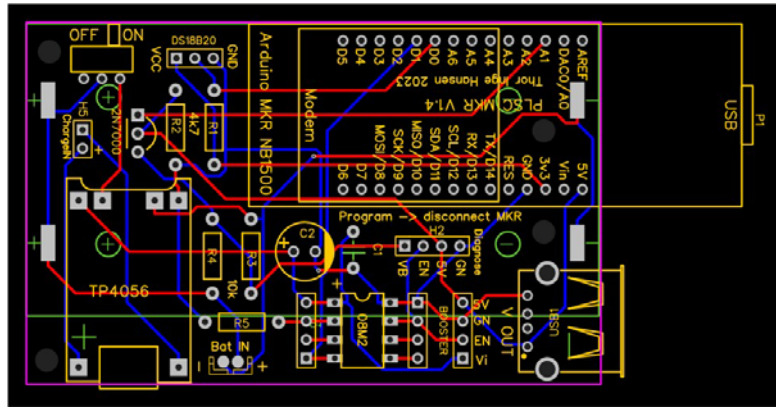
### Andre sensorer:

Andre sensorer kan tilkobles porter på MKR NB 1500 ved å bruke et prototyp shield-kort. Her kan det gjøres plass til både analoge og digitale sensorer.

### 4.11.3 Kretskort<sup>88</sup>

Til den omtalte løsningen er det laget to kretskort. Ett som inkluderer MKR NB 1500 og ett som er mer generelt og dermed kan brukes til andre typer mikrokontrollere som kan forsynes av 5V. Kortet er også forberedt for montering av to batterier av typen 18650 i holder på undersiden.

For komplett oversikt over kretsskjema og PCB'er se vedlegg B side 225.



<sup>88</sup>Kretskortet er lagt ut av Thor Inge Hansen ved Skien vgs



## 5 Presentasjon og behandling av måledata

I dette kapittelet skal vi gi eksempler på hvordan vi henter inn data til Excel og Google Earth for visning av resultatene. I tillegg skal vi se hvordan vi kan bruke Python<sup>89</sup> til å lese, plote og analysere data.

### 5.1 Skrivning til og henting av data fra server

#### 5.1.1 Skrivning til fil på serveren

De innsamlede dataene fra hver deltaker eller gruppe legges i den samme katalogen på serveren. Hver deltager kan imidlertid opprette en unik fil for ulike måleserier om det er ønskelig.

Katalogen er bestemt av eieren av serveren, i dette tilfellet Inst. for marin teknikk. Filnavnet bestemmes av et av argumentene i bufferet som overfører datapakken. Vi har tidligere vist oppbyggingen av tekststrengen som legges inn i bufferet (se også 3.7.2 side 70):

```
sprintf(buffer, "/cgi-bin/tof.cgi?Gruppe-1.txt",
, %d, %.2f, %.6f, %.6f, %.1f, %.1f, %.1f, %.1f, %.1f",
no, timeSec, longitude, latitude, temperature, w_temperature,
humidity, pressure, illuminance);
```

Første argument `/cgi-bin/tof.cgi` – `cgi` står for Common Gateway Interface og er et lite script (program) som web-servere bruker for å styre data på serveren. I vårt tilfelle vil scriptet ta tak i det neste argumentet etter “?” i dette eksempelet `Gruppe-1.txt` og opprette en fil med dette navnet, evt. skrive inn i en eksisterende fil med det samme navnet. Derneft legges dataene fortløpende inn i filen i henhold til ønsket format.

Det er derfor viktig at hver enkelt er seg bevisst hvilket filnavn som brukes slik at de finner igjen data sine.

#### 5.1.2 Lesing av data fra fil på serveren

Gå til følgende webadresse:

```
http://sensor.marin.ntnu.no/logs/
```

Dataene finnes under fila:

```
http://sensor.marin.ntnu.no/logs/<filnavn>.txt
```

Kursdeltakerne må selv holde orden på filnavnene slik at dataene ikke blandes.

Man kan åpne fila med et program som kan åpne tekst-filer, f.eks. Notepad++ eller lignende. Innholdet kan enten kopieres eller hele fila kan lagres for senere å hentes opp i f.eks. Excel.

---

<sup>89</sup>.Python programmene er utviklet av Thor Inge Hansen ved Skien vgs.



## 5.2 Skrive data til fil

Følgende gir noen anbefalinger for organisering av data:

### 5.2.1 Lagre rådata

Dersom man ønsker å redusere sannsynligheten for å miste data pga. feil i programmeringen, vil det være fornuftig å sende over, eller lagre rådata. Behandlingen av dataene kan likevel lett gjøres i etterbehandlingen i f.eks. Excel, om det skulle være nødvendig. Overfører man beregnede data og man er så uheldig å regne feil i sensornoden før data lagres eller overføres til serveren, så kan dataene være tapt for alltid.

### 5.2.2 Tidsangivelse

Inkluder en teller og en tidsangivelse for når dataene ble samlet inn. En teller kan være grei for å se om man har mistet noen målinger under veis. En annen måte er å skrive inn tiden fra Arduino'en ble startet (evt. restartet), ved bruk av funksjonen:

```
millis();
```

som returnerer en verdi som er lik antallet millisekunder siden mikrokontrolleren ble startet/restartet. En må imidlertid være klar over at funksjonen vil gå ut over sitt område etter ca. 50 dager, hvilket kan være for lite i noen tilfeller.

En annen mulighet er å bruke tidsangivelsen hentet fra GPS-mottakeren ("Epoch time") og legge den inn i fila. En nøyaktig tidsangivelse vil være essensiell dersom målinger skal brukes av meteorologisk institutt eller andre. Se *GPS biblioteket på side 108*. En fordel med å bruke epoketiden er at den er uavhengig om sensornoden blir slått av eller legges i dyp dvale. På den annen siden vil dette tidsstempet gå tapt dersom man ikke oppnår kontakt med GPS-satellittene.

En tredje mulighet er å bruke en RTC-klokke (Real Time Clock). Denne kan også vekke eller slå på mikrokontrolleren dersom denne legges i dvale eller slås av mellom målingene.

### 5.2.3 Skilletegn

Verdiene som skrives til fil skilles med et skilletegn. Her kan man i prinsippet bruke ulike tegn f.eks.: <> ; , eller tab. Dersom man bruker "." og ",", må man være klar over hva som brukes som desimalpunkt i den aktuelle sammenhengen. Imidlertid krever KML-fila med GPS-koordinatene at koordinatene skilles med komma, vi har derfor i denne sammenhengen valgt å bruke "," siden "." brukes som desimaltegn. Skal man bruke Python script for å behandle dataene så anbefales også ",".

### 5.2.4 Simulerte data

For å teste ut bruk av Excel kan man lage en testfil med simulerte data. I filen beskrevet under har vi kun brukt en teller og "," som skilletegn. I dette tilfellet går dette greit fordi vi vet at det tas én punktprøve hvert 5. sekund og at desimaltegnet er et punktum.

```
Serial.print(no); // Målenummer
```



```
Serial.print(",");
Serial.print(epochTime); // Epoketid GPS i sekunder siden 1.1.80
Serial.print(",");
Serial.print(longitude,6); // Posisjonens lengdegrader
Serial.print(",");
Serial.print(latitude,6); // Posisjonens breddegrader
Serial.print(",");
Serial.print(temperature,1); // Målt lufttemperatur i grader C
Serial.print(",");
Serial.print(w_temperature,1); // Målt vanntemperatur i grader C
Serial.print(",");
Serial.print(humidity,1); // Målt luftfuktighet i % rel. fuktighet
Serial.print(",");
Serial.print(pressure,1); // Målt lufttrykk i mBar
Serial.print(",");
Serial.println(illuminance,1); // Målt lysintensitet ved overflata i lux
```

Figuren under viser den endelige filen vist ved hjelp av Notepad++ :

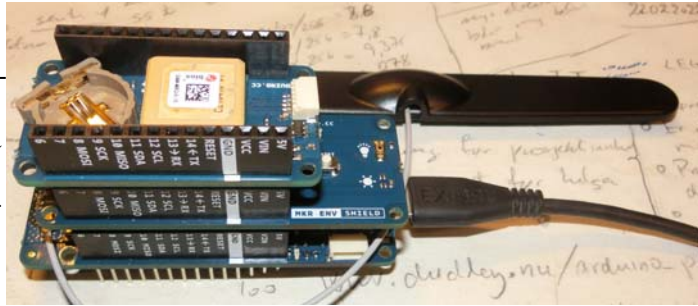
```
1 No, EpochTime, Longitude, Latitude, Temperature, Water temp., Humidity, Pressure, Illuminance
2 1, 1341739327, 10.454045, 63.426466, 21.8, 12.0, 46.1, 1090.0, 148.8
3 2, 1341739332, 10.454145, 63.427071, 21.8, 12.0, 45.7, 1089.9, 149.0
4 3, 1341739337, 10.454345, 63.426571, 21.6, 12.0, 46.3, 1089.5, 149.3
5 4, 1341739342, 10.454745, 63.426369, 22.1, 12.0, 45.6, 1089.6, 149.0
6 5, 1341739347, 10.454045, 63.426968, 22.3, 12.0, 46.3, 1089.4, 149.3
7 6, 1341739352, 10.454645, 63.426868, 21.6, 12.0, 45.5, 1089.3, 148.9
8 7, 1341739357, 10.454345, 63.426968, 22.1, 12.0, 45.9, 1089.6, 148.8
9 8, 1341739362, 10.454845, 63.426571, 21.5, 12.0, 46.2, 1089.8, 149.0
10 9, 1341739367, 10.454545, 63.426369, 22.2, 12.0, 45.7, 1089.8, 148.8
11 10, 1341739372, 10.454845, 63.426270, 21.7, 12.0, 46.3, 1089.6, 149.1
12 11, 1341739377, 10.454845, 63.427071, 21.6, 12.0, 45.7, 1089.8, 148.7
13 12, 1341739382, 10.454545, 63.426670, 21.9, 12.0, 45.6, 1089.9, 148.9
14 13, 1341739387, 10.454145, 63.426968, 22.1, 12.0, 46.2, 1090.1, 149.3
15 14, 1341739392, 10.454045, 63.426369, 21.9, 12.0, 45.8, 1089.9, 148.7
16 15, 1341739397, 10.454145, 63.426670, 21.8, 12.0, 46.1, 1089.9, 148.5
17 16, 1341739402, 10.454745, 63.426571, 21.5, 12.0, 45.5, 1089.8, 149.0
18 17, 1341739407, 10.454545, 63.426968, 21.6, 12.0, 46.1, 1089.5, 148.8
19 18, 1341739412, 10.454745, 63.426968, 22.1, 12.0, 45.8, 1089.7, 148.7
20 19, 1341739417, 10.454645, 63.426968, 22.3, 12.0, 46.0, 1089.9, 149.0
21 20, 1341739422, 10.454145, 63.426369, 22.0, 12.0, 45.5, 1089.8, 148.7
22 21, 1341739427, 10.454745, 63.426769, 22.2, 12.0, 45.9, 1089.9, 148.8
23 22, 1341739432, 10.454245, 63.426270, 21.5, 12.0, 46.1, 1089.8, 149.3
24 23, 1341739437, 10.454245, 63.426270, 21.9, 12.0, 45.8, 1089.7, 149.2
25 24, 1341739442, 10.454845, 63.426968, 22.2, 12.0, 45.5, 1090.1, 148.8
26 25, 1341739447, 10.454545, 63.426769, 22.2, 12.0, 45.5, 1090.1, 149.1
27 26, 1341739452, 10.454145, 63.427071, 21.9, 12.0, 45.9, 1089.3, 149.1
28 27, 1341739457, 10.454345, 63.426670, 22.2, 12.0, 45.7, 1089.8, 149.2
29 28, 1341739462, 10.454645, 63.426571, 22.0, 12.0, 45.6, 1089.7, 148.7
30 29, 1341739467, 10.454345, 63.427071, 21.6, 12.0, 46.2, 1089.6, 149.3
31 30, 1341739472, 10.454845, 63.426868, 22.2, 12.0, 45.8, 1089.3, 148.7
```

Legg merke til at vi har fått med en tekst øverst. Dette er en tekst som er skrevet ut i setup()-funksjonen slik at den kun kommer med en gang. Dette er bare mulig dersom vi lar være å slå av spenningen på mikrokontrolleren mellom hver måling. Det er viktig at vi lagrer fila som en vanlig tekstfil eller CSV-fil (Comma Separated Values).



## 5.3 Bruk av Excel for visualisering av data

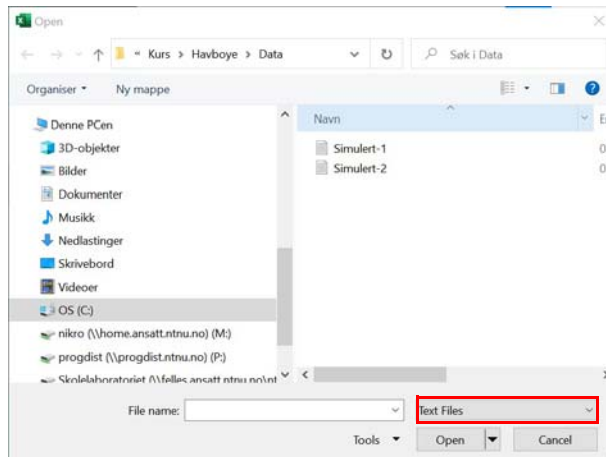
Et praktisk og vanlig verktøy for analyse av data er Excel. I dette avsnittet skal vi vise hvordan vi kan importere og tegne ut grafer av innsamlede data fra bøya. I dette eksempelet har vi følgende data: Måling nr., tidsangivelse (Epoketid), lengde- og breddegrad, luft- og vann-temperatur, luftfuktighet, lufttrykk og lysintensitet på overflata. Dataene for dette eksempelet er simulert.



### 5.3.1 Importer data fra en tekst-fil til Excel

Importer av data i Excel kan gjøres på ulike måter. En måte som fungerer er å hente inn filen med "Open". Da vil man ledes gjennom følgende vinduer for at dataene skal bli formatert på ønsket måte.

Velg ønsket fil:



Siden det er en tekst-fil så velg denne typen format i innboksen nederst i høyre hjørne, dermed blir slike filer synlige i fil-oversikten.



Velg “Data med skilletegn” (Delimited) og bestem fra hvilken rad du ønsker å starte å importere data. Likeså er det viktig å merke av om det brukes topptekst, som vi har gjort her. Trykk så “Neste”:

Text Import Wizard - Step 1 of 3

The Text Wizard has determined that your data is Delimited.

If this is correct, choose Next or choose the data type that best describes your data.

Original data type

Choose the file type that best describes your data:

- Delimited - Characters such as commas or tabs separate each field.
- Fixed width - Fields are aligned in columns with spaces between each field.

Start import at row: 1 File origin: MS-DOS (PC-8)

My data has headers.

Preview of file C:\D\Arduino\Kurs\Havboye\Data\Simulert-2.txt.

```
1 No, EpochTime, Longitude, Latitude, Temperature, Water temp., Humidity, Pressure, Illumi
2 1, 1341739327, 10.454045, 63.426468, 21.8, 12.0, 46.1, 1090.0, 148.8
3 2, 1341739332, 10.454145, 63.427071, 21.8, 12.0, 45.7, 1089.9, 149.0
4 3, 1341739337, 10.454345, 63.426571, 21.6, 12.0, 46.3, 1089.5, 149.3
5 4, 1341739342, 10.454745, 63.426369, 22.1, 12.0, 45.6, 1089.6, 149.0
6 5, 1341739347, 10.454045, 63.426968, 22.3, 12.0, 46.3, 1089.4, 149.3
7 6, 1341739352, 10.454645, 63.426868, 21.6, 12.0, 45.5, 1089.3, 148.9
```

Buttons: Cancel, < Back, Next >, Finish

Velg hvilken type skilletegn som er benyttet. I vårt tilfelle har vi benyttet “komma”.

Text Import Wizard - Step 2 of 3

This screen lets you set the delimiters that your data contains. You can see how your text is affected in the preview below.

Delimiters

- Tab
- Semicolon
- Comma
- Space
- Other:

Treat consecutive delimiters as one

Text qualifier: "

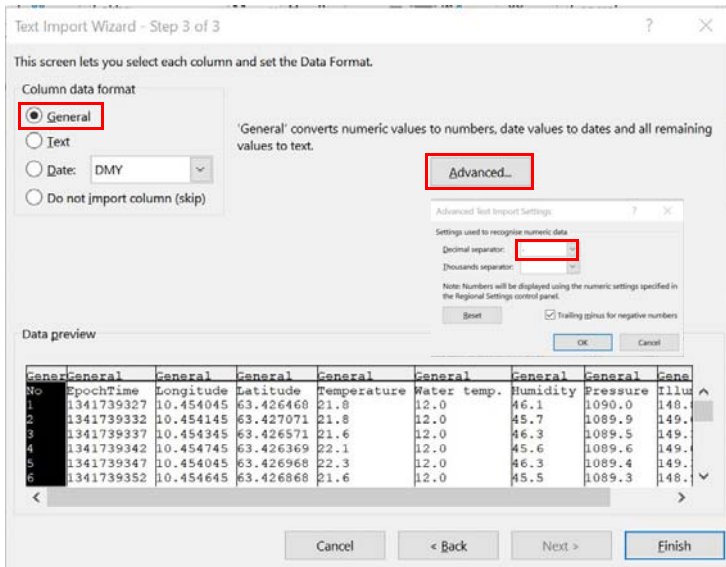
Data preview

No	EpochTime	Longitude	Latitude	Temperature	Water temp.	Humidity	Pressure	Illum
1	1341739327	10.454045	63.426468	21.8	12.0	46.1	1090.0	148.8
2	1341739332	10.454145	63.427071	21.8	12.0	45.7	1089.9	149.0
3	1341739337	10.454345	63.426571	21.6	12.0	46.3	1089.5	149.3
4	1341739342	10.454745	63.426369	22.1	12.0	45.6	1089.6	149.0
5	1341739347	10.454045	63.426968	22.3	12.0	46.3	1089.4	149.3
6	1341739352	10.454645	63.426868	21.6	12.0	45.5	1089.3	148.9

Buttons: Cancel, < Back, Next >, Finish



Tilslutt velges hvilken type data det gjelder, i vårt tilfelle er det standard tallformat (General). Dersom vi skal importere flyttall (med komma), velges desimalpunktet, “,” eller “.” ved å velge “Avansert”. Det må vi gjøre i vårt tilfelle, ellers har dataene en tendens til å bli til datoer eller annen tekst.



Dermed skal dataene være importert i regnearket som vist på figuren under.

	A	B	C	D	E	F	G	H	I	J
1	No	EpochTime	Longitude	Latitude	Temperature	Water temp.	Humidity	Pressure	Illuminance	
2	1	1,34E+09	10,45405	63,42647	21,8	12	46,1	1090	148,8	
3	2	1,34E+09	10,45415	63,42707	21,8	12	45,7	1089,9	149	
4	3	1,34E+09	10,45435	63,42657	21,6	12	46,3	1089,5	149,3	
5	4	1,34E+09	10,45475	63,42637	22,1	12	45,6	1089,6	149	
6	5	1,34E+09	10,45405	63,42697	22,3	12	46,3	1089,4	149,3	
7	6	1,34E+09	10,45465	63,42687	21,6	12	45,5	1089,3	148,9	
8	7	1,34E+09	10,45435	63,42697	22,1	12	45,9	1089,6	148,8	
9	8	1,34E+09	10,45485	63,42657	21,5	12	46,2	1089,8	149	
10	9	1,34E+09	10,45455	63,42637	22,2	12	45,7	1089,8	148,8	
11	10	1,34E+09	10,45485	63,42627	21,7	12	46,3	1089,6	149,1	
12	11	1,34E+09	10,45485	63,42707	21,6	12	45,7	1089,8	148,7	
13	12	1,34E+09	10,45455	63,42667	21,9	12	45,6	1089,9	148,9	
14	13	1,34E+09	10,45415	63,42697	22,1	12	46,2	1090,1	149,3	
15	14	1,34E+09	10,45405	63,42637	21,9	12	45,8	1089,9	148,7	
16	15	1,34E+09	10,45415	63,42667	21,8	12	46,1	1089,9	148,5	
17	16	1,34E+09	10,45475	63,42657	21,5	12	45,5	1089,8	149	
18	17	1,34E+09	10,45455	63,42697	21,6	12	46,1	1089,5	148,8	
19	18	1,34E+09	10,45475	63,42697	22,1	12	45,8	1089,7	148,7	
20	19	1,34E+09	10,45465	63,42697	22,3	12	46	1089,9	149	





Siden vi ikke trenger å gjøre beregninger på noen av dataene, så kan vi illustrere verdiene i hver kolonne som grafer om vi skulle ønske det.

### 5.3.2 Lage grafer

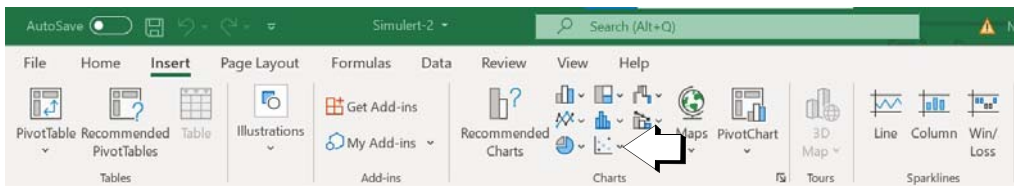
Slik kan vi lage grafer:

1. Merk de to kolonnene som skal representeres av en graf.

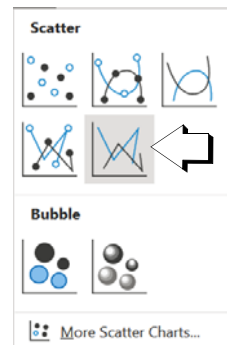
Det kan være lurt å lage en overskrift på kolonnene om det ikke alt er gjort. Overskriften vil automatisk tilordnes seriene i grafen. Merk de kolonnene som skal utgjøre datasettet. I vårt tilfelle kan det være fornuftig å bruke Epoketiden langs x-aksen. Vi får da tegnet grafene som funksjon av tiden, f.eks. temperatur som funksjon av tiden.

	A	B	C	D	E	F	G	H	I	J
1	No	EpochTime	Longitude	Latitude	Temperatur	Water ten	Humidity	Pressure	Illuminance	
2	1	1,34E+09	10,45405	63,42647	21,8	12	46,1	1090	148,8	
3	2	1,34E+09	10,45415	63,42707	21,8	12	45,7	1089,9	149	
4	3	1,34E+09	10,45435	63,42657	21,6	12	46,3	1089,5	149,3	
5	4	1,34E+09	10,45475	63,42637	22,1	12	45,6	1089,6	149	
6	5	1,34E+09	10,45405	63,42697	22,3	12	46,3	1089,4	149,3	
7	6	1,34E+09	10,45465	63,42687	21,6	12	45,5	1089,3	148,9	
8	7	1,34E+09	10,45435	63,42697	22,1	12	45,9	1089,6	148,8	
9	8	1,34E+09	10,45485	63,42657	21,5	12	46,2	1089,8	149	
10	9	1,34E+09	10,45455	63,42637	22,2	12	45,7	1089,8	148,8	
11	10	1,34E+09	10,45485	63,42627	21,7	12	46,3	1089,6	149,1	
12	11	1,34E+09	10,45485	63,42707	21,6	12	45,7	1089,8	148,7	
13	12	1,34E+09	10,45455	63,42667	21,9	12	45,6	1089,9	148,9	
14	13	1,34E+09	10,45415	63,42697	22,1	12	46,2	1090,1	149,3	
15	14	1,34E+09	10,45405	63,42637	21,9	12	45,8	1089,9	148,7	
16	15	1,34E+09	10,45415	63,42667	21,8	12	46,1	1089,9	148,5	
17	16	1,34E+09	10,45475	63,42657	21,5	12	45,5	1089,8	149	
18	17	1,34E+09	10,45455	63,42697	21,6	12	46,1	1089,5	148,8	
19	18	1,34E+09	10,45475	63,42697	22,1	12	45,8	1089,7	148,7	
20	19	1,34E+09	10,45465	63,42697	22,3	12	46	1089,9	149	

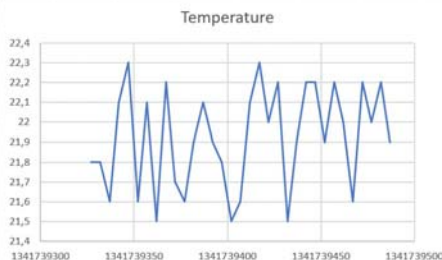
2. Velg type representasjon fra menyen “Sett inn” (Insert) og velg f.eks. punktdiagram med linjer mellom punktene i diagrammet:



3. Velg linjediagram fra nedtrekksmenyen som vist på figuren under til høyre.

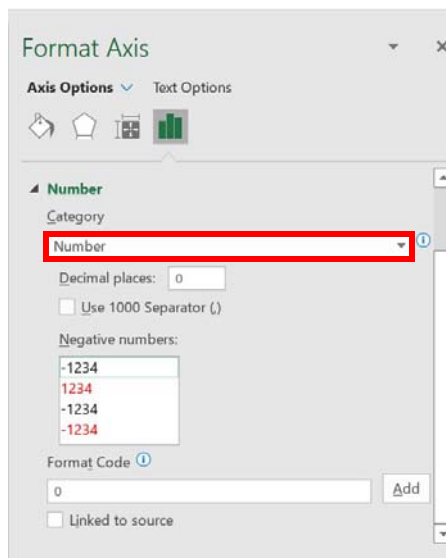


4. Når dette er gjort vil diagrammet tegnes ut som vist på figuren under.

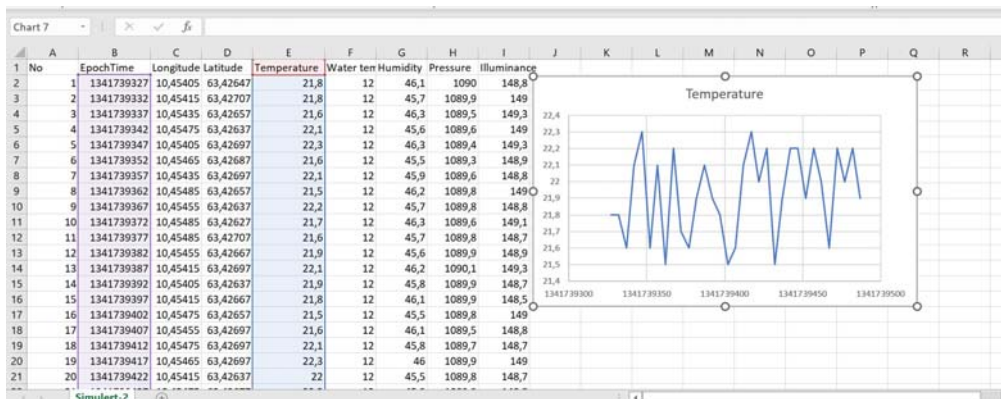




5. Ønsker vi å endre på formatet på aksene, så klikker vi på den akse-benevnningen vi ønsker å endre og vi får opp vinduet vist til høyre. Her kan man f.eks. endre på tallformatet til akse-benevnningen, f.eks. fra vitenskapelig notasjon til vanlig tallnotasjon for tidsangivelsen langs x-aksen.



6. Vi kan dermed ende opp med en presentasjon som vist under:



## 5.4 Bruk av Python for å presentere og analysere dataene<sup>90</sup>

### 5.4.1 Installasjon og oppstart med Python

La oss først se hvordan vi etablerer programmeringsmiljøet med Python. Dette er blant annet hentet fra boka *Python for realfag* [3].

<sup>90</sup>. Dette avsnittet bygger på arbeidet til Thor Inge Hansen ved Skien videregående skole som har utviklet presentasjon og analyseverktøyet.

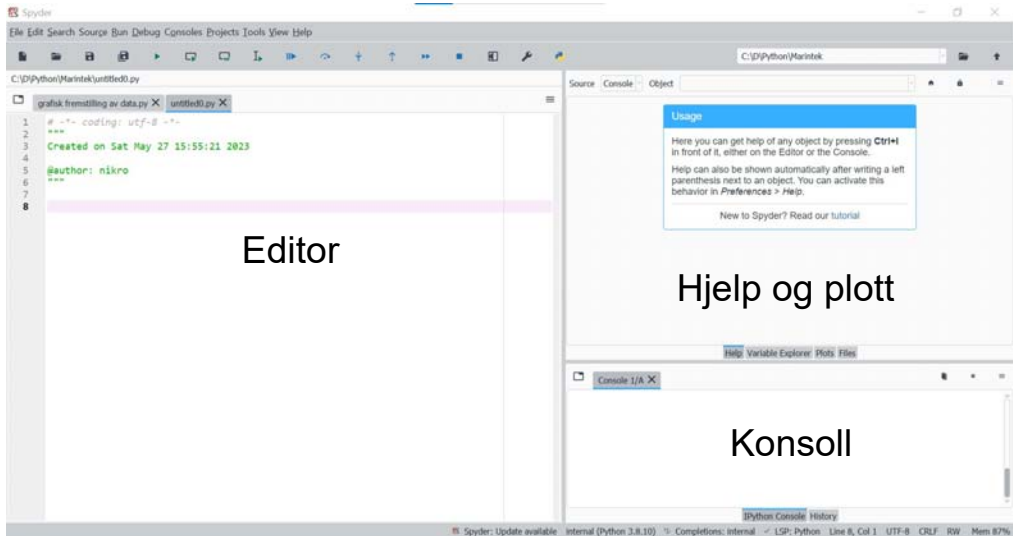


For å installere Python-pakken går man til hjemmesiden til firmaet Anaconda og laster ned og installerer:

<https://www.anaconda.com/download/>

I tillegg til Spyder som er Python editoren, får man med flere bibliotekspakker og andre mer eller mindre nyttige programmer.

Når vi starter Spyder får vi opp en side som ligner på den som er vist i figuren under, med tre felter eller vinduer med ulike funksjoner: Editor, hjelp og plott, og et konsoll som viser responsen som programmet gir, inkludert advarsler og feilmeldinger.



I “Editoren” skriver vi inn kommandoene, evt. programkoden vi ønsker å kjøre. I vinduet “Hjelp og plott” kan vi få tips og hint evt. oppsøke hjelpefunksjoner, vi kan vise innhold i variabler, vise plott og utskrifter eller oversikt over filer. I vinduet “Konsoll” får vi også svar fra programmet, feilmeldinger, advarsler (“warnings”) og kommandohistorien.

I stedet for å liste opp alle mulighetene til Spyder så skal vi gjennomgå et aktuelt progameksempel, men først la oss se litt på et typisk fileformat for dataene våre slik de kan framstå etter at de er lagt ned på filen.

I dette eksempelet legger vi kretsen i dvale mellom hver måling.

### 5.4.2 Organisering av data i filen

La oss ta utgangspunkt i måledata organisert linje for linje i en file. Vi tenker oss at vi ønsker å gjøre målinger til gitte tidspunkter, f.eks. en gang hvert 15. minutt. Når vi først gjør en måling ønsker vi å måle flere ulike parametere samtidig. Det kan være et navn som identifiserer dataene, posisjon der målingene ble gjort bestemt ved hjelp av GPS, tidspunktet, temperatur, spenningen



på batteriet, med mer. Dataene legges etter hverandre på en linje med komma mellom. Når så neste måling skal gjøres, legges de på neste linje slik at det etter hvert dannes en tabell av data som vist i eksempelet under.

```
SeptKursTest_2.txt,1685052745,63.426197,10.453953,7,0.0,20.2,0.0,0.0,0.00,4.02,115088
SeptKursTest_2.txt,1685053136,63.426189,10.453951,3,0.0,20.1,0.0,0.0,0.00,4.01,72369
SeptKursTest_2.txt,1685053538,63.426369,10.454008,4,0.0,20.0,0.0,0.0,0.00,4.00,84543
SeptKursTest_2.txt,1685053955,63.426304,10.454157,7,0.0,19.9,0.0,0.0,0.00,4.05,100712
SeptKursTest_2.txt,1685054342,63.426220,10.453920,4,0.0,19.8,0.0,0.0,0.00,4.08,68519
```

I dette eksempelet slås mikrokontrollen og sensorene helt av mellom målingene, dermed kan vi ikke legge inn en tekstlinje øverst i fila ved programstart. Lengst til venstre ser vi filnavnet som identifiserer måleserien vår, deretter følger epoketiden, bredde- og lengdegrad og antall satellitter. Så følger to åpne felter som vi ennå ikke har tatt i bruk, deretter temperaturmålingen. Så kommer tre tomme verdier før vi møter batterispenningen. Siste verdi på hver linje er antall millisekunder det går fra programmet starter og til dataene er sendt og lagt ned på serveren (denne fila) og spenningen så slås av. Vi ser at sistnevnte variere en god del.

Komma er et greit skilletegn når vi skal bruke Python til å analysere dataene siden desimaltegnet er et punkt. Slike filer kalles CSV-filer (Comma Separated Values).

Det kan også være greit å legge en header over hver kolonne som forteller hvilke data det er snakk om. Det er ikke alltid så lett dersom vi starter det samme programmet om og om igjen for hver måling slik vi gjør.

La oss nå se hvordan vi kan hente utvalgte måledata inn i Python for presentasjon.

### 5.4.3 Lesing og presentasjon av data fra file<sup>91</sup>

La oss se på et grunnleggende eksempel. Eksempelet leser data fra fila vist i forrige avsnitt. Eksempelet burde være enkelt å modifisere og utvide. Programeksempelet finnes i sin helhet i vedlegg F.1 side 267.

Her går vi gjennom programmet del for del slik at det skal være mulig å forstå kommandolinjene og dermed være lettere å gjøre endringer og bygge ut programmet. Noen vil kanskje synes at gjennomgangen er i grundigste laget, men kan være nyttig for de som ikke kjenner Python fra før.

```
import numpy as np
import matplotlib.pyplot as plt
import datetime as dt
import pandas as pd
```

Først importeres diverse biblioteker som brukes i forbindelse med utregningene og presentasjonene mm. `numpy` er et bibliotek som håndterer arrayer av data, `matplotlib.pyplot` del av et plottbibliotek, `datetime` et bibliotek som i dette tilfellet konverterer epoketid til tid og dato og `pandas` er et kraftig bibliotek for dataanalyse.

---

91. Programmet er skrevet av Thor Inge Hansen ved Skien videregående skole



I Python er det dessuten vanlig å lage forkortelser av biblioteksnavnene slik at det skal være lettere å hente ut funksjoner fra bibliotekene. `numpy` forkortes f.eks. `np` (`numpy as np`) og `matplotlib.pyplot` forkortes `plt` (`matplotlib.pyplot as plt`) osv.

```
%% Valg av tidsvindu for plotting av data
vintertid = False
```

De neste to linjene setter variabelen `vintertid` til verdien `False`, siden det når dette programmet ble brukt var sommertid. Vi skal ganske snart bruk denne variabelen i omregningen av *epoketid* til dato og klokkeslett. Programlinjer som begynner med `#` er kommentarer og vil ikke bli prosessert av programmet.

## Henting og strukturering av data fra file på serveren

```
%% import og laging av arrays

#Henter inn fila som en dataframe i pandas
data = pd.read_csv('http://sensor.marin.ntnu.no/logs/SeptKursTest_2.txt',
delimenter=',')
```

Måleverdiene hentes ut av datafila ved hjelp av biblioteksfunksjonen `pd.read_csv` fra pandas-biblioteket (`pd`). Biblioteksfunksjonen har som vi ser, to argumenter. Det første er *nettstedet*: `http://sensor.marin.ntnu.no/logs/` i fila `SeptKursTest_2.txt`. og det andre argumentet angir skilletegnet mellom verdiene, i dette tilfellet er det `“,”`. Verdiene legges inn i dataarrayet `data`.

```
epochTime = data["1685052745"].values
volt = data["4.02"].values
temp = data['20.2'].values
runTime = data['115088'].values/1000 # Gjør ms om til s
```

Det neste som gjøres er å plukke ut verdiene i utvalgte kolonner og legge disse i egne array eller lister som har navnene `epochTime`, `volt`, `temp` og `runTime`. Disse henter ut samtlige verdier i de angitte kolonnene. Siden vi ikke har et kolonne-navn, bruker vi for enkelhets skyld den første dataverdien i hver kolonnen for å angi kolonnen. “`epochTime`” er tidspunktet dataene er hentet inn, her som antall sekunder etter 01.01.1980. “`volt`” er den målte batterispenningen, “`temp`” er den målte temperaturen (senere vanntemperaturen) og “`runTime`” er tiden det tar fra spenningen på mikrokontrolleren slås på til målingen er gjennomført, data er lagt ned på serveren og spenningen på mikrokontrolleren slås av. “`runTime`” er derfor nyttig for å beregne den totale på-tiden og således forholdet mellom på- og av-tiden gjennom måleserien.

## Omregning fra epoketid til dato og tid

```
%% Funksjon for å lage datetime-objeter fra epochTime

def epochToDatetime(x):
    liste=[]
    for i in range(len(x)):
        if vintertid:
            t=dt.datetime.utcnow().timestamp()+ dt.timedelta(hours=1)
```



```
else:
    t=dt.datetime.utcnow().timestamp(x[i]) + dt.timedelta(hours=2)
    liste.append(t)
return liste
```

Så defineres funksjonen `epochToDatetime(x)` med argumentet “x”. “x” er her variabelen eller arrayet med epoketider som vi ønsker gjøre om til “år”, “dato”, “timer”, “minutter” og “sekunder”. I starten av funksjonen lager vi oss en åpen liste (liste), dvs. en liste med åpent antall elementer, som inneholder dato og tid. For å gjøre denne omregningen benyttes biblioteket `datetime (dt.)` og funksjonen `dt.datetime.utcnow().timestamp(x[i])`. Siden dette er UTC (Greenwich Mean Time) tid må vi legge til henholdsvis 1 eller 2 timer avhengig om det er sommer- eller vintertid, til dette bruker vi en if-setning og tester på variabelen vintertid (`if vintertid:`). Variabelen “t” holder enkeltverdier og legges til lista (`liste.append(t)`) etter som enkeltverdiene beregnes i for-loopen (`for i in range(len(x)):`). For å vite antall runder i for-loopen, finner vi lengden av variabelen x (`len(x)`). Tilslutt returneres den komplette lista fra funksjonen.

```
### Anvender funksjonen ovenfor
timelist=epochToDatetime(epochTime)
```

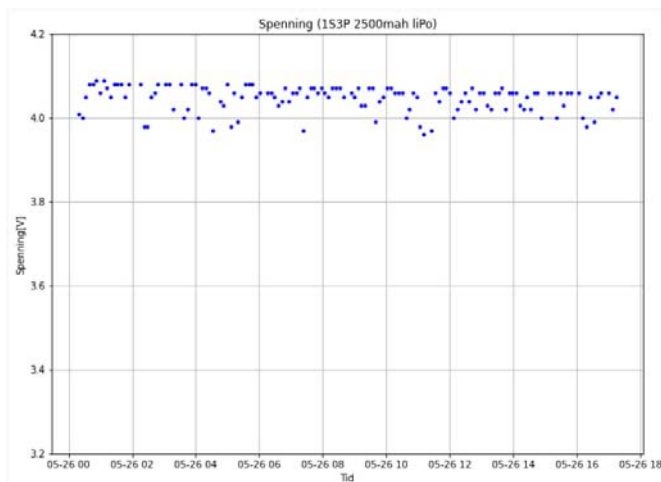
Vi anvender så den nylig definerte funksjonen på arrayet av epoketidsmålinger og får ut en liste over tidspunkter i en hensiktsmessig form.

## Plotting av data

Vi er nå klare for plotting av dataene våre. Vi vil i første omgang lage fire plott.

Først plottes vi spenningsnivået som funksjon av tiden.

Her ser vi hvordan spenningen varierer med ca. 0,1 V fra måling til måling over et tidsintervall på ca. 18 timer den 26. mai 2023.



Programbiten under viser hvordan dette plottet framkommer:

```
plt.figure(1,figsize=(11,8)) # Angir figurnummer og figur størrelse (cm)
plt.plot(timelist,volt,'b.') # Plotter spenning som funk. av tid med blå .
```

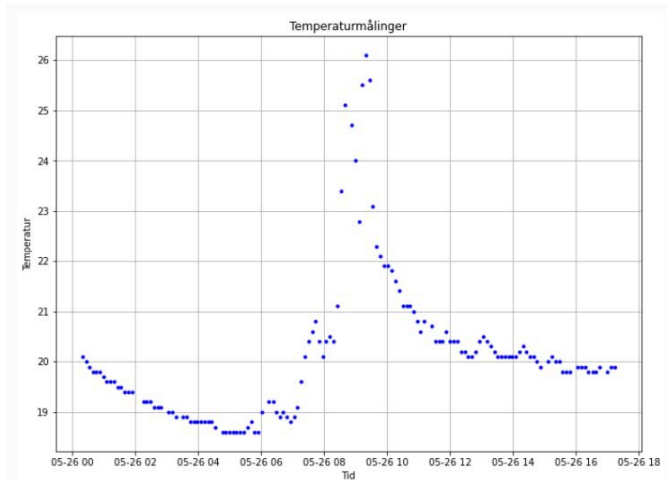


```
plt.ylabel('Spenning[V]') # Angir tekst på den vertikale aksen
plt.xlabel('Tid') # Angir tekst på den horisontale aksen
plt.title('Spenning (1S3P 2500mah liPo)') # Angir figuroverskrift
plt.ylim(3.2,4.2) # Angir skalaen på den vertikale aksen
plt.grid() # Angir at det skal tegnes opp et rutenett
```

I `plt.figure` vil normalt figuren bli angitt i inch, men dersom den generelle måleenheten i programmet er satt til cm er det denne som gjelder. I `plt.plot` vil måleverdiene angis med blå punkter 'b.'.

Dernest plotter vi måleverdiene for temperatur som funksjon av tiden.

Her ser vi hvordan temperaturen endres over et tidsintervall på ca. 18 timer den 26. mai 2023. Sola skinner inn gjennom vinduet fra kl. 07:00 og fram til ca. kl. 09:00.



Her er det ingen overraskelser mht. programmeringen:

```
plt.figure(2, figsize=(11,8))
plt.plot(timelist,temp,'b.')
plt.xlabel('Tid')
plt.ylabel('Temperatur')
plt.title('Temperaturmålinger')
plt.grid()
```





Dersom vi ønsker å plote flere kurver i samme diagram som vist i figuren til høyre, føyer vi til en ekstra linje med `plt.plot()`.

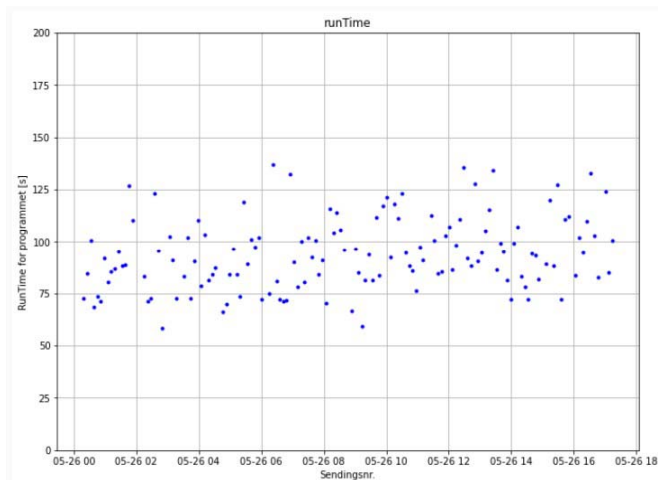
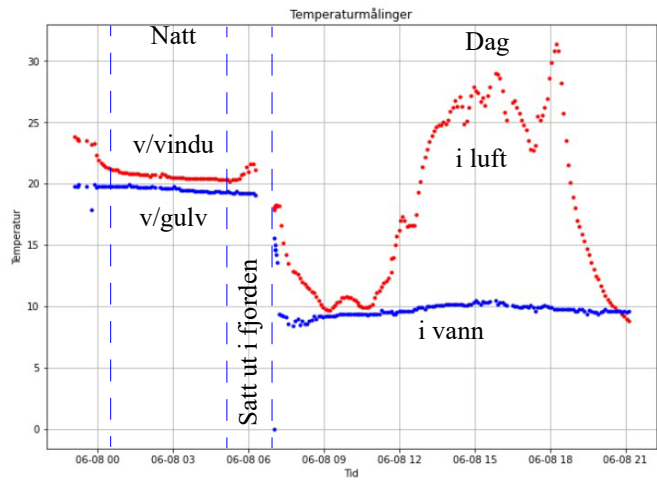
Her har vi plottet temperaturen i vann (blå kurve) og lufttemperaturen (rød kurve).

Vi legger merke til at de to sensorene ikke er helt samstemte der begge er innendørs. Den ene er plassert ved gulvet den andre foran vinduet.

Når de plasseres ut i fjorden så ser vi at temperaturen i vann er ganske stabil mens temperaturen i lufta i instrumentboksen, varierer mye.

Til slutt plotter vi *på*-tiden som funksjon av tiden.

Her ser vi hvordan på-tiden varierer over et tidsintervall på ca. 18 timer den 26. mai 2023. Vi ser at på-tiden varierer fra ca. 60 til 140 sekunder.



Her er heller ingen overraskelser mht. programmeringen:

```
plt.figure(3,figsize=(11,8))
plt.plot(timelist,runTime,'b.')
plt.ylim(0,200)
plt.title('runTime')
plt.xlabel('Sendingsnr.')
plt.ylabel('RunTime for programmet [s]')
plt.grid()
```



## 5.4.4 Analyse av data med Python<sup>92</sup>

I dette avsnittet skal vi se hvordan vi kan gjøre enkle beregninger på måledataene.

### Beregning av middelverdier

I sin enkleste form er en middelverdi summen av alle de aktuelle verdiene delt på antallet. La oss se på et aktuelt eksempel, nemlig middelverdien av avstanden mellom målingene gjort i en måleserie. Vi tar da utgangspunkt i epoketiden nedtegnet for hver måling.

```
%% Regning av gjennomsnittlig intervall

for i in range(len(epochTime)-1):
    interval[i]=int((epochTime[i+1]-epochTime[i]))

interval_mean=np.mean(interval[0:len(interval)-1])
```

Alle epoketidspunktene ligger lagret i arrayet `epochTime`. Funksjonen `len(epochTime)` bestemmer antall elementer i arrayet som utgjør én mer enn antall runder i for-loopen fordi antall intervaller er én mindre enn antall elementer. Derneest beregnes alle intervallene (`epochTime[i+1]-epochTime[i]`) som samles i arrayet: `interval`. Tilslutt beregnes middelverdien av alle elementene i arrayet ved hjelp av funksjonen `np.mean()`. Vi legger merke til at elementene i arrayet som middelverdien beregnes ut fra, er spesifisert som: `[0:len(interval)-1]`, hvilket burde være unødvendig så lenge som alle verdiene er tatt med i beregningen.

*Kommentar:* Dersom målinger uteblir fra måleserien vil den beregnede middelverdien øke i forhold til det nominelle intervallet mellom målingene. Jo flere målinger som uteblir jo større blir avviket fra den nominelle verdien.

### Telling av avvikende målinger

En måte å unngå problemet nevnt under kommentaren over er å forsøke å finne antallet måleintervaller som overskrider en gitt verdi og dermed gir et bedre grunnlag for å beregne en nominell verdi.

```
%% Finner antall ganger intervall mellom sending er over 150 s
count = 1

for i in range(len(interval)):
    if interval[i]>150: #150sek = 2,5 min = feilsending
        count+=1
```

Her telles de gangene intervallet overskrider 150 sekunder (2,5 min). Denne grensen må selvfølgelig settes i henhold til måleintervallet satt i programmet.

*Kommentar:* Med kjennskap til antall manglende målinger vil man kunne korrigere middelverdien og komme nærmere det nominelle intervallet mellom målinger. Imidlertid vil ikke metoden korrigere for to påfølgende manglende målinger. For også å kunne korrigere for slike feil må man telle opp antallet av to påfølgende manglende meldinger, osv.

---

92.Dette er egentlig en gjennomgang av arbeidet gjort av Thor Inge Hansen ved Skien vgs



## Estimat av batteritid

Et nyttig estimat er hvor lenge batteriene vil vare før målestasjonen kobler ned. Dette kan gjøres ut fra en måleserie som går over noe tid. Dersom vi studerer målingene som er vist på figuren på side 145, så vil vi se at spenningen faller jevnt over tid, i tillegg ser vi at spenningen har en del støy. Dersom vi antar en nedre grense for spenningen når den slutter å måle, kan vi ved hjelp av regresjon estimere på hvilket tidspunkt spenningen har nådd ned til denne grensespenningen.

```
### Lineær regresjon med beregning av total tid ned til min_volt
min_volt=3.2

epochLin = np.array(timeStamp).reshape((-1, 1))
model = LinearRegression().fit(epochLin, volt)

a = model.coef_
b = model.intercept_

volt_lin_reg=a*timeStamp+b

epoch_end=(min_volt-b)/a
duration=(int(epoch_end)-int(timeStamp[0]))/(3600*24)
```

Her gjøres en lineær regresjon ut fra innledende målinger gjort over ca. 18 timer. Nedre grense for spenningen der man regner med at utstyret slutter å virke er satt til 3,2 V (`min_volt`).

Det første man gjør er å linearisere måletidspunktene, dvs. gi dem lik avstand (`reshape((-1, 1))`). Dernezt gjøres en lineær regresjon på alle måleverdiene av spenningen innen det tilgjengelige intervallet (`LinearRegression().fit(epochLin, volt)`). Dette resulterer i to parametere (a og b) som er henholdsvis stigningstallet og skjæringspunktet med y-aksen, som gjør det mulig å lage en lineær funksjon som estimerer forløpet av batterispenningen (`volt_lin_reg=a*timeStamp+b`), hvilket gjør det mulig å beregne tidspunktet når spenningskurven når ned til grensespenningen på `min_volt` (3,2V) som her er angitt som *duration*, som oppgis i dager.

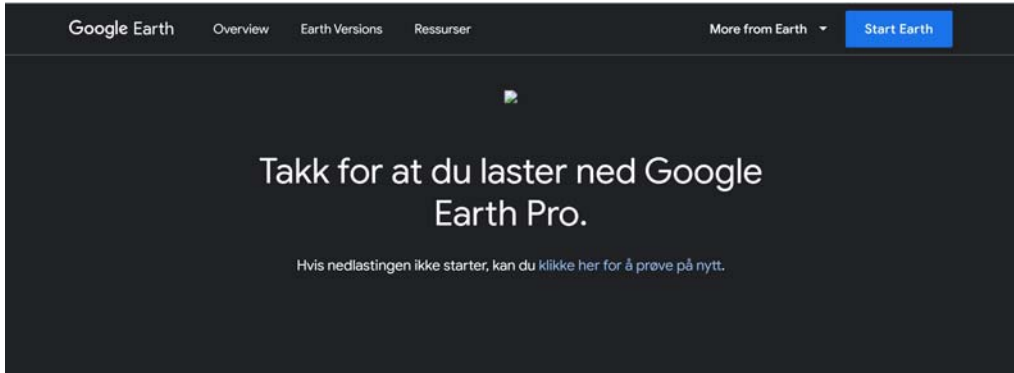
*Kommentar:* Nå er det slett ikke sikkert at batterispenningen avtar som en lineær funksjon mellom full spenning og nedre grense på 3,2V.



## 5.5 Bruk av Google Earth for visning av posisjon fra GPS

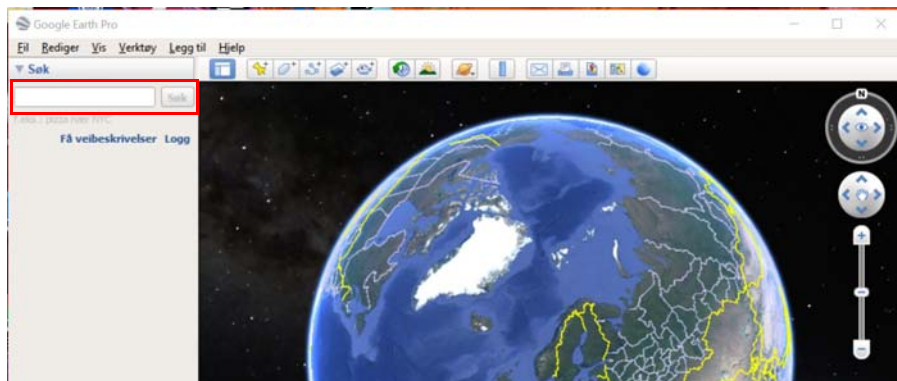
I dette avsnittet skal vi gi en oppskrift for hvordan vi kan vise en posisjon eller en trase i Google Earth.

Google Earth kan lastes ned og installeres fra følgende adresse: <https://www.google.com/earth/versions/download-thank-you/>



### 5.5.1 Plotting av en enkeltposisjon

Når man starter Google Earth får man opp følgende vindu.



Ved å skrive inn bredde- og lengdegrader i innboksen øverst til venstre i vinduet, vil man kunne “forflytte seg” til det angitte stedet på kloden.

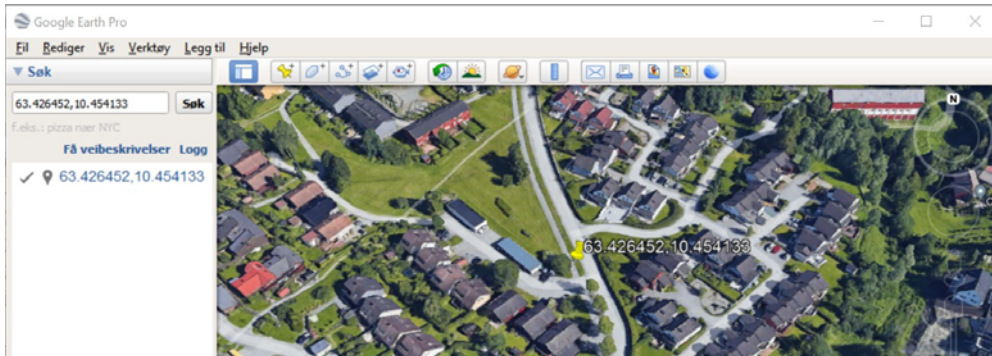
Lengde- og breddegrader legges inn som vist under:

```
63.426452,10.454133  
Breddegrader[°],Lengdegrader[°]
```

Her er selvfølgelig rekkefølgen viktig.



Legg merke til at koordinatene minst bør ha med 4 siffer etter komma, gjerne 5 eller 6, høyde godtas ikke.



### 5.5.2 Plotting av en trase i Google Earth

Dersom man ønsker å plote en trase i Google Earth kan man benytte et programmeringsspråk kalt “Keyhole Markup Language” (KML) som er utviklet for visualisering av to- og tredimensjonale strukturer knyttet til kartdata.

Siden språket er temmelig omfattende og vi kun trenger å bruke en beskjeden del av det, så benytter vi en ferdige programkode og klipper inn våre data for lengde-, breddegrad og høyde. Disse legges inn som en liste med data i kml-koden (se under), før kml-fila lagres med et ønsket navn.

Koordinater og høyde data legges inn som vist under:

```
-112.2550785337791,36.07954952145647,2357  
Lengdegrader[°],Breddegrader [°],Høyde [m]
```

Legg merke til at rekkefølgen på koordinatene er omvendt av det vi la inn i innboksen på forsiden av Google Earth.

Programkode skrevet i kml (legg merke til at et sett med eksempelkoordinater og høyde er klippet inn – rød kode). Her er det lagt inn 11 posisjoner. Du må gjerne legge inn flere eller færre.

```
<?xml version="1.0" encoding="UTF-8"?>  
<kml xmlns="http://www.opengis.net/kml/2.2">  
  <Document>  
    <name>Paths</name>  
    <description>Examples of paths. Note that the tessellate tag is by default  
      set to 0. If you want to create tessellated lines, they must be authored  
      (or edited) directly in KML.</description>  
    <Style id="yellowLineGreenPoly">  
      <LineStyle>  
        <color>7f00ffff</color>  
        <width>4</width>
```



```
</LineStyle>
<PolyStyle>
  <color>7f00ff00</color>
</PolyStyle>
</Style>
<Placemark>
  <name>Absolute Extruded</name>
  <description>Transparent green wall with yellow outlines</description>
  <styleUrl>#yellowLineGreenPoly</styleUrl>
  <LineString>
    <extrude>0</extrude>
    <tessellate>0</tessellate>
    <altitudeMode>absolute</altitudeMode>
    <coordinates>
      -112.2550785337791,36.07954952145647,2357
      -112.2549277039738,36.08117083492122,2357
      -112.2552505069063,36.08260761307279,2357
      -112.2564540158376,36.08395660588506,2357
      -112.2580238976449,36.08511401044813,2357
      -112.2595218489022,36.08584355239394,2357
      -112.2608216347552,36.08612634548589,2357
      -112.262073428656,36.08626019085147,2357
      -112.2633204928495,36.08621519860091,2357
      -112.2644963846444,36.08627897945274,2357
      -112.2656969554589,36.08649599090644,2357
    </coordinates>
  </LineString>
</Placemark>
</Document>
</kml>
```

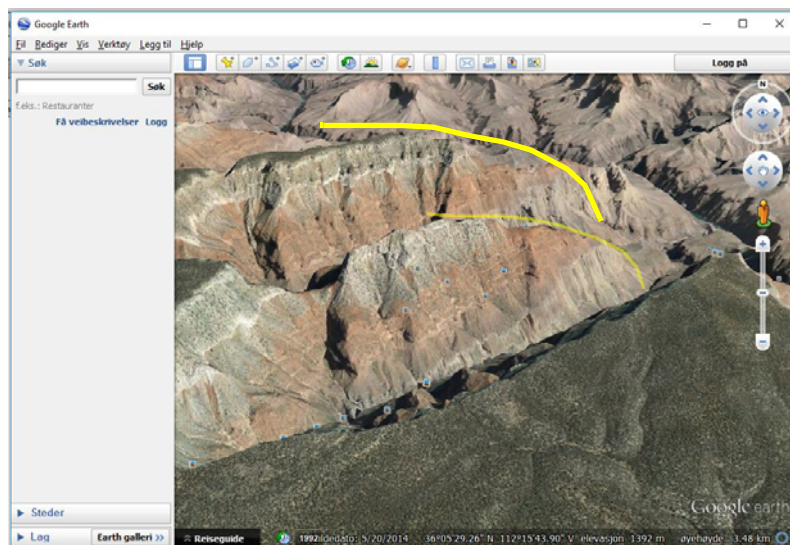
De koordinatene som pr. i dag ligger i eksempelet, angir en helikopterflyvning over Grand Canyon i Colorado, USA.

Eksempeldataene byttes ut med de aktuelle dataene **og kodefila lagres under et ønsket filnavn som ender med .kml**.

Filen hentes opp i Google Earth ved å dobbelklikke på filnavnet. Siden fila ender på .kml, så skal den automatisk bli gjenkjent av Google Earth, starte programmet og laste inn datafilen.



En vil da få tegnet inn traseen som angitt av lista med koordinater og vist på riktig sted på jorda. Eksempelet under viser traseen til helikopteret over Grand Canyon (gul linje).



Dersom man ønsker å vise hvor man har gått en tur, så kan det være upraktisk å måtte vise absolutt høyde. Høydemålinger kan ha store avvik slik at en kan oppleve at traseen går mange meter over eller forsvinner under bakken, til tross for at man hadde begge beina på jorda. I slike situasjoner kan det være greit å bytte ut kommandoen:

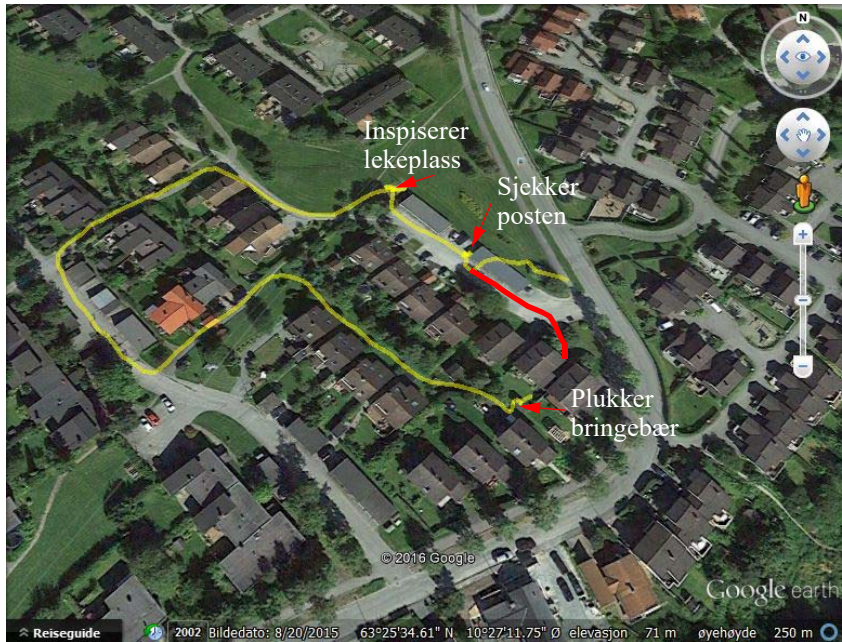
```
<altitudeMode>absolute</altitudeMode>
```

med kommandoen:

```
<altitudeMode>clampToGround</altitudeMode>
```

så vil kurven følge bakken som vist på traseen på bildet under.





Vi legger imidlertid merke til at mottakeren har problemer i starten. I dette området er avviket stort før den tar seg inn. Den røde kurven angir den riktige ruta. Deretter er den svært nøyaktig. Posisjonsdata samples hvert 3. sekund på turen.

### 5.5.3 Editering av kml-fila

Hvilket program skal man så bruke for å legge inn koordinater og høyde? Et nyttig editeringsverktøy til dette formålet er Notepad++. Dette programmet er en avansert teksteditor som ikke gjør noen endringer med filformatet såfremt man ikke ønsker det. Notepad++ kan lastes ned fra:

<https://notepad-plus-plus.org/downloads/>

For å forenkle prosessen med å klippe inn data i kml-koden, er det praktisk å skrive koordinat- og høydedata i det formatet som programmet ønsker: Lengdegrad, breddegrad, høyde. Husk å bruke punktum som desimalpunkt og komma mellom hver verdi. **NB! Det skal ikke være mellomrom etter kommaene.**



## 6 Sensorer

I dette kapittelet skal vi beskrive aktuelle sensorer og hvordan vi kan koble disse til mikrokontrolleren. Men la oss først få litt oversikt over hva som kan være aktuelt å måle av parametere i havet.

### 6.1 Aktuelle målinger i havet langs kysten

**Odd Arne Arnesen** ved Mausund feltstasjon har foreslått en rekke ulike målinger, vi kan blant annet nevne:

- Temperatur
- Bølgehøyde
- Vindstyrke
- Salinitet (saltholdighet)
- Lys
- Plankton sensor (om mulig, Biologisk stasjon ute i Trølla har mye kunnskap om sensorer mm)
- Trykk

Trykk kan være lufttrykk, men om en bøye ankres opp kan måling av vanntrykk danne grunnlag for beregning av dybde, og slik gjøre det mulig å bestemme f.eks. temperatur og saltholdighet som funksjon av dybden.

Men også

- pH
- Oksygeninnhold

... kan være av stor interesse. Surhetsgrad fordi det er av stor betydning for skalldyra som lever i havet, og oksygeninnholdet er viktig for alt liv, ikke minst for oppdrettsfisk.

Også marinbiolog **Jussi Evertsen** ved Vitenskapsmuseet ved NTNU har foreslått undersøkelser, det er verdt å gjøre i havet.

*Han presiserer at temperaturen i havet er en av de viktigste fysiske faktorene i havmiljøet, fordi den påvirker mange fysiske og biologiske egenskaper, og kontrollerer hastigheten til kjemiske reaksjoner og biologiske prosesser.*

*Variasjonen i temperatur og saltholdighet påvirker dessuten tettheten til sjøvannet, som igjen påvirker den vertikale bevegelsen av vann – hva skjer med de kjemiske og biologiske prosessene i vannsøyla fra overflata og ned til havbunnen?*

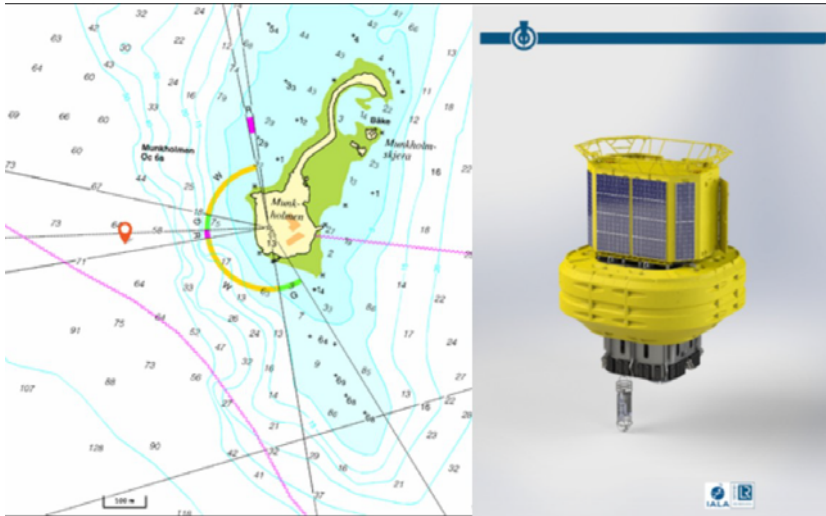
*Temperaturen i havet påvirker også konsentrasjonen av løste gasser i vannet – oksygen og karbondioksid – som er tett forbundet med biologiske aktivitet.*

Så temperaturmålinger er viktigere enn man kanskje tror.

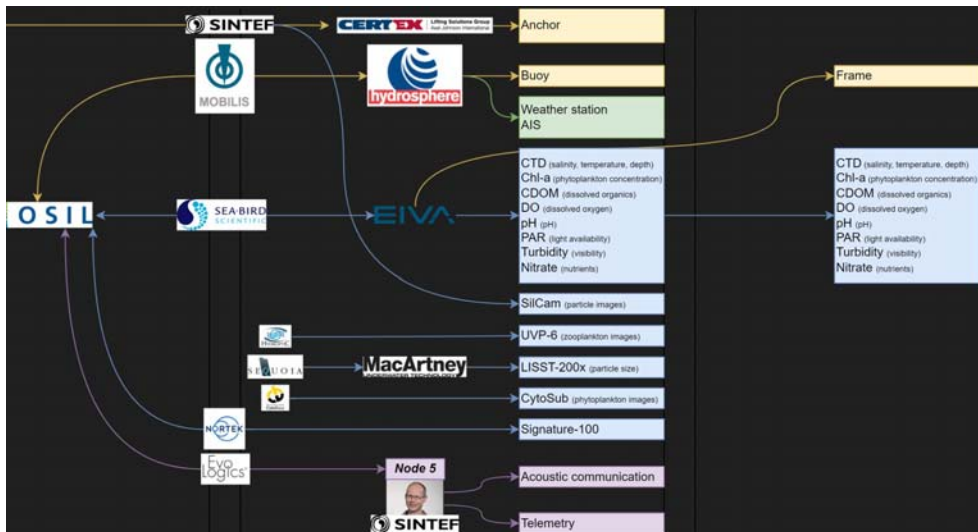


### 6.1.1 Bøya ved Munkholmen

*Emlyn John Davis*<sup>93</sup> er kontaktperson for OceanLab Observatory ved SINTEF/NTNU. De har for tiden utplassert to bøyer i Trondheimsfjorden, en rett vest for Munkholmen, se figuren under, og en lenger ut i fjorden ved Ingdalen. Bøya ved Munkholmen er av denne typen:



Bøya er utstyrt med en rekke sensorer som er levert og betjenes av ulike institusjoner som vist på figuren under<sup>94</sup>. Sensorene som er levert av EIVA er senkbare slik at de kan posisjoneres i ulike dybder.



93. [emlyn.davis@sintef.no](mailto:emlyn.davis@sintef.no)

94. <https://oceanlabobservatory.no/#Equipment>



Som vi ser så er den utstyrt med solceller og en lang rekke sensorer. Her er noen av sensorene den har:

- Værdata
- CTU (Saltholdighet, temperatur og dybde)
- Chl-a (Phytoplankton konsentrasjon)
- CDOM (Oppløst organisk materiale)
- DO (Oppløst oksygen)
- pH (Surhet)
- PAR (Lys-tilgjengelighet)
- Turbiditet (Gjennomsiktighet i vannet)
- Nitrat
- SilCam (Bilder av partikkeltetthet)
- UVP-6 (Bilder av Zooplankton)
- LISST-200X (Partikkel størrelse)
- CytoSub (Bilder av Phytoplankton)
- Akustisk kommunikasjon og telemetri

Selv om dette er profesjonelle sensorer som det er lite aktuelt å utstyr vår bøye med, så kan den være til inspirasjon ved valg av sensorer.

### 6.1.2 Bøya i Vestfjorden

Våren 2022 er det lagt ut en bøye i Vestfjorden nær Lofoten som en hjelp for skipsfart og fiskefartøyer. Måledata er gjort tilgjengelig via Internett.

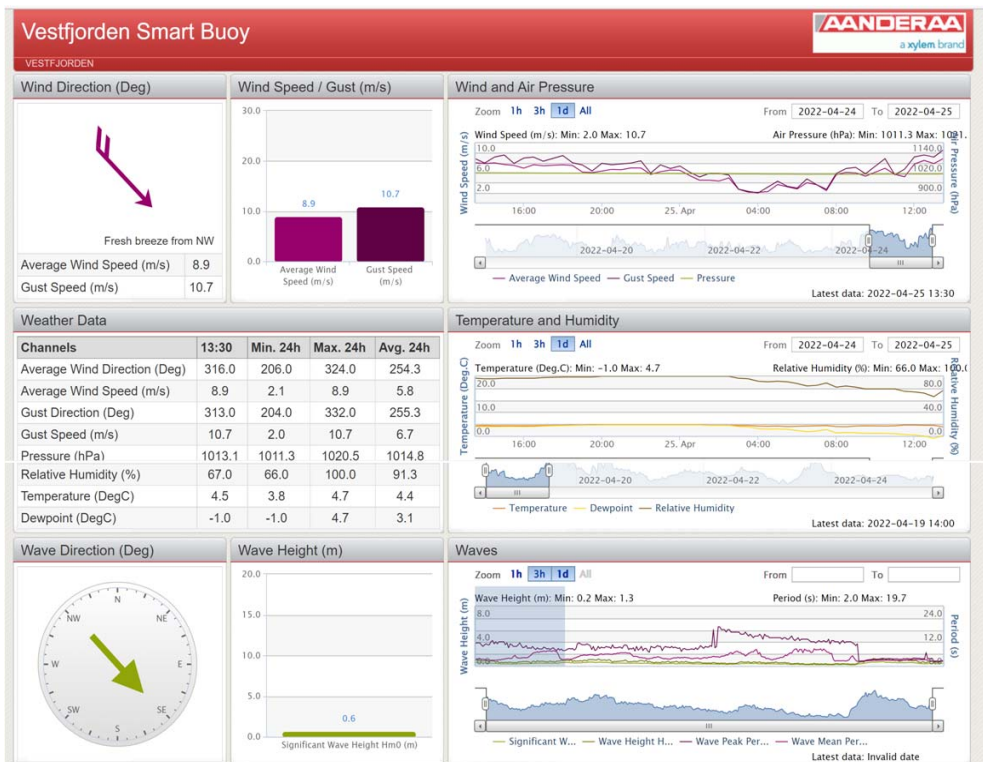


Dersom man går til følgende nettadresse får man opp en side med oversikt over data som oppdateres fortløpende:

[http://s2.data.aanderaa.com/AADI\\_DisplayProgram/setup/HC\\_vestfjorden/default.aspx](http://s2.data.aanderaa.com/AADI_DisplayProgram/setup/HC_vestfjorden/default.aspx)



Figuren under viser et utsnitt av målinger.



### 6.1.3 Meteorologisk institutt

Institutt for marin teknikk ved Førsteamanuensis **Håvard Holm** har også vært i kontakt med Meteorologisk institutt som kan være interessert i værddata, som f.eks. bølgehøyde, vindstyrke, vindretning og nedbør i tillegg til temperatur. Ikke alle de nevnte er like lett å måle, men kan være større eller mindre utfordringer for elevene. Den vil fordype seg i måling av værddata kan ta en titt i heftet: *Videregående opplæring, Arduino: Værstasjon* [8].

Det påpekes at det er viktig for resultatet at sensorene har god kvalitet. Man kan imidlertid se for seg flere kategorier sensorer:

1. Profesjonelle sensorer med høy kvalitet
2. Kommersielt tilgjengelige sensorer som ikke koster så mye, men med rimelig god kvalitet
3. Egenutviklede sensorer

Alle disse sensorene burde kunne brukes evt. prøves ut bare man er klar over de begrensningene sensorene innen hver kategori har. Man kan f.eks. bruke profesjonelle sensorer for å bedømme kvaliteten av kommersielle og egenutviklede sensorer. Kalibrering er også viktig.

Vi skal senere komme tilbake til mulige sensorløsninger, se kapittel 6 side 155.



## 6.2 Organisering av oppkobling

Vi har tidligere beskrevet shield-kort som inkluderer sensorer og kort som egner seg til å koble opp elektronikk knyttet til sensorer.

Vi har tidligere sett hvordan enkelte shield-kort okkuperer enkelte porter på kortet. Denne oversikten kan det være greit å ha nå når vi skal velge porter for ytterligere sensorer.

Tabell 2: Oversikt over bruk av mikrokontrollerens porter

Port #	Port funksjon	Shield	Port #	Port funksjon	Shield
1	AREF		15	D6~	ENV DRDY Temp, Hum
2	A0		16	D7~	ENV DRDY, Air pressure
3	A1		17	D8~	MOSI - SD-kort
4	A2	ENV LYS	18	D9~	SCK - SD-kort
5	A3	Batterinivå	19	D10~	MISO - SD-kort
6	A4		20	D11~	ENV SDA I <sup>2</sup> C GPS SDA I <sup>2</sup> C
7	A5		21	D12~	ENV SCL I <sup>2</sup> C GPS SDA I <sup>2</sup> C
8	A6		22	D13~	
9	D0~	DS18B20 VannTemp	23	D14~	
10	D1~	MKR restart command	24	RESET	Reset from WD
11	D2~	WD disable during sleep	25	GND	
12	D3~	Interrupt fra ext. RTC	26	+3,3V	
13	D4~	ENV SD CS	27	Vin	
14	D5~	pinWDReset	28	+5V	

Vi velger f.eks. å allokere port A0 til analog måling av temperatur ved hjelp av en NTC-motstand.

Tabellen under gir en oversikt over noen aktuelle sensorer:

Betegnelse	Type	Nettadresse	Pris
Leverandør: RS-online			
NTC 10kΩ	Temperatur, 10 stk.	<a href="https://no.rs-online.com/web/p/thermistor-ics/1793885/">https://no.rs-online.com/web/p/thermistor-ics/1793885/</a>	NOK 32,02
DS18B20	Temperatur u/vann	<a href="https://no.rs-online.com/web/p/sensor-development-tools/2049893/">https://no.rs-online.com/web/p/sensor-development-tools/2049893/</a>	NOK 81,35
Leverandør: RFRobot			
SEN0237-A	Oppløst oksygen	<a href="https://www.dfrobot.com/product-1628.html">https://www.dfrobot.com/product-1628.html</a>	\$ 169,00
DFR0300-H	Konduktivitet (K=10)	<a href="https://www.dfrobot.com/product-1797.html">https://www.dfrobot.com/product-1797.html</a>	\$ 79,90
SEN-0161	pH sensor	<a href="https://www.dfrobot.com/product-1025.html">https://www.dfrobot.com/product-1025.html</a>	\$ 29,50
SEN-0189	Turbidimeter	<a href="https://www.dfrobot.com/product-1394.html">https://www.dfrobot.com/product-1394.html</a>	\$ 9,90
SEN-0165	ORP-sensor	<a href="https://www.dfrobot.com/product-1071.html">https://www.dfrobot.com/product-1071.html</a>	\$ 89,00
<b>Totalt</b>	<b>Kurs =10,63 (13.10.22)</b>		<b>\$377,30</b>





## 6.3 Temperaturmåling

Det er mange måter å måle temperatur på. Her skal vi ta for oss måling med NTC-motstand som er en billig om enn noe omstendelig sensor å bruke, senere skal vi også bruke DS18B20 som egner seg spesielt godt for måling i vann.

### 6.3.1 Temperaturfølsom motstand (NTC- og PTC-motstander)

Metaller vil normalt ha økende resistans med økende temperatur. I et halvledermateriale vil flere ladningsbærere løftes opp i ledningsbåndet slik at ledningsevnen til halvledermaterialet øker, dvs. at resistansen blir mindre.

De fleste motstandsmaterialer endrer resistans som funksjon av temperaturen. Som regel er dette uønsket, men i noen spesielle tilfeller ønsker man nettopp en slik endring og utformer komponenten og materialet deretter. Slike motstander brukes også i forbindelse med måling eller deteksjon av temperaturendringer, eller til å kompensere for uønsket temperaturdrift i elektronisk utstyr.

- NTC – *Negative Temperatur Coefficient*, dvs. at resistansen avtar med økende temperatur.
- PTC – *Positive Temperatur Coefficient*, dvs. at resistansen øker med økende temperatur.

### 6.3.2 NTC-motstanden

NTC-motstander er laget av et materiale hvis resistivitet varierer sterkt med temperaturen. Som navnet sier (*Negative Temperature Coefficient* – NTC) så avtar resistansen med økende temperatur.

NTC-motstander er derfor vanligvis bygget opp som en polykrystalinsk *halvleder* som kan bestå av en blanding av krom, mangan, jern, kobolt og nikkel, som sintres<sup>95</sup> sammen med et plastisk bindemiddel.

En forenklet sammenheng mellom resistansen ( $R_{NTC}$ ) og temperaturen ( $T$ ) kan uttrykkes som:

$$R_{NTC} = Ae^{B/T} \quad (6.1)$$

hvor  $A$  og  $B$  er *tilnærmet konstante* innen begrensede temperaturområder.

I datablader for NTC-motstander oppgis gjerne resistansen ( $R_r$ ) for en referansetemperatur ( $T_r$ ). I et temperaturområde nær referansetemperaturen antas  $B$ -verdien å være tilnærmet konstant (eksempelvis så angir  $B_{25/85}$  at  $B$ -verdien er tilnærmet konstant innen området  $25^\circ\text{C}$  til  $85^\circ\text{C}$ ).

Vi kan da sette opp følgende to ligninger:

$$R_{NTC} = Ae^{\frac{B_{25/85}}{T}} \quad (6.2)$$

---

<sup>95</sup>. Sintring betyr at metallpulver knyttes sammen ved hjelp av oppvarming, men uten å smelte.





$$R_r = A e^{\frac{B_{25/85}}{T_r}} \quad (6.3)$$

Ved å eliminere  $A$  fra disse uttrykkene, kommer vi fram til følgende sammenheng, løst med hensyn til resistansen  $R_{NTC}$  i NTC-motstanden:

$$R_{NTC} = R_r \cdot e^{\left(\frac{B_{25/85}}{T} - \frac{B_{25/85}}{T_r}\right)} \quad (6.4)$$

Dette uttrykket går under betegnelsen *Beta-formelen*.

Når vi skal beregne verdien for en NTC-motstand ved en gitt temperatur, slår vi opp  $B$ -verdien,  $R_r$  og  $T_r$  i databladet, sørger for at de aktuelle temperaturene ligger innenfor området til  $B$ -verdien, og beregner  $R$  ved å sette inn ønsket temperatur  $T$  hvor temperaturen angis i grader Kelvin.

Siden  $B$  ofte er angitt for området  $25^\circ - 85^\circ\text{C}$ , havner man lett på siden av det spesifiserte området, siden vi ofte ønsker å måle i området  $0^\circ - 25^\circ\text{C}$ . Siden vi lineariserer og kalibrerer sensoren trenger det ikke å bety så mye for vår anvendelse.

#### NTCLE400E3103H – NTC Thermistor 10kΩ

Fra databladet<sup>96</sup> for *NTCLE400E3103H* finner vi følgende:  $R_{25}$  er referansemotstand ( $R_r$ ) ved  $25^\circ\text{C}$  ( $T_r = 298\text{ K}$ ):

ELECTRICAL DATA AND ORDERING INFORMATION				SAP MATERIAL AND ORDERING NUMBER <sup>(1)(2)</sup>		
$R_{25}$ (Ω)	$R_{25}$ -TOL. (± %)	$B_{25/85}$ (K)	$B_{25/85}$ -TOL. (± %)	EPOXY TYPE	SLEEVED TYPE	PIPE TYPE
2200	3	3977	0.75	NTCLE400E3222H	NTCLS100E3222H	NTCLP100E3222H
4700	3	3977	0.75	NTCLE400E3472H	NTCLS100E3472H	NTCLP100E3472H
5000	3	3977	0.75	NTCLE400E3502H	NTCLS100E3502H	NTCLP100E3502H
10 000	3	3977	0.75	NTCLE400E3103H	NTCLS100E3103H	NTCLP100E3103H
47 000	3	4090	1.5	NTCLE400E3473H	NTCLS100E3473H	NTCLP100E3473H
100 000	3	4190	1.5	NTCLE400E3104H	NTCLS100E3104H	NTCLP100E3104H

Figur 6.1 Datablad for NTC-motstand *NTCLE400E3103H*, 2,2 – 100 kΩ (Epoxy type)

Med disse dataene kan vi skrive:

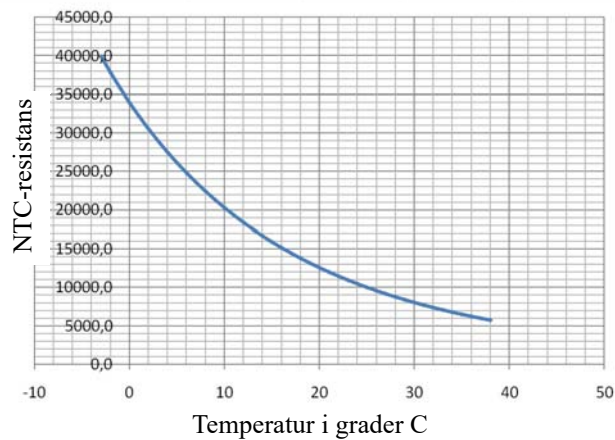
$$R_{NTC} = 10\text{k} \cdot e^{\left(\frac{3977}{T} - \frac{3977}{298}\right)} \quad (6.5)$$

hvor  $B_{25/85} = 3977$  (NTCLE400E3103H – 10 kΩ) og referansetemperaturen  $T_r = 298\text{ K}$ .

96. Databladet er hentet fra: <https://docs.rs-online.com/6544/0900766b8167e65c.pdf>

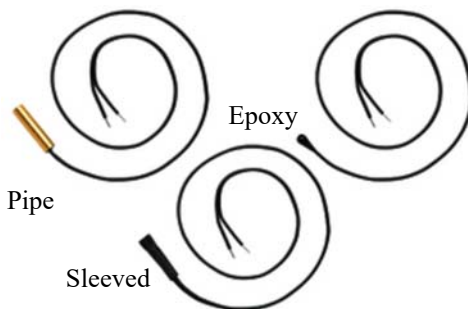


Dersom vi beregner verdier for  $R_{NTC}$  i temperaturområdet  $25^{\circ} - 85^{\circ}\text{C}$ , får vi følgende graf:



Figur 6.2 NTC motstand som funksjon av temperaturen for NTCLE400E3103H – 10 kΩ

En annen viktig parameter for NTC-motstander, er hvor raskt resistansen endrer seg ved sprang i temperaturen. Denne parameteren betegnes *NTC-motstandens tidskonstant* ( $\tau$ ), og angir den tiden det tar for resistansen og endre seg til 63,2% av den nye resistansen etter at temperaturen har endret seg 1 K (Kelvin) over omgivelsestemperaturen. En antar at temperaturendringen ikke er forårsaket av indre oppvarming på grunn av elektrisk strøm som flyter gjennom NTC-motstanden.



Response time <sup>(1)</sup> :		
NTCLE400...Epoxy	≈ 7	s
NTCLS100...Sleeve	≈ 15	
NTCLP100...Pipe	≈ 10	

I databladet finner vi, som forventet, at temperaturrensen er avhengig av innkapslingen av sensoren. Den valgte typen sensor leveres i tre typer innpakning som vist på figuren til venstre – innstøpt i epoxy, omsluttet av en mansjett (sleeved) eller montert i et lite metallrør (pipe).

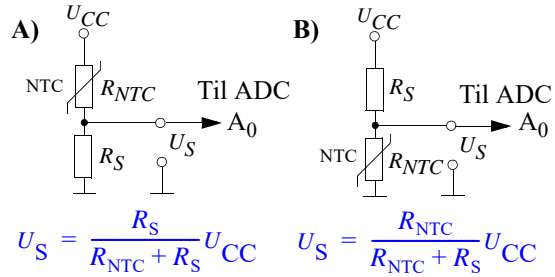
Vi registrerer at tidskonstanten for Epoxy-varianten er 7 sek.

Vi har valgt denne varianten fordi den har en relativt lang tilførselsledning på 400 mm og fordi sensoren er pakket inn i epoxy og dermed sannsynligvis er godt beskyttet for inntrengning av vann.



## Oppkobling til ADC

Siden grensesnittet til mikrokontrolleren krever en spenning, kobles NTC-motstanden i serie med en motstand ( $R_S$ ) som vist i figuren til høyre, i en såkalt *spenningsdeler*. Dersom vi velger verdien til seriemotstanden lik referanseverdien til NTC-motstanden ( $R_{25}$ ), så viser det seg at sammenhengen mellom temperatur og spenning blir relativt lineært i området rundt referansetemperaturen ( $T_r$ ). Spenningsnivået,  $U_S$ , beregnes fra formlene som antydte på figuren til høyre. Legg merke til at oppkoblingen på tegning A gir økende spenning,  $U_S$ , med økende temperatur, mens oppkoblingen i tegning B gir fallende spenning,  $U_S$ , med økende temperatur.



*Spenningsdeler for konvertere variasjon i resistans til variasjon i spenning.*

I vårt eksempel velger vi å plassere NTC-motstanden nærmest  $U_{CC}$  som vist i figur A over. Vi får da en økende spenning som funksjon av økende temperatur.

## Optimal seriemotstand

Vi har registrert at motstandsverdien til NTC-motstanden er svært ulineær som funksjon av temperaturen. Nå er det ikke motstanden vi måler, men spenningen på utgangen av spenningsdeleren, så la oss se hvordan spenningen avhenger av temperaturen og hvordan lineariteten varierer med verdien til seriemotstanden.

På bakgrunn av ligningene foran kan vi sette opp et uttrykk for temperaturen som funksjon av spenningen som evt. kan legges inn i programmet i mikrokontrolleren.

Vi setter:

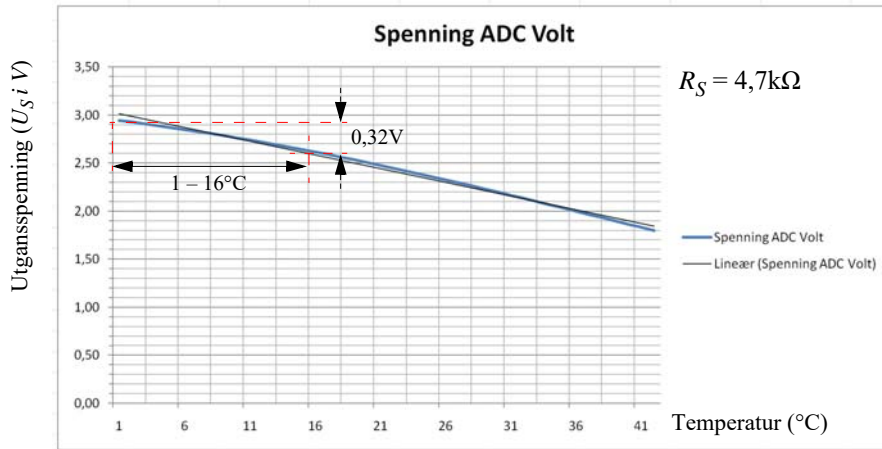
$$R_{NTC} = 10k \cdot e^{\left(\frac{3977}{T} - \frac{3977}{298}\right)} \quad (6.6)$$

inn i ligningen:

$$U_S = \frac{R_{NTC}}{R_{NTC} + R_S} U_{CC} \quad (6.7)$$



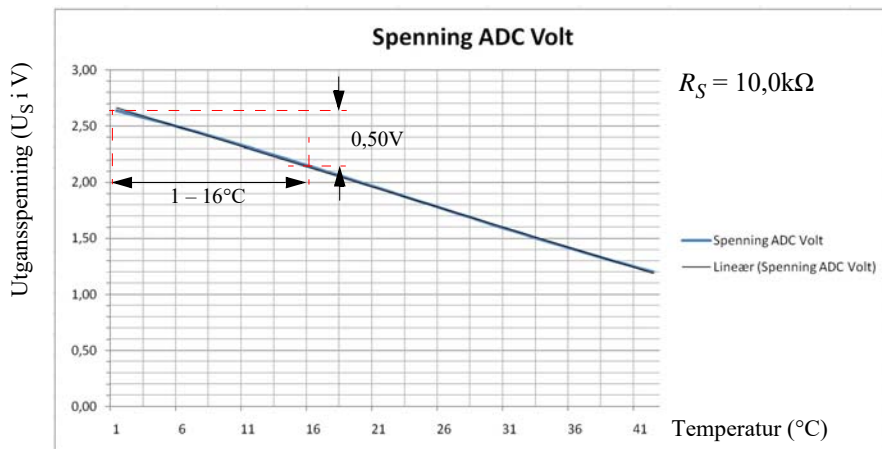
Dermed kan vi beregne  $U_S$  som funksjon av temperaturen for ulike verdier av seriemotstanden,  $R_S$ . I figuren under har vi modellert spenningen  $U_S$  som funksjon av temperaturen i området 0 – 42°C med en seriemotstand på  $R_S = 4,7 \text{ k}\Omega$ :



Sammen med spenningskurven har vi lagt inn den best tilpassede lineære sammenhengen (tynn rett linje). I temperaturområdet 1 – 42°C er spenningssvinget lik  $U_{diff} = 0,82 \text{ V}$ .

Selv om kurven buer noe så er den relativt lineær i området 1 – 16°C som er det temperaturområdet som sannsynligvis er mest aktuelt ved måling i havet langs kysten. Spenningsvariasjonen over det nevnte temperaturområde blir ca. 0,32V, dvs. 21,3 mV pr.  $^{\circ}\text{C}$ .

Dersom vi velger en seriemotstand  $R_S = 10 \text{ k}\Omega$  blir kurven mer lineær over hele området fra 1 – 42°C.



Vi registrerer at avvikene i endene av området er mindre med en seriemotstand på 10k $\Omega$  enn med 4,7k $\Omega$ . Dessuten er spenningssvinget i området 1 – 42°C lik  $U_{diff} = 1,08 \text{ V}$ . I temperaturområdet 1 – 16°C er spenningssvinget på ca. 0,50V, dvs. 33,3 mV pr.  $^{\circ}\text{C}$ .

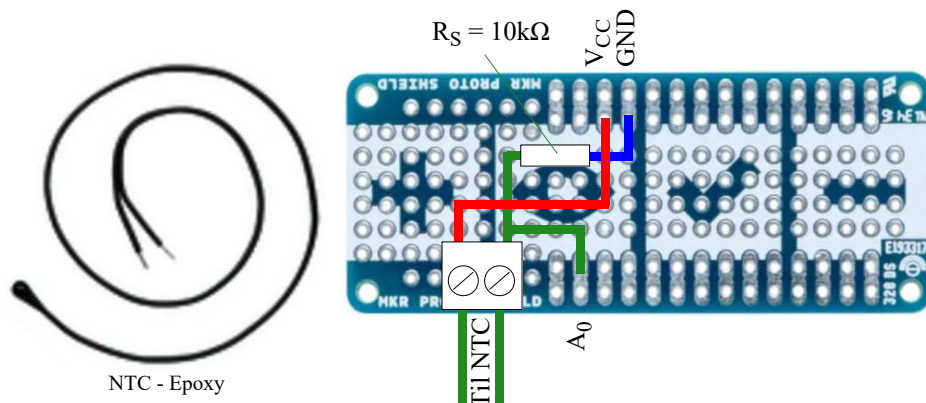


Vi ser at  $10\text{k}\Omega$  gjør at kurven blir mer lineær og gir et større spenningsving, og som dermed utnytter det dynamiske området til ADC'en bedre. Likevel ser vi at spenningsvinget er svært beskjedent.

Fra avsnitt 4.4.1 side 92 husker vi at MKR NB 1500 har inntil 12 bit oppløsning og at forsyningsspenningen og sannsynligvis referansespenningen til AD-konverteren, er  $3,3\text{V}$ . Det betyr at hvert bit tilsvarer et spenningsving på  $3,3\text{V}/4096 = 0,8\text{ mV}$  dvs. ca. 42 bitverdier pr.  $^{\circ}\text{C}$  som gir en teoretisk oppløsning på  $0,024\text{ }^{\circ}\text{C}$ . Dermed vil usikkerhet og støy i målingene være dominerende og det vil være mye å hente på midling.

## Oppkobling

Før vi kan utføre målinger må vi koble opp sensoren. Det kan gjøre vi på et prototyp kort eller et kobling Brett for uttesting.



Sensor-kabelen er  $400\text{ mm}$ , men kan skjøtes. Man må imidlertid være klar over at en skjot lett kan lekke sjøvann, hvilket vil påvirke målingene sterkt. Sørg derfor at skjøten ligger tørt og godt over vannlinjen eller er forsvarlig sikret mot lekkasje.

Det neste vi må gjøre er å kalibrere sensoren.

## Kalibrering av temperatursensoren

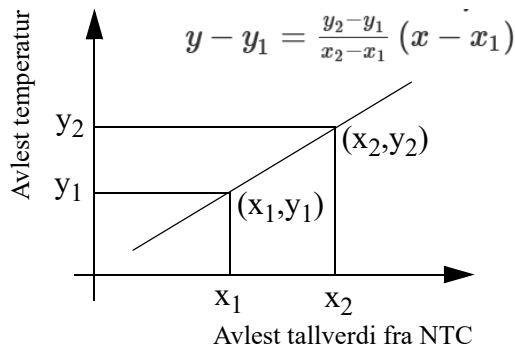
I fortsettelsen antar vi at det er en omtrent lineær sammenheng mellom måleverdi og temperaturer i det aktuelle temperaturområdet. Dette er ikke langt fra sannheten, i et avgrenset område (f.eks.  $1 - 16^{\circ}\text{C}$ ), dersom man velger motstandsverdien i seriemotstanden lik referansemotstanden.

Under denne betingelsen kan følgende metode benyttes med tilstrekkelig nøyaktighet:

1. Fyll et begerglass med vann. Bruk vann med to ulike temperaturer, for eksempel  $5^{\circ}\text{C}$  og  $15^{\circ}\text{C}$ . Sett gjerne et stort glass med vann i kjøleskapet og bruk dette når vannet er avkjølt til å blande vann med ulike temperaturer.
2. Fyll et begerglas med kaldt vann fra kjøleskapet og stikk sensoren ned i vannet. Vent 1 minutt før du foretar målingen.



3. Bruk et kalibrert termometer og mål “virkelig” temperatur i det kalde vannet ( $y_1$ ), les også av tallverdien, gjerne den digitale verdien levert fra AD-konverteren (A0), som avleses spenningsdeleren som inneholder NTC-motstanden ( $x_1$ ). Dette må gjerne være den digitale tallverdien som leveres fra AD-konverteren.
4. Bland så ut det kalde vannet med litt varmt vann slik at du får en vanntemperatur på ca.  $16^\circ\text{C}$ . Stikk så sensoren og termometeret ned i begerglasset og vent ca. 1 min.
5. Mål “virkelig” temperatur med termometeret ( $y_2$ ) og les av tallverdien fra AD-konverteren ( $x_2$ ).



Figuren over viser hvordan topunktsformelen kan brukes for å finne et uttrykk for sammenhengen mellom målte verdier og temperatur.

Topunktsformelen for lineære ligninger kan også skrives slik:

$$y = ((y_2 - y_1)/(x_2 - x_1)) * (x - x_1) + y_1 \quad (6.8)$$

Hvis  $k = (y_2 - y_1)/(x_2 - x_1)$  (stigningskoeffisienten), kan vi skrive ligningen slik:

$$y = k(x - x_1) + y_1 \quad (6.9)$$

Sett verdiene inn i formelen over og finn et uttrykk for  $y$  (temperatur i  $^\circ\text{C}$ ) som funksjon av tallverdien hentet fra AD-konverteren som måler spenningen fra NTC-motstanden ( $x$ ).

I vedlegg I.5 finner du et program som kan brukes for å avlese måleverdiene fra AD-konverteren.



### 6.3.3 Bruk av DS18B20

DS18B20 er en digital temperatursensor levert av MAXIM IC. Den leveres i minst to modeller som vist på figuren til høyre. Modellen til høyre er vanntett og egner seg til vårt bruk.

Sensoren leverer data på en entråds-buss merket *Data* eller *DQ* på figuren. I tillegg trengs spenning ( $V_{DD}$ ) og jord ( $GND$ ). Sensoren kan operere med power spenninger fra 3,0 – 5,5V.

Sensorer har et måleområde på  $-55$  til  $125^{\circ}\text{C}$  med en oppløsning på fra 9 – 12 bit.

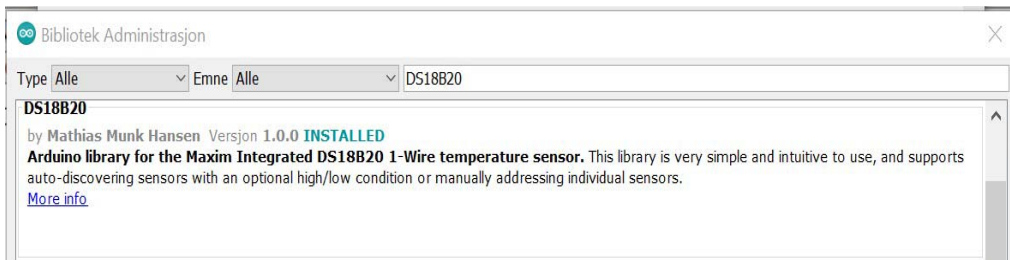
Hver sensor har et 64 bits unikt serienummer slik at flere kan kobles til bussen og adresseres individuelt.



#### Installasjon av biblioteker

For å kunne bruke DS18B20 må vi installere to biblioteker: Ett for bruk av entrådbuss (oneWire.h) og en for sensoren (DS18B20.h).

**DS18B20:** Gå til: *Skisse* > *Inkluder bibliotek* > *Administrer bibliotek* og skriv DS18B20 i søkefeltet og velg biblioteket DS18B20 skrevet av **Mathias Munk Hansen**:



**OneWire:** Gå til: *Skisse* > *Inkluder bibliotek* > *Administrer bibliotek* og skriv OneWire i søkefeltet og velg biblioteket OneWire skrevet av **Paul Stoffregen**:



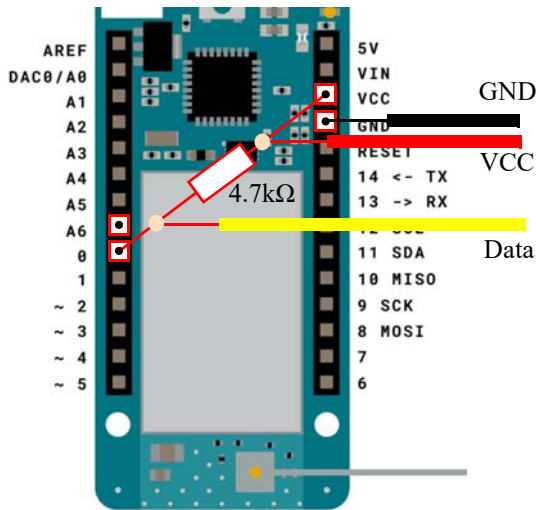




## Enkel oppkobling

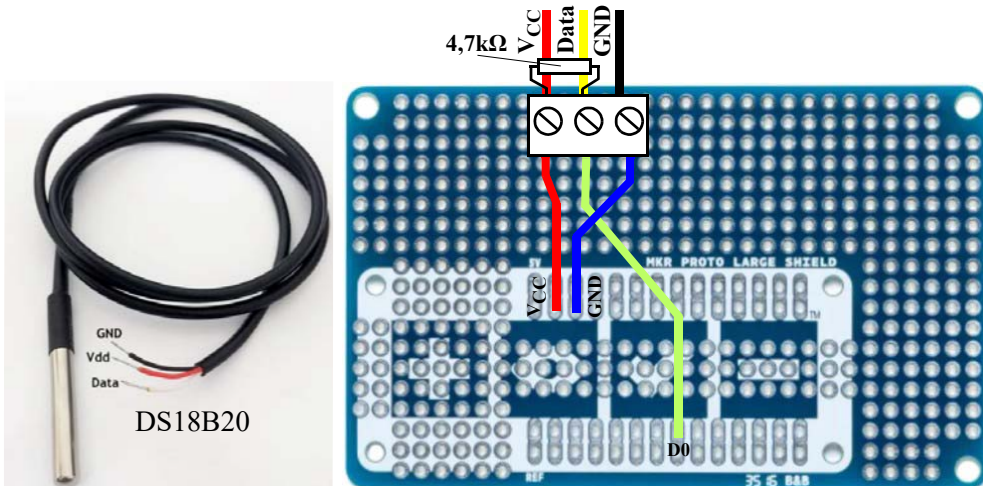
Ved hjelp av en stiftlist kan vi lage oss en enkel tilkobling som vist på tegningen under. Vi kobler oss da til stiftene som settes rett ned i hylsekontaktene på mikrokontrollerkortet eller shield-kortet om vi har montert et

slikt. Vi har plassert stiftene i D0 og A6, og VCC og GND, to på hver side av mikrokontrollerkortet. Vi legger merke til at det skal være en motstand på  $4,7\text{k}\Omega$  mellom Data utgangen og VCC (3,3V). I tillegg skal jordledningen loddes til GND.



## Oppkobling med koblingsplint

Vi velger å bruke en koblingsplint for tilkobling av DS18B20. Signalet, den gule ledningen er koblet til D0.





Prototypkortet skal plasseres under mikrokontrollerkortet MKR NB 1500 som vist på figuren under. PASS PÅ å plasser kortet riktig vei. Teksten på hylsekontaktene viser at det rett plassert.



### Testprogrammet (Testprogram-DS18B20-Temp-1)

Testprogrammet er ganske enkelt der man nøyer seg med å sette opp og lese av sensoren. Programmet er gjengitt i vedlegg I.6.

```
/*  
 * Programmet setter øvre og nedre alarmtemperatur og henter  
temperaturen  
 * og skriver ut denne. Nils Kr. Rossing 04.07.22  
 */
```

```
#include <MKRNB.h>  
#include <DS18B20.h>  
#include <OneWire.h>
```

```
DS18B20 ds(0); //Sensoren er koblet til pinne D0  
uint8_t address[] = {40, 250, 31, 218, 4, 0, 0, 52};  
uint8_t selected;
```



```
void setup()
{
  Serial.begin(115200);
  selected = ds.select(address);

  Serial.println("Testprogram-DS18B20-Temp-1");
}

void loop()
{
  Serial.print("Temperature is: ");
  Serial.print(ds.getTempC());
  Serial.println(" C");

  delay(2000);
}
```

#### 6.4 Sensor for måling av oppløst oksygen i vann (SEN0237-A)

Sensoren måler oppløst oksygen i vann som er viktig for livet i havet. Sensoren er tilpasset bruk sammen med Arduino og leveres av firmaet DFRobot<sup>97</sup>.

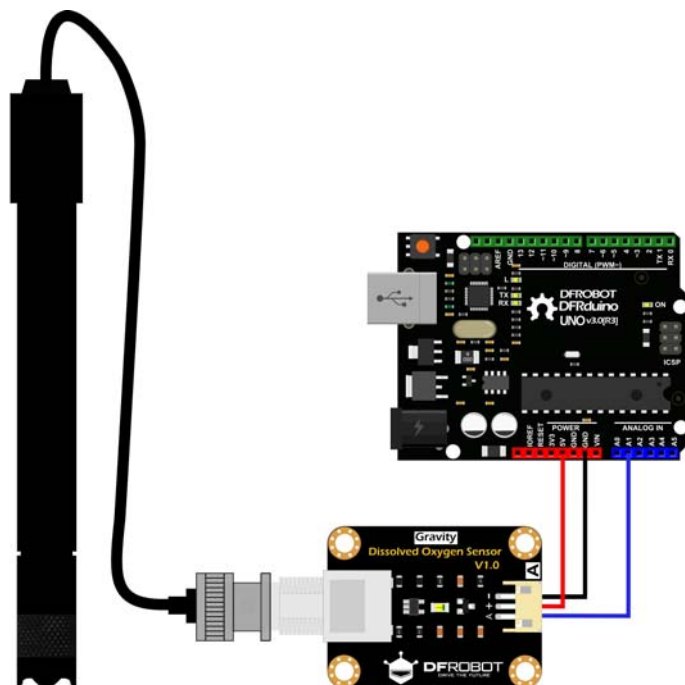


---

<sup>97</sup><https://www.dfrobot.com/product-1628.html>



Proben er galvanisk og trenger ikke tid for polarisering. *Under bruk skal beholder som omslutter sensoren fylles med en 0,5 mol/L NaOH-oppløsning (elektrolytt). Oppløsningen skal fylle hulrommet mellom membran og elektrode.* Beholderen med oppløsningen kan lett erstattes hvilket gjør vedlikehold enkelt. En ekstra beholder følger også med settet. Sensoren er utstyrt med et kretskort som konverterer signalet slik at det kan leses av en analog inngang hos mikrokontrolleren, her vist med en Arduino UNO på figuren under.



### **Merk følgende:**

1. Sensoren skal tilføres i en 0,5 mol/L NaOH oppløsning før bruk. Dette er en etsende oppløsning som krever forsiktighet. Den kan heller ikke sendes med leveransen av sensoren og må derfor lages på stedet. Man bør bruke hansker når man behandler oppløsningen. Skulle væsken komme på huden, skylle med mye vann. Vær oppmerksom på at NaOH kan være vanskelig å få av huden.
2. Vær oppmerksom på at membranen som slipper gjennom oksygenet, lett kan skades av fingerneglar og andre skarpe gjenstander. Vær derfor forsiktig når sensoren håndteres.
3. Sensoren vil under målingen forbruke noe oksygen, det er derfor viktig at det er bevegelse i oppløsningen slik at oksygenet fordeles jevnt i vannet omkring sensoren.
4. Sensoren er primært for laboratoriebruk, *hvilket betyr at den egner seg dårlig for kontinuerlig måling.* For kontinuerlig måling bør man bruke industriell standard som ligger i en annen prisklasse<sup>98</sup>.



### **Egenskaper:**

- Sensoren er galvanisk og krever derfor ingen polarisering
- Oppløsningen med tilhørende beholder med membran kan erstattes
- Forsyningsspenning 3,3 – 5,5V
- Analog utgangsspenning: 0 – 3,0V, passer også godt til MKR NB 1500 mikrokontroller
- Krever Gravity kontakt for tilkobling til kortet

### **Spesifikasjoner:**

*Proben som måler oppløst oksygen:*

- Type: Galvanisk
- Måleområde: 0 – 20mg/L
- Responstid: Opp til 98% av full respons i løpet av 90 sekunder ved 25°C
- Fungerer med vanntrykk fra 0 – 50Psi (0 – ca. 3500 mBar)  
Hvilket tilsvarer en dybde på ned til 25 m uho. (under havets overflate).
- Levetid ca. 1 år ved normal bruk, som sannsynligvis ikke betyr kontinuerlig bruk.
- Skifte av membran: 1 – 2 måneder ved gjørmete vann  
4 – 5 måneder ved rent vann  
Skifte av oppløsning: Hver måned (væsken rundt proben)
- Kabellengde: 2 meter
- Probetilkobling: BNC

*Kretskort for konvertering av signalet:*

- Forsyningsspenning: 3,3 – 5,5V
- Strømforbruk: < 1mA
- Probekontakt: BNC
- Signalkontakt: Gravity Analog Interface (PH2.0-3P)
- Dimensjoner: 42mm \* 32mm

### **Virkemåte og forberedelse av sensoren**

Som det framgår av figuren under så må beholderen med membranen foran på proben fylles med en oppløsning av NaOH med en konsentrasjon på 0,5 mol/L.

Siden det ikke er lov å sende med NaOH må denne blandes på stedet.

### ***Framstilling av NaOH oppløsning:***

For å lage en 1 molar oppløsning av et stoff, tar man en mengde av stoffet tilsvarende atomvekten

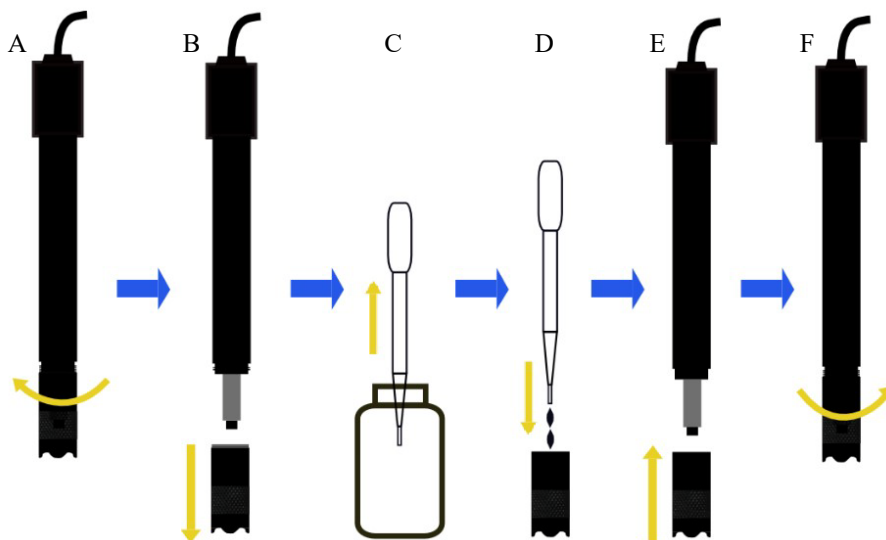
98. <https://atlas-scientific.com/probes/industrial-dissolved-oxygen-probe/>



i gram og tilsetter 1 liter vann. Atomvekten for NaOH er  $23 (\text{Na}) + 16 (\text{O}) + 1 (\text{H}) = 40$ . Dvs. for å lage 1 dL 1 molar oppløsning trengs 4 g NaOH, hvilket vil si at det trengs 2 g for å lage 1 dL 0,5 mol oppløsning, eller **1 g i 0,5 dL vann**.

### Tilsett NaOH oppløsningen til proben

Figuren under viser hvordan vi fyller proben (beholderen) med NaOH oppløsningen:

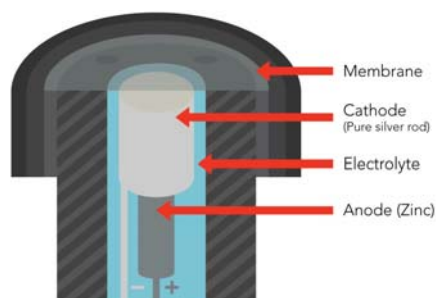


- Skru av beholderen med membranen foran på proben ...
- ... og ta den helt av
- Bruk en pipette for å fylle ...
- ... membranbeholderen 2/3 full med 0,5mol/L NaOH
- Skru membranbeholderen på proben
- Det er bra om litt av oppløsningen kommer ut idet beholderen skrues på, dermed vet vi at den er full

Pass på å sette lokk på resten av NaOH oppløsningen slik at man unngår at  $\text{CO}_2$  i lufta forringer oppløsningen.

### Virkemåte<sup>99</sup>

Forrest i membranbeholderen sitter en semi-permeabel membran (PTFE) som kun slipper gjennom oksygen ( $\text{O}_2$ ). Inne i membranbeholderen sitter proben som har en platina-katode og en sink-



<sup>99</sup><https://atlas-scientific.com/blog/how-do-dissolved-oxygen-probes-work/>



anode. Elektrolytten vil løse opp oksygenet slik at det blir istand til å motta et elektron (reduseres) ved katoden. På denne måten oppstår en spenningsforskjell og en strøm mellom katode og anode. Denne spenningsforskjellen forsterkes av en lavstøy forsterker. Jo mer oppløst oksygen, jo høyere spenning.

Det målte oppløste oksygeninnholdet er temperaturavhengig og må derfor kalibreres mht temperaturen.

## Kalibrering

Ved førstegangsbruk, eller etter en viss brukstid er det behov for kalibrering. Dette kan gjøres på to måter:

1. *Ett-punkts kalibrering*: Metoden måler mettet oppløst oksygen ved *en* temperatur. Denne metoden er enkel og egner seg når temperaturen under målingene er konstant og forutsigbar.
2. *To-punkts kalibrering*: Metoden måler mettet oppløst oksygen ved to eller flere temperaturer. Denne metoden egner seg dersom man ønsker å måle over et temperaturområde.

Her vil vi beskrive en ett-punktskalibrering og en to-punktskalibrering. *Vi trenger derfor rent vann som er mettet mht. oppløst oksygen.*

For å kunne utføre kalibreringen må vi kunne lese av spenningen fra sensoren. Det gjør vi via en av de analoge inngangene hos mikrokontrolleren. Vi velger å bruke følgende enkle kode for avlesning av spenningen og temperaturen kan vi måle med et termometer:

```
// Avlesning av spenning og temperatur for kalibrering av
oppløstoksygen
// Nils Kr. Rossing 16.12.22

#include <Arduino.h>
#include <DS18B20.h>
#include <OneWire.h>

DS18B20 ds(0); //Sensoren er koblet til pinne D0
uint8_t address[] = {40, 250, 31, 218, 4, 0, 0, 52};
uint8_t selected;

#define VREF 3300//VREF(mv)
#define ADC_RES 4096//ADC Resolution

void setup()
{
  Serial.begin(115200);
  selected = ds.select(address);
  analogReadResolution(12); // Sett oppløsningen til AD-konverte-
ren 12 bit
}

void loop()
{
```





```
float Spenning = 0;
// Les av og skriv ut spenning
/*
for (int i=0; i<100; i++)
{
  Spenning = Spenning + analogRead(A0);
}
  Spenning = Spenning/100;
*/
Spenning = analogRead(A0);

Serial.print("Spenning: ");
Serial.print(float(Spenning*VREF/ADC_RES));
Serial.println("V");

// Les av og skriv ut temperatur
float Temperature = ds.getTempC();
Serial.print("Temperatur: ");
Serial.print(Temperature);
Serial.println(" C");

Serial.println();
delay(1000);
}
```

Vi skal nå se på to ulike måter å kalibrere sensoren på:

### ***Ett-punktskalibrering***

1. Forbered sensoren for måling som omtalt foran
2. Dypp sensoren i rent springvann rist av overflødig vann
3. Eksponer sensoren for luft med en passende luftstrøm, ikke bruk en vifte.
4. Les av spenningen når den stabilisert seg og les av temperaturen. Verdien vi har fått er mettet oppløst oksygen for denne aktuelle temperaturen.

Resultatet kan f.eks. være:

```
CAL1_V = 1600mV
CAL1_T = 25°C
```

Metningspunktet for oppløst oksygen i vann varierer mye med temperaturen. Siden vanntemperaturen i vårt tilfelle vil variere, vil det sannsynligvis gi riktigere resultatet dersom det gjøres en to-punktskalibrering over et temperaturområde som omfatter det området vi forventer: f.eks. 0 – 20°C



### **To-punktskalibrering**

1. Tilbered to kopper med rent vann. Sett den ene koppen i kjøleskapet og den andre til oppvarming. Sørg for at temperaturen ikke overskrider 40°C.
2. Bruk en av følgende metoder for å lage vann med mettet oksygen. Bruk glasset med høyest temperatur:
  - A. Bruk en magnetrører og la den gå i ca. 10 minutter for å mette vannet med oksygen
  - B. Eller bruk en luftpumpe og pump luft ned i rent vann i 10 min.
3. Stopp røringen eller pumpingen og dypp sensoren ned i begeret etter at boblene er forsvunnet og fortsett å røre langsomt i vannet for å unngå at det dannes bobler.
4. Når spenningen er stabil, les av spenningen og temperaturen
5. Gjør det samme med glasset med det kalde vannet.

Resultatet kan f.eks. være:

```
CAL1_V = 1600mV  
CAL1_T = 25°C
```

og

```
CAL2_V = 1300mV  
CAL2_T = 15°C
```

### **Måleprogram:**

De målte kalibreringsverdiene settes inn i måleprogrammet som vist under i rødt:

```
// Oksygen_Maaling inkluderer kalibrering og måling av temperatur  
// Programmet bygger på programmet fra DFRobot:  
// https://wiki.dfrobot.com/Gravity\_\_Analog\_Dissolved\_Oxygen\_Sensor\_SKU\_SEN0237#More  
// men er skrevet om for å passe bedre for MKR NB 1500  
// Nilos Kr. Rossing 16.12.22  
  
#include <Arduino.h>  
#include <DS18B20.h>  
#include <OneWire.h>  
  
DS18B20 ds(0); //Sensoren er koblet til pinne D0  
uint8_t address[] = {40, 250, 31, 218, 4, 0, 0, 52};  
uint8_t selected;  
  
#define DO_PIN A0  
  
#define VREF 3300 //VREF (mv)  
#define ADC_RES 4096 //ADC Resolution  
  
//Single-point calibration Mode=0
```



```
//Two-point calibration Mode=1
#define TWO_POINT_CALIBRATION 0

//Single point calibration needs to be filled CAL1_V and CAL1_T
#define CAL1_V 2430 //mv
#define CAL1_T 33    //?
//Two-point calibration needs to be filled CAL2_V and CAL2_T
//CAL1 High temperature point, CAL2 Low temperature point
#define CAL2_V 1080 //mv
#define CAL2_T 12   //?

const uint16_t DO_Table[41] = {
    14460, 14220, 13820, 13440, 13090, 12740, 12420, 12110, 11810,
    11530,
    11260, 11010, 10770, 10530, 10300, 10080, 9860, 9660, 9460, 9270,
    9080, 8900, 8730, 8570, 8410, 8250, 8110, 7960, 7820, 7690,
    7560, 7430, 7300, 7180, 7070, 6950, 6840, 6730, 6630, 6530, 6410};

uint8_t Temperaturet;
uint16_t ADC_Raw;
uint16_t ADC_Voltage;
uint16_t DO;

int16_t readDO(uint32_t voltage_mv, uint8_t temperature_c)
{
    #if TWO_POINT_CALIBRATION == 0
        uint16_t V_saturation = (uint32_t)CAL1_V + (uint32_t)35 * tempera-
            ture_c - (uint32_t)CAL1_T * 35;
        return (voltage_mv * DO_Table[temperature_c] / V_saturation);
    #else
        uint16_t V_saturation = (int16_t)((int8_t)temperature_c - CAL2_T) *
            ((uint16_t)CAL1_V - CAL2_V) / ((uint8_t)CAL1_T - CAL2_T) + CAL2_V;
        return (voltage_mv * DO_Table[temperature_c] / V_saturation);
    #endif
}

void setup()
{
    Serial.begin(115200);
    analogReadResolution(12); // Sett oppløsningen til AD-konverteren
    12 bit
    selected = ds.select(address);

    Serial.println("Oksygen_Maaling");
}

void loop()
{
    Temperaturet = (uint8_t) ds.getTempC();
}
```



```
ADC_Raw = analogRead(DO_PIN);
ADC_Voltage = uint32_t(VREF) * ADC_Raw / ADC_RES;

Serial.print("Temperaturet:\t" + String(Temperaturet) + "\t");
Serial.print("ADC RAW:\t" + String(ADC_Raw) + "\t");
Serial.print("ADC Voltage:\t" + String(ADC_Voltage) + "\t");
Serial.println("DO:\t" + String(readDO(ADC_Voltage, Temperaturet))
+ "\t");

delay(1000);
}
```

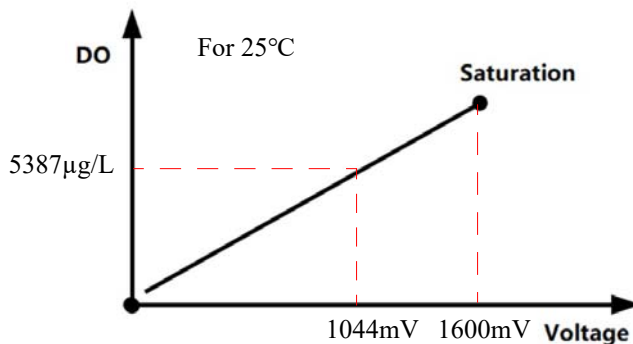
Etter at kalibreringsverdiene fra målingene er lagt inn i programmet og det er compilert og kjørt, kan vi få ut et resultatet som ligner på denne tabellen:

Temperatur:	25°C	ADC RAW:	215	ADC Voltage:	1049	DO:	5413
Temperatur:	25°C	ADC RAW:	214	ADC Voltage:	1044	DO:	5387
Temperatur:	25°C	ADC RAW:	214	ADC Voltage:	1044	DO:	5387
Temperatur:	25°C	ADC RAW:	213	ADC Voltage:	1040	DO:	5362
Temperatur:	25°C	ADC RAW:	213	ADC Voltage:	1040	DO:	5362

Her er temperaturen i °C, dernest avlest ADC digital verdi, så avlest spenningen i milliVolt, og tilslutt den beregnende oppløste mengden oksygen i µg/L. Ved å dele på 1000 får vi mg/L.

### ***Tolkning av ett-punktskalibrering***

Figuren under viser det målte metningspunktet for oppløst oksygen i vann ved temperaturen 25°C.

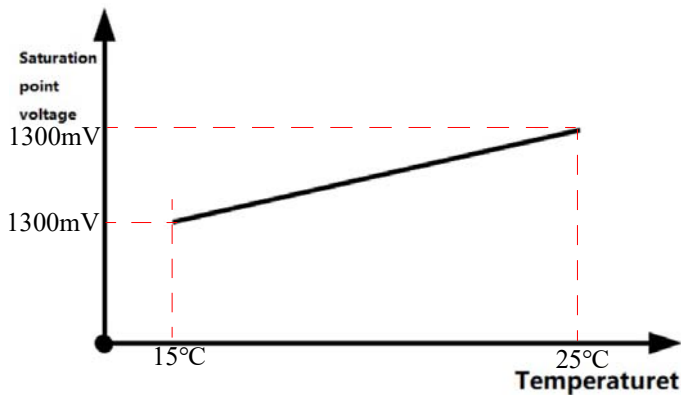


For 25°C antas en lineær sammenheng mellom spenning oppløst oksygen.



### Tolkning av to-punktskalibrering

Figuren under viser samlingen av metningspunkter for oppløst oksygen i vann med temperaturer fra 15 – 25°C.



Tabellen under gir en oversikt over metningspunktene som funksjon av temperaturen. Tabellen er integrert i programmet som beregner oppløst oksygen ut fra målt temperatur og spenning.

T °C	D0 mg/L	T °C	D0 mg/L	T °C	D0 mg/L
0	14.60	16	9.86	32	7.30
1	14.22	17	9.64	33	7.17
2	13.80	18	9.47	34	7.06
3	13.44	19	9.27	35	6.94
4	13.08	20	9.09	36	6.84
5	12.76	21	8.91	37	6.72
6	12.44	22	8.74	38	6.60
7	12.11	23	8.57	39	6.52
8	11.83	24	8.41	40	6.40
9	11.56	25	8.25	41	6.33
10	11.29	26	8.11	42	6.23
11	11.04	27	7.96	43	6.13
12	10.76	28	7.83	44	6.06
13	10.54	29	7.68	45	5.97
14	10.31	30	7.56	46	5.88
15	10.06	31	7.43	47	5.79



## Målinger

1. Vi tok springvann i to erlenmeyerkolber. En satte vi i romtemperatur (varm) på laboratoriet og en i kjøleskapet (kald).
2. Ved kalibrering tok vi først den varme og rørte kraftig i denne i 2 – 3 min. Vi antok da at den var mettet med oksygen. Deretter satte vi den i et vannbad på ca. 40°C og målte spenning og temperatur lik: 2430 mV @ 33°C.
3. Dernest tok vi den kalde og rørte også denne kraftig i 2 – 3 min. Deretter målte vi spenning og temperatur til 1080 mV @ 12.



4. Vi la så inn disse verdiene i måleprogrammet som beskrevet og målte en prøve av havvann, ved 21 °C. Etter at den hadde stabilisert seg fikk vi verdier i nærheten av 3,1 mg/L.
5. Ved å starte magnetrøreren så spratt verdien for oksygeninnholdet opp til 6,9 mg/L.
6. Ved å røre kraftig i havvannet så kom verdien opp til 4,4 mg/L

Skal vi tolke resultatene kan det se ut til at røring med magnetrøreren er langt mer effektivt enn kraftig omrøring med glasstav. Ved demontering og vasking av sensoren oppdaget vi at selve elektroden berørte membranen på innsiden og at det var merker etter berøringen.

### Noen vanlige spørsmål:

1. **Hvordan lage en mettet oppløsning av oksygen i vann?**  
Bruk en akvariepumpe og kjør luft gjennom vannet i 20 minutter og vannet er 100% mettet.
2. **Hvordan lage vann med 0 oppløst oksygen?**  
Bland Natriumsulfid ( $\text{Na}_2\text{SO}_3$ ) i vannet inntil det blir mettet av stoffet. Dette stoffer tiltrekker seg oksygenet.
3. **Oppbevaring av sensorer**  
A) Oppbevaring inntil en uke: Dypp sensoren i destillert vann for å unngå at NaOH-oppløsningen fordampes. Koble sensoren fra kretskortet.



B) Oppbevaring ut over en uke: Skru av beholderen og tøm ut elektrolytten (NaOH). Vask elektodene i probespissen i destillert vann. Vask også beholderen. Tørk alle delene med litt tørkepapir, vær forsiktig med membranen. Skru på beholderen uten å fylle i ny elektrolytt. Legg sensoren tilbake i boksen.

4. **Hvilke problemer er det vanlig å møte?**

A) Dersom avlesningen ikke viser 0 oppløst oksygen, dersom man setter den i en oppløsning uten oksygen, kan man pusse over katoden i proben.

B) Dersom målinger viser urimelige verdier eller målingen driver, undersøk om membranen er ødelagt. Dersom membranen er skadet eller forurenset, bytt beholder.

## 6.5 Sensor for måling av vannets ledningsevne og saltholdighet (DFR0300-H)

Sensoren måler vannets ledningsevne (konduktivitet). Det er en nær sammenheng mellom ledningsevnen og saltholdigheten i sjøvann (salinitet). Sensoren er tilpasset bruk sammen med Arduino og leveres av firmaet DFRobot<sup>100</sup>.



Sensoren er spesielt beregnet til å måle ledningsevnen i saltvann med et måleområde opp til 100mS/cm (milliSiemens). Sensoren fungerer med spenningskilder fra 3,0 – 5,0V. For å unngå polarisering av proben tilføres den en vekselspanning (AC) hvilket øker presisjonen og levetiden til proben. Sensoren anvender en ett-punktskalibrering og er istand til å gjenkjenne standard bufferløsninger.

Merk følgende:

1. Sensoren er beregnet på laboratoriebruk og bør derfor ikke stå i saltvann over lengre tid. *Dette kan være et alvorlig ankepunkt mot å bruke denne til vårt formål.*

---

100.<https://www.dfrobot.com/product-1797.html>





2. Platinaelektroden er dekket med et svart lag som ikke bør berøres eller skades på noe vis. Skader i dette laget vil kunne medføre unøyaktige målinger. Proben kan evt. rengjøres med destillert vann.

### Spesifikasjoner:

*Proben som måler ledningsevnen i saltvann:*

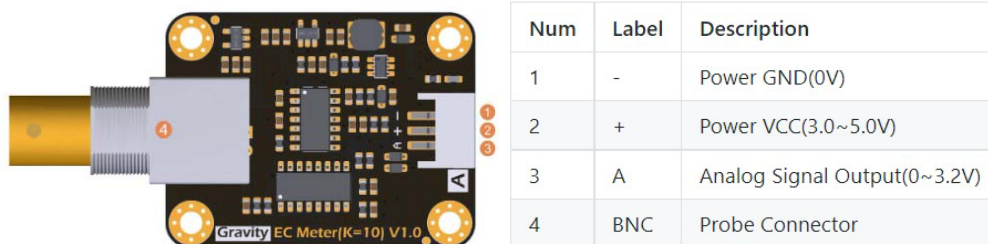
- Type: For laboratoriebruk
- Måleområde: 10 – 100mS/cm
- Cellekonstant: 10 ±2
- Temperaturområde: 0 – 40°C
- Levetid > 0,5 år under normal bruk, som ikke betyr kontinuerlig bruk.
- Kabellengde: 1 meter

*Kretskort for konvertering av signalet:*

- Forsyningsspenning: 3,0 – 5,0V
- Strømforbruk: TBD (To Be Determined)
- Probekontakt: BNC
- Signalkontakt: Gravity Analog Interface (PH2.0-3Pin)
- Målenøyaktighet ±5% av F.S. (fullt utslag)

### Signalbehandlingskort og tilkoblinger

Figuren under viser signalbehandlingskortet med tilkoblinger.



### Bruk og virkemåte

For å få mest mulig nøyaktige målinger anbefales på det sterkeste å gjøre samtidige målinger av temperaturen. Det anbefales å skylle proben i destillert vann mellom målinger av ulike væsker slik at en unngår å forurense målingene. *Det anbefales også å bruke eksternt power supply, dvs. at man skiller spenningen som forsyner mikrokontrolleren fra den som forsyner signalbehandlingskortet for sensoren.*



---

Sensoren er primært for laboratoriebruk, hvilket betyr at den egner seg dårlig for kontinuerlig måling. For kontinuerlig måling bør man bruke industriell standard som ligger i en annen prisklasse<sup>101</sup>. *Dette er en egenskap som ha betydning for vår bruk.*

***Utstyr:***

- 1 x Arduino mikrokontroller f.eks. MKR NB 1500
- 1 x Analog Electrical Conductivity Meter Board (K=10)
- 1 x Electrical Conductivity Probe (K=10)
- 1 x Standard Buffer oppløsning 12.88mS/cm
- 1 x Gravity 3pin Sensor kabel
- 1 x Testløsning

***Programvare:***

- Arduino IDE
- Last ned og installer DFRobot\_EC10 biblioteket<sup>102</sup>
- Eksempelkode for kalibrering (se vedlegg G.3 side 275)

---

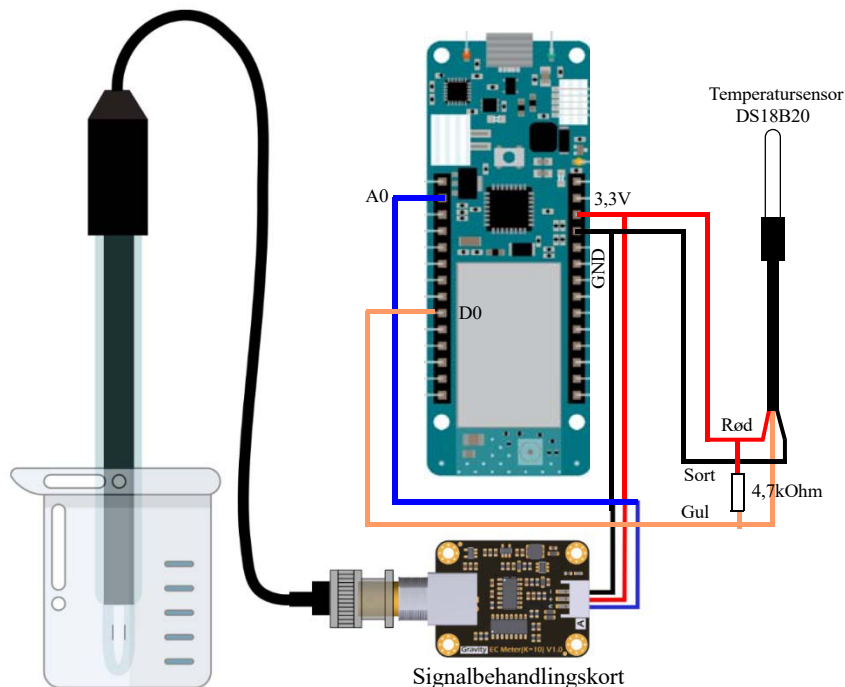
101. <https://atlas-scientific.com/kits/conductivity-k-10-kit/>

102. [https://codeload.github.com/DFRobot/DFRobot\\_EC10/zip/master](https://codeload.github.com/DFRobot/DFRobot_EC10/zip/master)



## Oppkobling

Figuren under viser en typisk oppkobling. Signalbehandlingskortet forsynes med 3,3 V og signalutgangen tilkobles en av de analoge inngangene, i dette tilfellet A0. I eksempelet under er benyttet mikrokontrolleren MKR NB 1500 som normalt har 3,3V som arbeidsspenning. Her har vi også inkludert temperatursensoren DS18B20 som vi kan stikke ned i kolben.



## Kalibrering og tilrettelegging for måling

Siden vi bruker samme arbeidsspenning på både mikrokontrolleren og signalbehandlingskortet er det uproblematisk å koble disse sammen. Vi velger likevel å sette opp AD-konverteren til 12 bit, som vi har anledning til hos Arduino MKR NB 1500.

Siden eksempelprogrammet som følger med sensoren er beregnet til å måle ledningsevnen i væsken og dessuten bruker et bibliotek som ikke er kompatibelt med Arduino MKR NB 1500, så lager vi et enkelt program som vi kalibrerer direkte mot salinitet (saltholdighet), dermed unngår vi mange problemer.

Vi har derfor laget et program som vi kan bruke for å kalibrere salinitets-sensoren. Vi gjør følgende antakelser:

- ... at det er en nær lineær sammenheng mellom spenningen fra sensoren og saliniteten målt i promille saltholdighet.
- ... at saliniteten (ledningsevnen) til rent destillert vann gir en spenning nær 0,0V



- ... at forsyningsspenningen levert av mikrokontrolleren til signalbehandlingskortet er tilstrekkelig stabil

I tillegg gjør vi følgende valg:

- AD-konverteren hos mikrokontrolleren settes til 12 bit
- Vi regner ikke om til spenning, men kalibrerer sensoren på bakgrunn av digitale verdier
- Vi bruker middelverdien av minst 100 målinger for å redusere betydningen av støy
- Vi lager oss kalibrerte blandinger (0 ‰, 8,75 ‰, 17,5 ‰, 35 ‰ og 70 ‰) med NaCl som vi bruker til å finne en sammenheng mellom målt spenning og salinitet

Spenningen vil øke med økende salinitet.

### Framstilling av standard blandinger

Vi velger å lage oss ca. 200mL av følgende blandinger: 0 ‰, 8,75 ‰, 17,5 ‰, 35 ‰ og 70 ‰. Vi tar utgangspunkt i 500mL med 70 ‰ salinitet. En slik kan vi lage på følgende måte:

- Natriumklorid/salt, (NaCl)
- Destillert vann eller milliporevann (vi kan sannsynligvis også bruke springvann)
- Magnetrører
- Erlenmeyer kolber (5 stk. 250 mL)
- Erlenmeyer kolbe (1 stk. 1000 ml)

Fyll kolben med 500 ml destillert vann (eller springvann). Tilsett 33,03 g natriumklorid (NaCl) og bland til saltet er oppløst. Denne løsningen har salinitet på 70 ppt ved 25 °C.

Dernest lager vi de andre konsentrasjonene slik:

1. 500 mL (70 ‰) → 200mL 70,0 ‰
2. 500 mL (70 ‰) → 100 mL + 100 mL rent vann → 200 mL 35,0 ‰ – tilsvarer sjøvann
3. 500 mL (70 ‰) → 50 mL + 150 mL rent vann → 200 mL 17,5 ‰ – tilsvarer brakkvann
4. 500 mL (70 ‰) → 25 mL + 175 mL rent vann → 200 mL 8,75 ‰
5. 200 mL rent vann

### Målinger

Vi bruker måleprogrammet: *Salinitetsmåler\_DFR0300\_H* (se vedlegg G.3 side 275, programmet kan også lastes ned fra nettsiden [www.ntnu.no/skolelab/bla-hefteserie](http://www.ntnu.no/skolelab/bla-hefteserie)). Det er foretatt målinger av spenningen med ulike konsentrasjoner av saltblandinger, angitt i ‰ (promille) (se tabell 3).

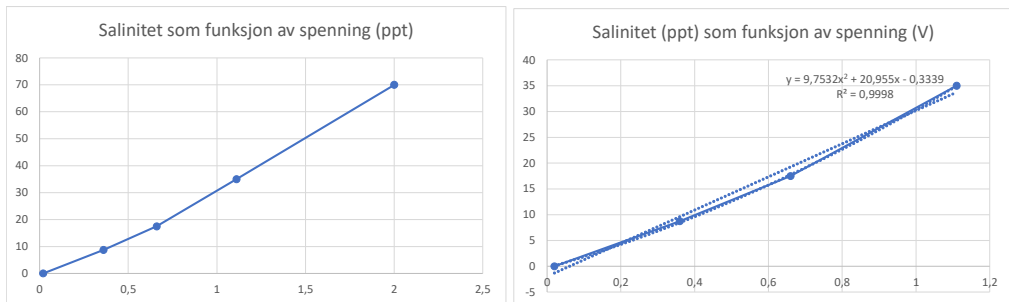
Tabell 3: Spenning målet med forskjellig salinitet ved 20°C.

Salinitet [‰]	Spenning [V]
0 ‰	0,02 V



Salinitet [‰]	Spenning [V]
8,75 ‰	0,26 V
17,50 ‰	0,66 V
35,00 ‰	1,11 V
70,00 ‰	2,00 V

Til venstre på figuren under har vi tatt med alle fem målingene (0 til 70‰), og vi ser at kurven er relativt lineær ned til 15 ‰, derfra og ned å krummer den. Siden vi er interessert i området rundt 35 ‰ og under, har vi valgt å gjøre en regresjon av 2. grad for de fire laveste målepunktene slik at vi kan sette opp en matematisk sammenheng mellom målt spenning og salinitet for den aktuelle temperaturen.



$$\text{Salinitet} = 9,7532 * \text{Spenning}^2 + 20,955 * \text{Spenning} - 0,3339 \text{ [‰]} \quad (6.10)$$

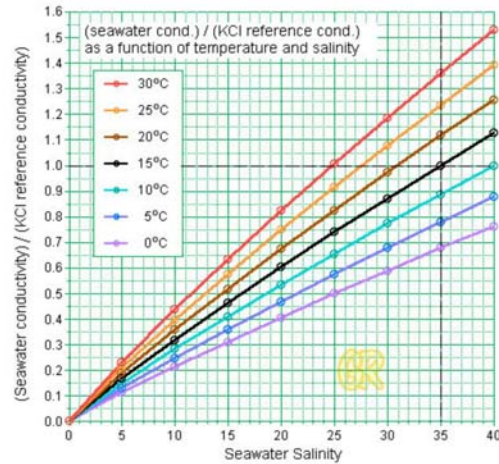
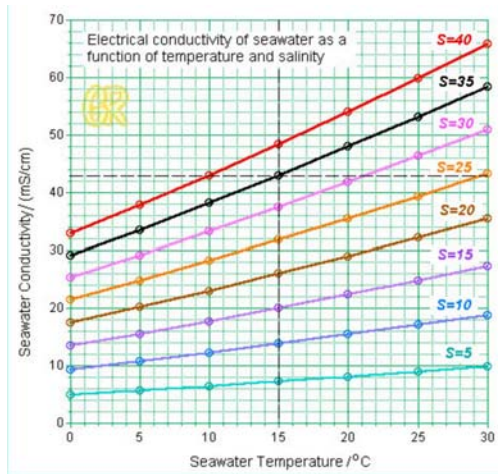
Måling av havvann hentet ute ved Trondheim biologiske stasjon (TBS) er målt til:

Saltholdighet i vannprøve fra Trolla (Trondheimsfjorden) 34,3 ‰ @ 20°C.

Det vi må huske på er at vi har målt saltholdigheten i sjøvannet referert til standarder laget med NaCl som skal være riktig ved 25°C. Målingene er imidlertid gjort ved 20°C. Siden den aktuelle temperaturen er vesentlig lavere så bør vi kompensere for endring i målt salinitet mht. temperaturen.

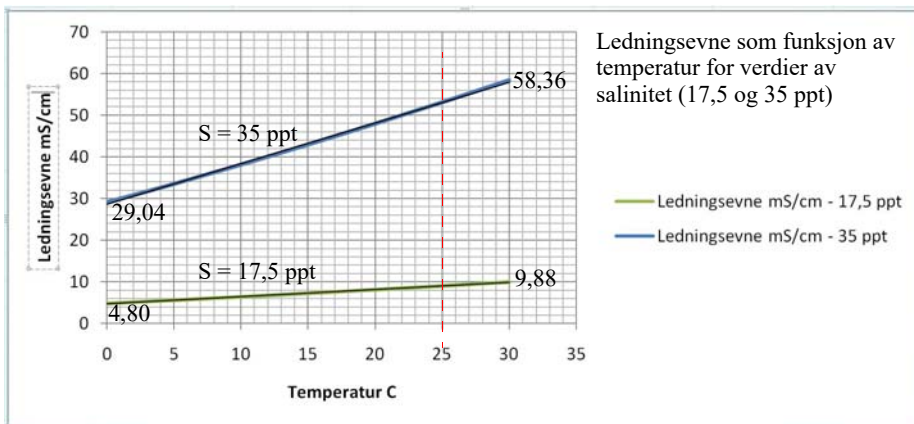


Ved bruk av kalkulatorer som disse kan man lage kurveskarer som viser sammenhengen mellom de ulike parametrene som vist på figuren under<sup>103</sup>:



Grafene til venstre viser ledningsevne som funksjon av temperaturen i °C for ulike verdier av salinitet. Grafene til høyre viser relativ ledningsevne som funksjon av salinitet for ulike temperaturer. Referanseløsningen er laget av KCl og har en salinitet på 35 ‰ ved 15°C<sup>104</sup>.

I vårt tilfelle, hvor vi ønsker å måle saliniteten i sjøvann nær verdien 35 ‰, så kan det være interessant å se hvordan ledningsevnen endrer seg med temperaturen under forutsetning av en nær konstant salinitet.



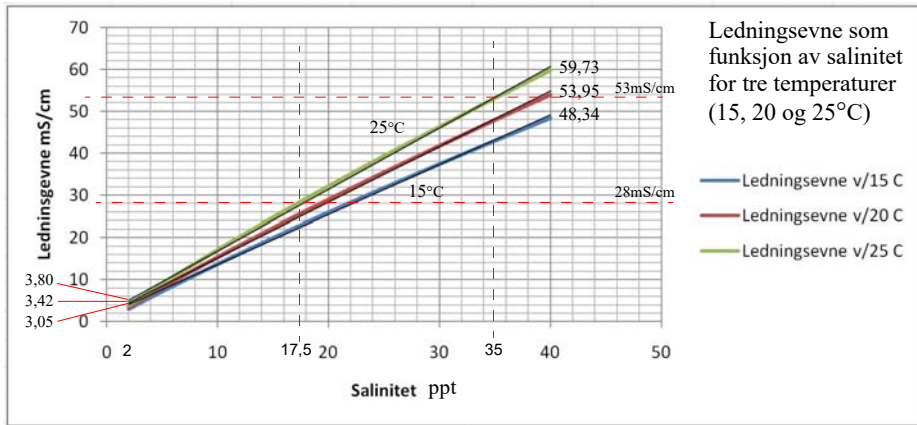
Grovt sett ser vi at ledningsevnen pr. cm dobles over en temperaturøkning fra 0°C – 30°C og at økningen er omtrent lineær over temperaturområdet.

103.<http://www.grabovrat.com/handbook/handbookH08g1.html>

104. Standard vannprøven er laget ved å blande ut 32.4356 gram KCl i 1 liter destillert vann som gir akkurat 35‰ ved 15°C og 1 atm.



Figuren under viser ledningsevnen i mS/cm som funksjon av salinitet ved tre temperaturer (15°, 20° og 25°C). Vi legger merke til at også denne sammenhengen er ganske lineær.

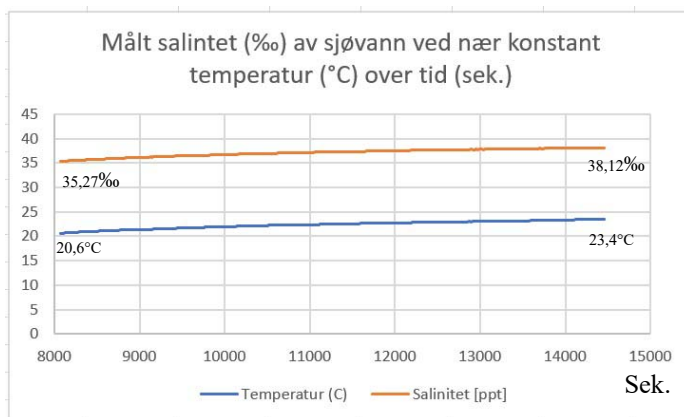


La oss si at vi gjør en kalibrering ved to verdier av salinitet (f.eks. 17,5 ppt og 35 ppt) ved en temperatur (f.eks. 25°C). Er det da mulig å finne en forenklet formel for salinitet som funksjon av en digital måleverdi innen et begrenset temperatur- og salinitets-område? Klarer vi å finne en slik sammenheng hvor vi kan sette inn våre kalibrerte verdier for så å interpolere/ekstrapolere verdier innen et begrenset temperatur- og salinitets-område, kan vi ha en nyttig modell for vårt formål.

For omregning fra ledningsevne til saltholdighet se vedlegg E.1 side 266. For ytterligere fordypning av måling av saltholdighet se referanse [2].

### Måling av salinitet som funksjon av temperatur

Innledningsvis ville vi se hvordan saliniteten i sjøvann endret seg over tid med omtrent konstant temperatur. Figuren under viser salinitet i en sjøvannsprøve i løpet av ca 2 timer. Selv om vi har forsøkt å holde temperaturen konstant, ser vi at den endrer seg noe (ca. 2,8 °C). Endringen skyldes sannsynligvis endring av temperatur i laboratoriet evt. endring på grunn av magnetrøreren som avgir litt varme.

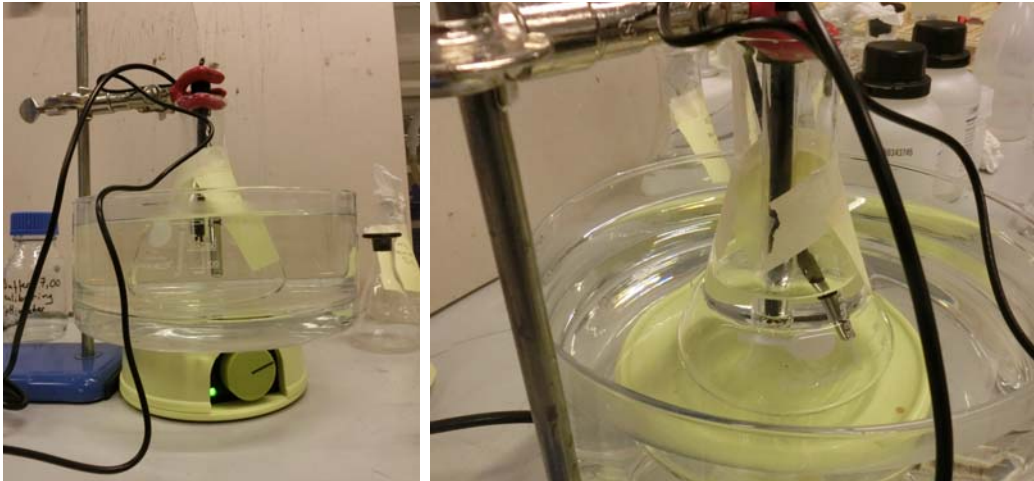




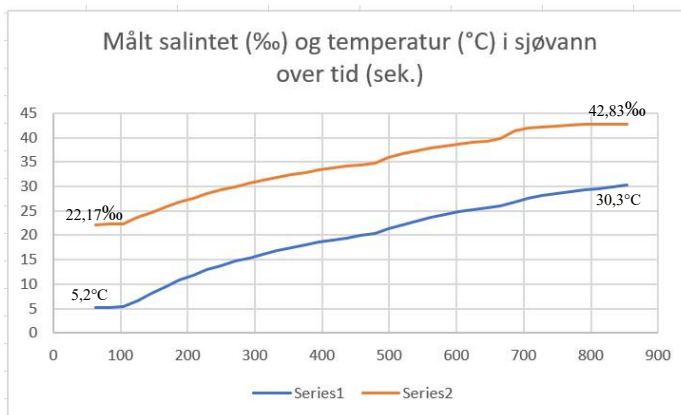


Målingene er gjort med salinitetsproben og sensoren DS18B20 og samlet inn med en Arduino MKR NB 1500. Det er benyttet en erlenmeyerkolbe fylt med ca. 200 mL sjøvann hentet fra Trondheimsfjorden (Trolla). Målingen er gjort under konstant omrøring.

For å kunne registrere målt saltholdighet som funksjon av temperaturen på en effektiv måte, valgte vi å sette en avkjølt (ca. 5°C) saltvannsprøven i et vannbad med en temperatur på nærmere 40°C. Vannbadet ble satt på en magnetrører som vist på bildet under. Det nedkjølte sjøvannet vil dermed få en langsom temperaturstigning etter som tiden går.

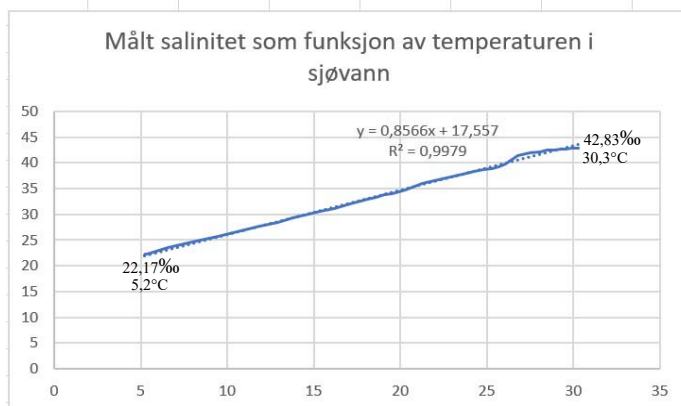


Vi antar at den reelle saliniteten til prøven er konstant som funksjon av temperaturen, og at en evt. registrert forandring av salinitet skyldes sensoren. Effekten kommer tydelig fram i den neste grafen målt salinitet som funksjon av temperaturen som endrer seg fra ca. 5°C til 30°C.



Figuren over viser hvordan målt salinitet øker med økende temperatur.

Denne sammenhengen er tydeliggjort i den neste figuren som viser målt salinitet som funksjon av temperaturen. Måleserien er gjort over drøyt 13 minutter i en 250 mL erlenmeyerkolbe med omrøring. Kolben er plassert i et vannbad med en starttemperatur på ca. 40°C.



Ved å legge inn trendlinjen ser vi at sammenhengen mellom temperatur og målt salinitet er ganske så lineær ( $R^2 = 99,79$ ), med følgende foreslåtte sammenheng:

$$\text{Salinitet} = 0,8566 * \text{Temperatur [C]} + 17,557 \text{ for } [5,2 - 30,3^{\circ}\text{C}] \quad (6.11)$$

Den tilsvarende formelen for nær konstant temperatur er:

$$\text{Salinitet} = 0,9836 * \text{Temperatur [C]} + 15,095 \text{ for } [20,6 - 23,4^{\circ}\text{C}] \quad (6.12)$$

Vi antar at målingen som er gjort over et større temperaturområde er mest nøyaktig så vi bruker den som utgangspunkt for å lage en korreksjon.

Vi tar utgangspunkt i våre salinitetsstandarder framstilt av NaCl-blandingen (på 8,75, 17,5, 35 og 70 ‰ @ 25°C). Med utgangspunkt i disse målingene fant vi en sammenheng mellom målt spenning og salinitet som er uttrykt i ligning 6.10 side 186:

$$\text{Salinitet} = 9,7532 * \text{Spennings}^2 + 20,955 * \text{Spenning} - 0,3339 [\text{‰}] \quad (6.13)$$

Med utgangspunkt i denne målte vi salinitet i sjøvann til 34,3 ‰ ved 20°C.

Vi ønsker å korrigere salinitetsmålingene mht. temperaturen og antar at korreksjonen er 0 ‰ @ 20°C. Som korreksjon tar vi utgangspunkt i ligning 6.11 side 190:

$$\text{tempSalinitet} = 0,8566 [\text{‰}/^{\circ}\text{C}] * \text{Temperatur } [^{\circ}\text{C}] + 17,557 [\text{‰}] \quad (6.14)$$

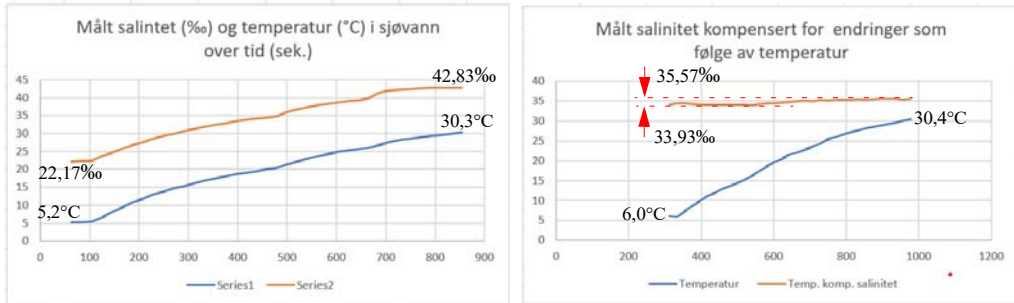
Vi finner korreksjonen ved å trekke fra

$$\text{deltaSalinitet} = 0,8566 [\text{‰}/^{\circ}\text{C}] * (\text{Temperatur}) [^{\circ}\text{C}] + 17,557 - 34,3 [\text{‰}] \quad (6.15)$$

$$\text{Temp. kompensert salinitet} = \text{Salinitet} - \text{deltaSalinitet} \quad (6.16)$$



Figuren under viser korrigert og ukorrigert salinitetsmåling:



### Eksempelkode for konduktivitetsensor (salinitet)– DFR0300-H

Programmet vist under er utviklet uten bruk av biblioteker og kan derfor lastes opp og kjøres på MKR NB 1500. Ved hjelp av målingene omtalt foran kompenseres for temperaturvariasjoner. Det antas at innen det aktuelle temperaturområdet er vannets salinitet uforandret.

```
// Salinitetsmåler_DFR0300_H
// Programmet måler og beregner Salinitets verdien til en væske på bakgrunn av
// kalibreringsverdier for ferskvann 0, 17,5 og 35 promille
// Supply-spenning 3,3 V
// Nils Kr. Rossing 14.08.23
```

```
#include <DS18B20.h>
#include <OneWire.h>
```

```
DS18B20 ds(0); //Sensoren er koblet til pinne D0
uint8_t address[] = {40, 250, 31, 218, 4, 0, 0, 52};
uint8_t selected;
```

```
// Deklarasjon av variabler
float Salinitet = 0; // Målt og beregnet salinitets-verdi
float korreksjon = 0; // Korrigeringsverdi mht temperatur
float korrSalinitet = 0; // Korrigert salinitet
float UD_35 = 0.82; // Målt digital spenningsverdi ved 35 promille
float UD_175 = 0.49; // Målt digital spenningsverdi ved 17,5 promille
```

```
void setup() {
  Serial.begin(115200); //Baud rate: 115 200
  selected = ds.select(address);
  analogReadResolution(12); // Sett oppløsningen til AD-konverteren 12 bit
}
```



```
void loop() {
  float Temperature = 0; // Målt temperatur ved hjelp av DS18B20
  float korrSalinitet = 0; // Korreksjonsverdi av salinitet mht
  float korreksjon = 0; // Korreksjonsverdi mht temperaturendringer
  long sensorValue = 0; // Les av analog inngang fra pinne A0:
  float UD_mid = 0; // Middelerverdi av målt spenning
  float UD_mid_volt = 0; // Middelerverdi av målt spenning

  for(int i=0; i<100; i++)
  {
    sensorValue = sensorValue + analogRead(A0);
  }
  UD_mid = float(sensorValue/100); // Finner middelerverdien

  UD_mid_volt = UD_mid * (3.32 / 4096.0); //
  Temperature = ds.getTempC(); // Måler temperaturen

  // Konverterer den analoge avlesningen (som går fra 0-4095) til en spenning 0 -
  // 3,0V:
  Serial.print(millis()/1000,1);
  //Serial.print(";");
  Serial.print("Midlere spenning: "); // Skriv ut den avleste verdien for
  // kalibrering:
  Serial.print(UD_mid_volt);

  // Beregner Salinitet
  Salinitet = 9.7532 * UD_mid_volt * UD_mid_volt + 20.955 * UD_mid_volt - 0.3339;
  // Polinomtilpasset av 2 grad R^2 0,9998
  korreksjon = 0.8566*(Temperature) + 17.557 - 34.3;
  korrSalinitet = Salinitet - korreksjon;

  Serial.print(", Midlere temperatur: "); // Skriv ut den avleste verdien for
  // kalibrering:
  //Serial.print(";");
  Serial.print(Temperature,1);

  Serial.print(", Ukorrigert salinitet: "); // Skriv ut den avleste verdien:
  //Serial.print(";");
  Serial.println(Salinitet,2);

  Serial.print(", Korrigert salinitet: "); // Skriv ut den avleste verdien:
  //Serial.print(";");
}
```



```
Serial.println(korrSalinitet,2);

delay(20000);
//int minutter = 1;
//for (int j=0; j<minutter; j++) delay(60000);
```

## 6.6 Sensor for måling av turbiditet (SEN-0189)<sup>105</sup>

Turbiditetssensoren måler vannkvaliteten ved å registrere konsentrasjonen av partikler i væsken (turbiditet), dvs. sensoren sender lys inn i væsken og ser hvor mye av lyset som spres. Jo mer som spres, jo flere partikler er det i vannet. På tilsvarende måte kan man også måle hvor mye av lyset som absorberes når det går gjennom et væskevolum. Et slikt måleinstrument kalles et *kolorimeter* og måler absorbansen i væsken, dvs. hvor stor del av lyset som absorberes på veien gjennom væsken.



Slik proben er utformet kan det se ut som om sensoren oppfører seg mer som et *kolorimeter* enn et turbidimeter.

Sensoren består av en lyskilde og en lyssensor plassert i hver sin grein foran på sensoren. Lys som sendes ut fra lysdioden passerer gjennom et væskevolum og vil så fanges opp av lyssensoren. Jo større konsentrasjon av partikler i væsken, jo mer lys blir spredt og kommer aldri fram til sensoren.

Siden denne sensoren ikke er et virkelig turbidimeter så er det man måler utsendt lys minus det som spres og det som absorberes i væsken.

### Spesifikasjon

Noen viktige parameterne:

- **Arbeidsspenning er 5V**, som ikke er optimalt for MKR, men håndterbart.
- **Strømforbruk under måling <40 mA**. Sensoren trekker altså en god del strøm. Primært skyldes dette lysdioden.
- **Responstiden er < 500 msek**, hvilket er ganske lang tid, men håndterbart.
- **Analog utgang** gir ut en spenning som funksjon av turbiditeten
- **Digital utgang** gir høy på utgangen når turbiditeten passerer et terskelnivå som kan settes med et potensiometer.

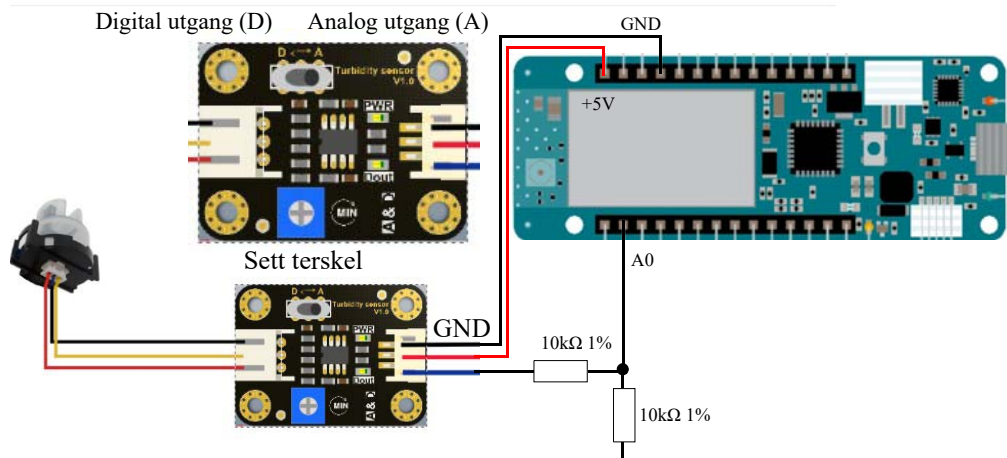
### Oppkobling:

Figuren under viser oppkoblingen til en MKR NB 1500. Siden MKR NB 1500 benytter en arbeidsspenning på 3,3 V må man sørge for at analoginngangen til MKR ikke overskrider 3,3V.

<sup>105</sup>[https://wiki.dfrobot.com/Turbidity\\_sensor\\_SKU\\_\\_SEN0189](https://wiki.dfrobot.com/Turbidity_sensor_SKU__SEN0189)

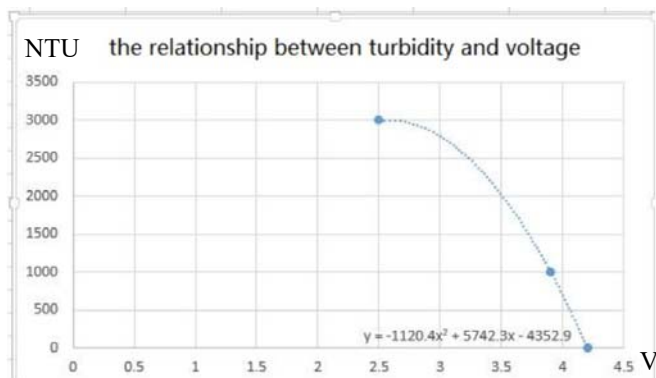


Dette gjør vi ved hjelp av spenningsdeler som halverer spenningen. Deretter må vi passe på å doble spenningen når vi bruker den i programmet vårt.



Ved hjelp av en bryter D/A kan vi velge mellom analogt og digitalt signal på utgangen.

**A** – Settes bryteren i posisjon A så leverer utgangen en spenning som faller med økende turbiditet (økende tetthet av partikler). Kurven i figuren under antyder sammenhengen mellom turbiditet og utgangsspenning. Vi merker oss at spenningen i dette diagrammet kan befinne seg fra ca. 4,2 til 2,5V i området 0 – 3000 NTU, som er i meste laget for Arduino MKR.



Som vi ser så er sammenhengen mellom spenning og turbiditet temmelig ulineær. Det er kun for lave turbiditetsverdier, 0 – 1500 NTU, at sammenhengen er tilnærmet lineær.

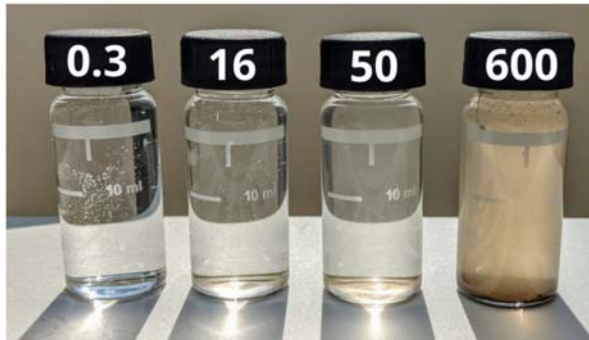
**D** – Settes bryteren i posisjon D så gir utgangen en høy spenning dersom turbiditeten overskrider et terskelnivå som kan settes av potensiometeret merket “Sett terskel” på figuren over.

Måleenheten NTU står for *Nephelometric Turbidity Units* og er vanlig brukt i USA. 1 NTU tilsvarer ca. 1 mg/L. En annen enhet som også er brukt i USA er Jackson Turbidity Unit (JTU) hvor 1 JTU = 1 NTU. I Europa er det vanlig å bruke enheten Formazin Nephelometric Units (FNU).

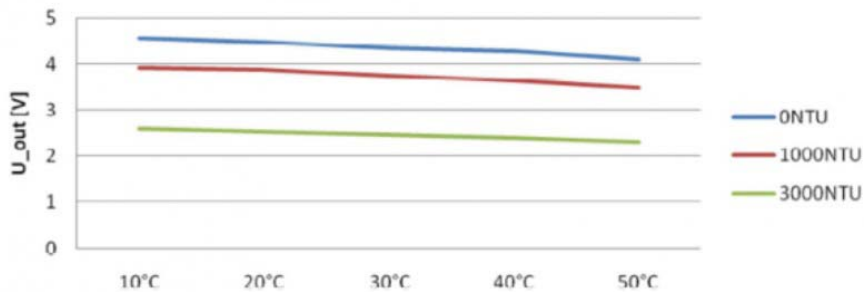


Her velger vi å bruke måleenheten NTU siden det er den enheten produsenten av dette måleinstrument bruker.

Figuren under antyder hvordan oppløsninger med forskjellig verdier av NTU kan se ut<sup>106</sup>.



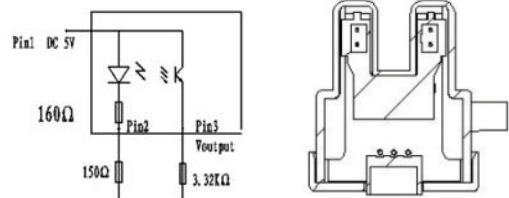
Måleresultatet er også avhengig av temperaturen som vist i figuren under.



Vi registrerer at spenningen kan falle med så mye som 0,5V når temperaturen øker fra 10 – 50°C. Dette skyldes sannsynligvis endringer i elektronikken og ikke forandringer i turbiditet.

### Koblingsskjema og mekanisk utforming

Figuren til høyre viser det elektriske koblingsskjema for sensoren. Den består av en lysdiode og en fototransistor. Vi legger merke til at lyskilden og sensoren er plassert rett overfor hverandre, dermed er det lysmengden som slipper gjennom oppløsningen framfor det som spres, som måles.



106. <https://freeup.world/turbidity/>

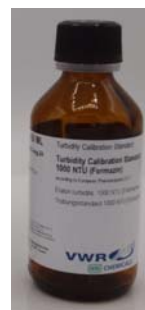




## Testbetingelse og kalibrering

Dersom sensoren settes i rent vann med en NTU < 0,5 så skal utgangspenningen være  $4,1V \pm 0,3V$ . Andre standarder kan lages ut fra en 400 NTU standard<sup>107</sup>. For å lage en 200 NTU standard tar man like deler 400 NTU og destillert vann og blander godt.

Det er heller ikke uvanlig å ta utgangspunkt i en Formazine løsning på f.eks. 100 mL av 1000 NTU eller 500mL av 500 NTU og blande ut disse til man får en serie med standardløsninger. Slike kan bl.a. kjøpes hos VWR<sup>108</sup>. Formazine er et meget tungt oppløselig stoff som danner små dråper i vannet og som dermed egner seg godt som standard for spredningsmålinger. Formazine er en blanding av 10 g/L hydrazine sulfat og 100 g/L hexamethylenetetramin med destillert vann.



*Men som sagt turbidimeteret SEN-0189 er ikke et ekte turbidimeter, men heller et kolorometer som både måler summen av spredning og absorpsjon.*

## Kalibrering og tilrettelegging for måling

Siden spenningen fra sensorkortet overskrider 3,3V må vi redusere spenningen før vi kan sende den inn på en analog inngang som vist i oppkoblingen. På denne måten mister vi oppløsning. Dette kan vi imidlertid ta igjen ved å konfigurere AD-konverteren til 12 bit, noe vi har anledning til hos Arduino MKR NB 1500.

Eksempelprogrammet leser kun av spenningen. Skal man lese av turbiditeten i NTU-enheter må avlesningene kalibreres, som krever standarder.

Vi har derfor laget et program som vi kan bruke for å kalibrere turbiditetssensoren. Vi gjør følgende antakelser:

- ... at det er en lineær sammenheng mellom spenningen fra sensoren og turbiditet målt i NTU-enheter.
- ... at turbiditeten for rent destillert vann har en NTU-verdi lik null og gir en spenning på 4,1V
- ... at spenningen levert av mikrokontrolleren er stabil nok

I tillegg gjør vi følgende valg:

- AD-konverteren i mikrokontrolleren settes til 12 bit
- Vi regner ikke om til spenning, men kalibrerer sensoren på bakgrunn av digitale verdier
- Vi bruker middelverdien av minst 100 målinger for å redusere betydningen av støy
- Vi bruker en oppløsning av Formazin med NTU-verdier på 200 og 400 under kalibreringen

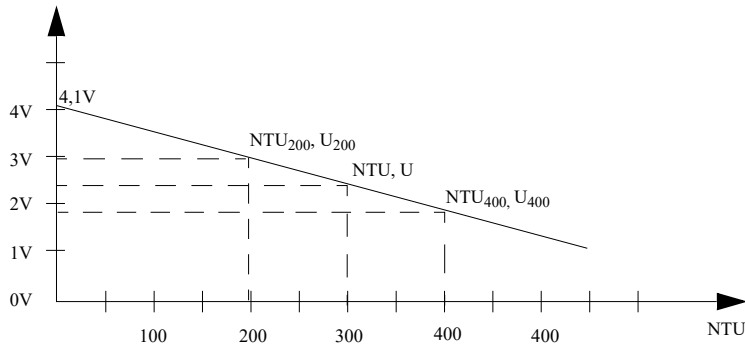
---

107. Beskrivelse av hvordan man lager en standard oppløsning på 400 NTU. <https://www.boquinstrument.com/how-to-make-turbidity-standard-solution>

108. <https://no.vwr.com/store/product/31990339/turbidity-standards-formazin>



Spenningen vil avta med økende NTU-verdier. Figuren under gir en grafisk framstilling av denne sammenhengen.



Vi kan da sette opp en to-punktsligning:

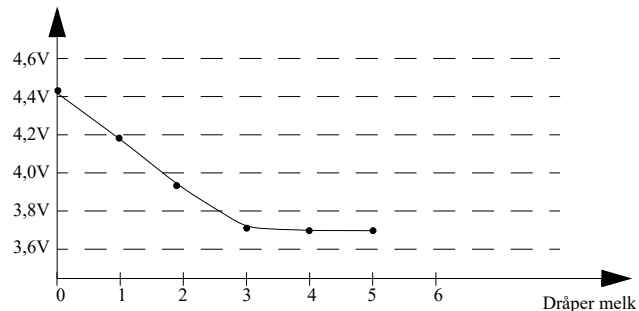
$$NTU = (NTU_{400} - NTU_{200}) / (U_{400} - U_{200}) (U - 4,1 \text{ [V]}) \quad (6.17)$$

NTU er uten benevnning.

### Enkel innledende test uten kalibrering

Som en enkel innledende test kan man bruke litt lettmeik for å lage en blanding som gir spredning. Vi starter med destillert vann og fortsetter med en dråpe melk, og øker så på med en og en dråpe for å se om vi får en rimelig lineær kurve. Vi har valgt å midle måleverdien over 100 målinger og vi bruker et 3D-printet kyvettekammeret som omtalt i neste avsnitt, hvilket vil stabilisere målingene litt. De er imidlertid ganske stabile i utgangspunktet. Vi bruker programmet: Turbidimeter-SEN0189.ino til denne målingen.

Dråper	Spenning
0	4,44 V
1	4,19 V
2	3,94 V
3	3,72 V
4	3,69 V
5	3,70 V



Vi ser at kurven synes å flate ut ved ca. 3 dråper. Hvilken turbiditet dette er vet vi ikke før vi har kalibrert sensoren vår. Siden kyvettekammeret er ganske lite så vil en dråpe utgjøre et ganske stort sprang i turbiditet. Vi legger også merke til at spenningen ved rent vann er høyere enn 4,1 V.

La oss se hvordan vi kan kalibrere sensoren vår slik at vi får verdien ut i NTU.



## Metode for kalibrering

Vi bruker følgende metode for å kalibere og måle med sensoren:

1. Vi har valgt å 3D-printe en sort ugjenomsiktig boks for å hindre lekkasje av lys. Boksen passer akkurat sensoren som vist på figuren til høyre.
2. Fyll boksen med kalibreringsvæske med NTU 400. Mål spenningen og noter verdien.
3. Fyll boksen med kalibreringsvæske med NTU 200. Mål spenningen og noter verdien.
4. Skriv de fire verdiene (NTU og spenningene) inn i programmet og gjør målinger på NTU 200 og NTU 400 og sjekk at verdiene blir som under kalibreringen.
5. Fyll boksen med destillert vann og noter at måleverdien skal være ca. 4,1 V.
6. Mål en "ukjent" prøve å se om det virker rimelig.



### Uttyning av kalibreringsvæske

Vi kan kjøpe kalibreringsvæske (Formazin) med forskjellig turbiditet. 500 og 1000 er ikke uvanlig. Ved å tilsette destillert vann vil turbiditeten reduseres.

La oss anta at vi har en flaske med NTU lik 500 og ønsker å tilsette vann slik at vi får en NTU på henholdsvis 400 og 200. La oss si at vi ønsker en sluttmengde på  $T_{mL} = 50$  mL. Spørsmålet er da hvor mye vi må tilsette av Fermazin ( $F_{mL}$ ) og destillert vann ( $V_{mL}$ ).

Vi kan sette opp følgende sammenheng:

$$\frac{\text{Ønsket volum fortdynnet standard} \times \text{Ønsket NTU-verdi}}{\text{NTU for standard som skal blandes ut}} = \text{Volum tilsatt av opprinnelig standard}$$

I vårt tilfelle blir beregningen for å lage en standard på 400 NTU lik:

$$\frac{50\text{mL} \times 400}{500} = 40\text{mL}$$

Dvs. at oppløsningen må inneholde 40 mL av standarden på 500 NTU og 10 mL destillert vann.

Tilsvarende kan vi gjøre følgende beregning for å lage en standard på 200 NTU:

$$\frac{50\text{mL} \times 200}{500} = 20\text{mL}$$

Dvs. at oppløsningen må inneholde 20 mL av standarden på 500 NTU for så å tilsette 30 mL destillert vann.



## Kalibrerte målinger

Instrumentet er kalibrert med NTU 200 og NTU 400, i tillegg til at det er gjort målinger ved rent vann. Følgende resultater er oppnådd:

Kalibrert oppløsninger [NTU]	Spenning [V]
Under kalibrering	
Kalibereringsmedium	Spenning
0 (rent vann)	4,36 V
200 NTU	4,14
400 NTU	3,97
Måling i etterkant	
Prøve	Målt verdi (NTU)
springvann	0 +/- 5
200 NTU	250 +/-3
400 NTU	460 +/-3
500 NTU	540 +/-3

## Eksempelprogram for måling av spenningen (Turbidimeter-SEN0189.ino)

```
void setup() {
  Serial.begin(9600);          //Baud rate: 9600
  analogReadResolution(12); // Sett oppløsningen til AD-konverteren 12 bit
}

void loop() {
  int sensorValue = 0;
  float voltage = 0;

  for (int i=0; i<100; i++)
  {
    sensorValue = analogRead(A0); // Les av analog inngang fra pinne A0:
    voltage = voltage + 2 * sensorValue * (3.3 / 4096.0);
  }

  voltage = voltage/100;

  // Konverterer den analoge avlesningen (som går fra 0-4095) til en spenning (0
- 4.1 +/- 0,3 V ved 0 NTU (Rent vann)):
  Serial.println(voltage); // Skriv ut den avleste verdien:
  delay(500);
}
```



```
}
```

### Eksempelprogram for deteksjon av passering av et terskelnivå

```
int ledPin = LED_BUILTIN;          // Connect an LED on pin 13, or use the onboard one
int sensor_in = 0;                // Connect turbidity sensor to Digital Pin 2

void setup(){
  pinMode(ledPin, OUTPUT);        // Set ledPin to output mode
  pinMode(sensor_in, INPUT);      //Set the turbidity sensor pin to input mode
}

void loop(){
  if(digitalRead(sensor_in)==LOW){ //read sensor signal
    digitalWrite(ledPin, HIGH);   // if sensor is LOW, then turn on
  }
  else{
    digitalWrite(ledPin, LOW);    // if sensor is HIGH, then turn off the led
  }
}

// Deklarasjon av variabler
float NTU = 0; // Målt og beregnet NTU-verdi
float NTU_200 = 0; // Kalibrert NTU-verdi 200 NTU.
float NTU_400 = 0; // Kalibrert NTU-verdi 400 NTU.
float UD_200 = 0; // Målt digital spenningsverdi ved NTU-verdi 200
float UD_400 = 0; // Målt digital spenningsverdi ved NTU-verdi 400

void setup() {
  Serial.begin(115200);          //Baud rate: 115 200
  analogReadResolution(12);     // Sett oppløsningen til AD-konverteren 12 bit
}
```

### Eksempelprogram for kalibrering og måling (Turbidimeter\_SEN0189\_Kal-1)

Programmet kan brukes til å avlese verdier for kalibrering slik at det kan gjøre en lineær beregning av NTU i henhold til de standarder som er brukt.

```
// Turbidimeter_SEN0189_Kal-1
// Programmet måler og beregner NTU verdien til en væske på bakgrunn av
// kalibreringsverdier for NTU-verdi 200 og NTU-verdi 400
// Nils Kr. Rossing 01.06.23

// Deklarasjon av variabler
```



```
float NTU      = 0;    // Målt og beregnet NTU-verdi
float NTU_200  = 200;  // Kalibrert NTU-verdi 200 NTU.
float NTU_400  = 400;  // Kalibrert NTU-verdi 400 NTU.
float U_0      = 4.36; // Målt spenningsverdi ved NTU-verdi 400
float U_200    = 4.14; // Målt spenningsverdi ved NTU-verdi 200
float U_400    = 3.97; // Målt spenningsverdi ved NTU-verdi 400

void setup() {
  Serial.begin(115200);    //Baud rate: 115 200
  analogReadResolution(12); // Sett oppløsningen til AD-konverteren 12 bit
}

void loop() {
  long sensorValue = 0; // Les av analog inngang fra pinne A0:
  float UD_mid     = 0; // Middelerverdi av målt spenning
  float UD_mid_volt = 0; // Middelerverdi av målt spenning

  for(int i=0; i<100; i++)
  {
    sensorValue = sensorValue + analogRead(A0);
  }

  UD_mid = 2.0 * float(sensorValue/100); // Finner middelerverdien
  UD_mid_volt = UD_mid * (3.3 / 4096.0); //
  // Konverterer den analoge avlesningen (som går fra 0-4095) til en spenning (0
  - 4.1 +/- 0,3 V ved 0 NTU (Rent vann)):

  Serial.print("Maalt midlere spenning: "); // Skriv ut den avleste verdien for
  kalibrering:
  Serial.println(UD_mid_volt);

  // Beregner NTU-verdien
  NTU = (NTU_400 - NTU_200)/(U_400 - U_200)*(UD_mid_volt-U_0);
  Serial.print("Maalt NTU: "); // Skriv ut den avleste verdien:
  Serial.println(NTU,0);
  delay(500);
}
```



## 6.7 Sensor for måling av pH (SEN-0161)<sup>109</sup>

pH-sensoren SEN-0161 (bildet under til venstre) er spesielt tilpasset bruk med Arduino og gir en nøyaktighet på  $\pm 0,1$  pH ved  $25^{\circ}\text{C}$ . Det forutsettes da at instrumentet er kalibrert. Arbeidsspenningen er 5,0V. Sensoren er en labororiesensor og er ikke beregnet for å stå i en oppløsning over lang tid. Til dette formålet kan man utstyre sensoren med en mer robust probe (FIT-0348<sup>110</sup>), se bildet under til høyre. Det finnes også en versjon 2 (SEN-0161-V2<sup>111</sup>) av denne proben som også fungerer for arbeidsspenninger ned til 3,3 V. Denne anbefaler imidlertid bruk av et bibliotek som ikke er kompatibel med MKR NB 1500.



pH-sensoren måler surhetsgraden av væsken på en skala fra 1 – 14. Verdier fra 1 – 7 er sure og fra 7 – 14 basiske. For verdien pH 7,0 sier vi at væsken er *nøytral*, hverken sur eller basisk. I virkeligheten er pH et mål for konsentrasjonen av frie  $\text{H}^+$  ioner i væsken.

pH er avhengig av temperaturen. For å få sammenlignbare data bør man derfor gjøre målingene ved en standardisert temperatur, det vanlige er  $25^{\circ}\text{C}$ .

### Måling av pH

Før vi ser på hvordan vi kan måle styrken på syrer og baser, så må vi vite litt om hva som karakteriserer slike. Syrer og baser er kort fortalt stoffer som når de løses opp i vann, danner ioner. Syrer danner  $\text{H}^+$  ioner, mens baser danner  $\text{OH}^-$  ioner. Ioner er atomer eller molekyler som har avgitt elektroner (f.eks.  $\text{H}^+$ ) eller mottatt elektroner (f.eks.  $\text{OH}^-$ ) og er dermed ladet.

pH er en forkortelse for *potens of Hydrogen* (eller *power of Hydrogen*). Hvis konsentrasjonen av  $\text{H}^+$ -ioner i en løsning er 0,01 mol pr. liter, er pH til løsningen 2,0. Sammenhengen er at tallet 0,01 kan skrives som potensen  $10^{-2}$ . Dersom konsentrasjonen av  $\text{H}^+$ -ioner i en løsning er 0,0001 mol pr. liter, dvs. at konsentrasjonen er langt lavere enn i forrige eksempel, så sier vi at pH i løsningen er 4,0, fordi 0,0001 kan skrives som  $10^{-4}$ .<sup>112</sup> Vi ser at pH angis som  $-\log(10^{-\text{pH}}) = \text{pH}$ .

109.<https://www.dfrobot.com/product-1025.html>

110.<https://www.dfrobot.com/product-1074.html>

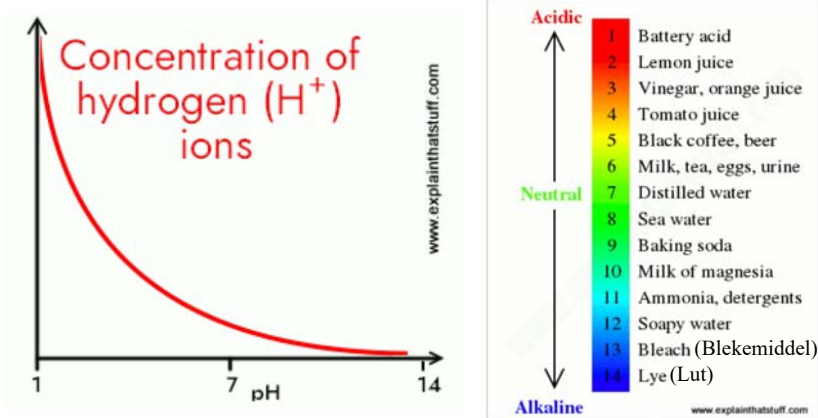
111.<https://www.dfrobot.com/product-1782.html>

112. pH ble innført av den danske kjemikeren Søren Peter Lauritz Sørensen i 1909 da han var leder for den kjemiske avdeling ved Carlsberg Laboratorium i København. Den gang var regning med logaritmer allmennkunnskap. I dag har logaritmeregning gått i glemmeboken, og pH ville neppe ha blitt innført i dag. I stedet ville man ha brukt dekadiske forstavelser (SI-prefikser). Da kan 0,0000323 mol/L skrives 32,3  $\mu\text{mol/L}$  (mikromol pr. liter). (Store Norske Leksikon - <https://snl.no/pH>)





Til venstre på figuren under ser vi hvordan antallet  $H^+$ -ioner avtar logaritmisk med økende pH. Til høyre på figuren ser vi typiske eksempler på kjente stoffer med forskjellig pH<sup>113</sup>.



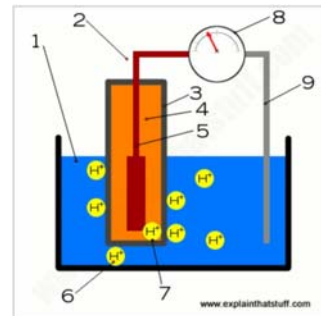
Når vi skal måle pH-verdien i en væske, utnytter vi kunnskapen om at jo syrligere væsken er jo flere  $H^+$  ioner inneholder den.

### Virkemåte

La oss studere hvordan pH-proben virker.

Vi vet nå at i en syrlig væske er det langt flere  $H^+$ -ioner enn i en mindre syrlig eller basisk oppløsning. Dette utnytter vi ved måling av pH. Vi kan sammenligne væsken med elektrolytten i et batteri. Proben som stikkes ned i væsken, kan sammenlignes med polene på et batteri, og spenning som oppstår på proben vil være et mål for antallet  $H^+$ -ioner. Jo flere  $H^+$ -ioner jo høyere spenning.

La oss se nærmere på oppbyggingen av proben (ref. figuren til høyre):



1. ... er væsken eller oppløsningen vi skal måle pH i.
2. ... er en glass-elektrode som består av ...
3. ... en tynn beholder laget av silikatglass iblandet metallsalter, og som er fylt med ...
4. ... en kaliumkloridoppløsning med ...
5. ... en sølv- eller sølvklorid elektrode inne i kaliumkloridoppløsningen
6. ... er  $H^+$ -ioner i væsken som vi ønsker å måle pH i. Disse ionene virker sammen med overflaten av glasselektroden
7. ... er  $H^+$ -ioner kaliumkloridoppløsningen som virker sammen med innsiden av glasset

113. <https://www.explainthatstuff.com/how-ph-meters-work.html>



8. ... er voltmeteret som måler spenningsforskjellen mellom inn- og utsiden av glasset og som er et mål for pH-verdien
9. ... er en passiv elektrode som fungerer som referanse i målingen og som slutter kretsen

Kaliumkloridoppløsningen inne i glasselektroden er nøytral med en pH lik 7,0. Dersom vi f.eks. antar at væsken vi ønsker å bestemme pH til er sur, så vil det være en høyere tetthet av  $H^+$ -ioner på utsiden av glasselektroden enn inni, og omvendt om væsken er basisk. Det er denne forskjellen i  $H^+$ -konsentrasjon som gir potensialforskjellen vi måler med voltmeteret. Det er en lineær sammenheng mellom forskjellen i pH mellom væsken og den innvendige kaliumkloridoppløsningen og potensialforskjellen.

På utsiden av glasselektroden vil det skje en utveksling av  $H^+$ -ioner i væsken og metall-ionene i glasset. Tilsvarende vil også skje på innsiden av glasselektroden. Det er denne utvekslingen som er den egentlige årsaken til potensialforskjellen.

VOLTAGE (mV)	pH value	VOLTAGE (mV)	pH value
414.12	0.00	-414.12	14.00
354.96	1.00	-354.96	13.00
295.80	2.00	-295.80	12.00
236.64	3.00	-236.64	11.00
177.48	4.00	-177.48	10.00
118.32	5.00	-118.32	9.00
59.16	6.00	-59.16	8.00
0.00	7.00	0.00	7.00

Tabellen over viser hvordan denne potensialforskjellen endrer seg med forskjellen i pH-verdi.

Vi må også være klar over at utvekslingen av ioner er temperaturavhengig, noe det er viktig å kompensere for.

### Framstilling av en 3 mol kaliumklorid oppløsning (KCl)

Vi vet at en 1 mol oppløsning av et stoff er molekylvekta av stoffet i gram oppløst i 1 liter destillert vann. Molekylvekta av  $K + Cl = 39,1 + 35,5 = 74,6$  g. 1 mol = 74,6 g i 1 liter vann. Eller 3 mol = 223,8 g i 1 liter vann. Dvs. 22,4 g i 1 dL destillert vann.



## Oppkobling

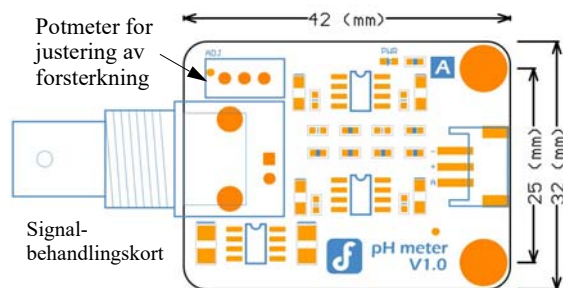
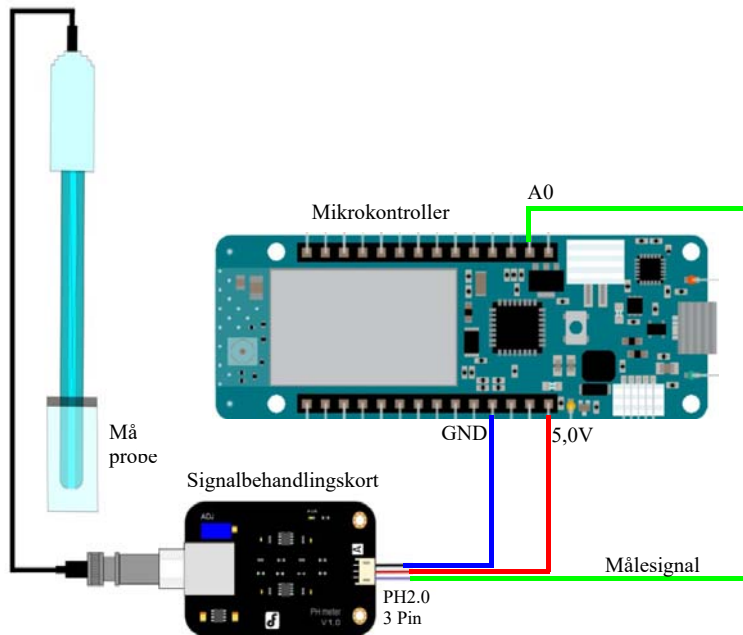
Selve proben kobles til signalkortet (signalkonverteringskortet) med en BNC-plugg. Legg merke til at utgangssignalet fra signalkortet føres inn på en analog inngang hos mikrokontrolleren (her A0).

### Pass på følgende:

- Unngå at signalbehandlingskortet blir fuktig. Et fuktig kort vil endre målebetingelsene og gi avvik i målingene.
- Foran i proben sitter en liten glasskule som ikke må skades, noe som lett kan skje dersom kula støtes mot harde overflater.
- Pass på at proben ikke er tilkoblet signalbehandlingskortet over lengre tid uten at det står spinning på kortet.
- Når sensoren ikke brukes skal proben beskyttes av en hette fylt med 3,3 mol/L KCl

### Spesifikasjoner:

- Forsyningsspenning: 5.0V
- Probekontakt: BNC
- Signalkontakt: PH2.0 – 3P
- Målenøyaktighet:  $\pm 0.1 @ 25^{\circ}\text{C}$
- Signalkortets størrelse: 42 mm\*32 mm
- Probetype: For laboratoriebruk. *Dette er viktig å merke seg. Den er ikke beregnet for å stå ute i felt over lengre tid.*
- Måleområde: pH 0~14
- Temperaturområde: 0~60°C
- Responstid: < 1 min.
- Kabellengde fra signalkort til probe: 66 cm





- Signalkort inneholder et potensiometer for å justere forsterkningen

## Kalibrering og bruk

Det er viktig å skylle proben mellom målinger gjort i ulike væsker, slik at man unngår å forurense prøvene.

Følgende trengs for kalibrering og måling:

- 1 x Mikrokontroller
- 1 x pH Signal Conversion Board V1
- 1 x pH Probe
- 1 x Standard Buffer-løsning på pH 7,0
- 1 x Standard Buffer-løsning på pH 4,0 evt. pH 9,18
- 1 x Gravity 3pin Sensorkabel
- 1 x Testopløsning

Koble opp som vist på figuren over og fjern hetta foran på proben.

Det er viktig å bruke en stabil spenningskilde på signalkortet. Normalt vil man hente den fra Arduino-kortet, men denne kan være beheftet med støy fra mikrokontrolleren, eller som hos MKR NB 1500 så vil den ikke normalt være tilgjengelig med mindre man lager den selv siden MKR benytter 3,3 V. En power booster genererer en spenning på 5,2 V. Her bør man tenke seg om.

Ved kalibrering bør man ha to standard bufferløsninger tilgjengelig. For måling av sure løsninger bør man ha en buffer lik pH 4,0, ved måling av alkaliske løsninger er det anbefalt at man har en bufferløsning på pH 9,18.

1. Før kalibrering vask proben med destillert vann (gjerne ionebyttet), og tørk av etter vasking.
2. Koble opp proben som vist på oppkoblingen foran. Velg en passende analog inngang for tilkobling av signalet (A0 i eksempelet).
3. Last opp eksempelkode som vist under. Eksempelet bør justeres mht. at vi skal bruke MKR NB 1500 (Velg standard LEDpin for lysdioden på MKR-kortet: LED\_BUILTIN).
4. Stikk proben ned i en standard bufferløsning med pH 7,0. Evt. ha den i hetta foran på proben. Alternativt kortslutt probeutgangen (BNC-kontakten).
5. Åpne monitoren og les av målt pH-verdi. Denne skal være innenfor  $\text{pH } 7,0 \pm 0,3$ . La oss anta at den viser pH 6,88.
6. Dvs. vi registrerer et avvik på  $-0,12$ .
7. For å korrigere for dette avviket så setter vi verdien 0,12 inn i programmet der det står:

```
#define Offset 0.12 //Kompenser for avvik på 0,12
```



- Tørk av proben og stikk den ned i standard bufferløsningen med pH 4,0 og les av verdien med programmet.
- Juster bort avviket fra pH 4,0 med potensiometeret på signalkortet. Vent til resultatet er stabilisert på rundt pH 4,0. Du er nå klar til å måle pH i syrlige oppløsninger.
- Dersom løsningen er basisk, stikker vi proben ned i en standard bufferløsning på pH 9,18 og juster potensiometeret slik at den avleste verdien er ca. pH 9,18.



### Praktisk gjennomføring av kalibrering

Til kalibreringen trenger vi noen standardiserte bufferløsninger av pH4, pH7 og pH 9,18. De to første kan det være mest aktuelt å bruke dersom vi skal måle noe surt, de to siste når vi skal måle noe som er basisk som f.eks. sjøvann som har en pH-verdi på ca. 8.

Normalt vil man tenke at rent drikkevann har en pH på ca. 7,0, hvilket er riktig. Det er imidlertid stor forskjell på drikkevann med pH 7 og en bufferløsning på pH 7,0. Som navnet sier så vil en buffer være ganske robust mot forurensning av ioner. En prøve med drikkevann vil lett la seg forurense av f.eks. en bufferløsning på pH4, da denne inneholder over 1000 ganger så mange  $H^+$  ioner som rent drikkevann. En bufferløsning på pH7 vil til en viss tåle å forurenset slik at den vil beholde sin pH-verdi, den kompenserer for mindre forurensninger.

### Oppskrift på bufferløsninger med ulik pH:

La oss se nærmere på hvordan vi kan lage en bufferløsning med definert pH-verdi.

En buffer er i denne sammenheng en oppløsning med en bestemt pH, og som er robust mht. pH-verdi selv om den "forurennes" av tilsetninger med en annen pH. Buffere med definerte pH-verdier kan kjøpes, men kan også lages om man har de rette ingrediensene.

#### Fosfatbuffere pH 4 og 7

Følgende to fosfat-sitrat-buffere er egnet for kalibrering av pH-sensoren for pH-verdier under 7 (sure):

Ingrediens → pH ↓	$Na_2HPO_4$ , 0,2M (2,84 masse%)	$C_6H_8O_7$ (Sitronsyre), 0,1M (1,92 masse%)
4	38,55 ml	61,45 ml
7	82,35 ml	17,65 ml

#### Fosfatbuffere pH 7 - 11

Følgende fosfat-buffere er egnet for kalibrering av pH-sensoren for pH-verdier over 7 (basiske):

- 0.1M  $Na_2HPO_4$  (14.2g / l)
- 0.1M HCl
- 0.1M NaOH

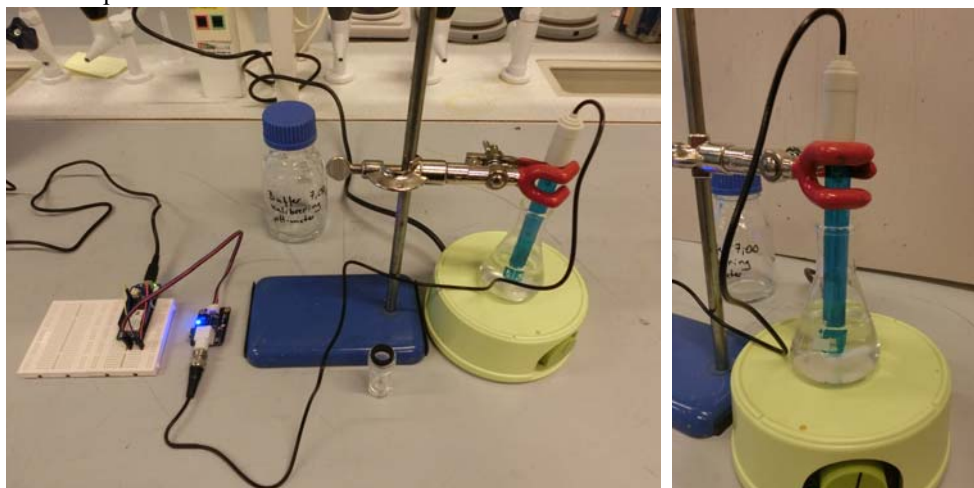


Ulike pH-verdier oppnås ved å blande disse i de mengder som vist i tabellen under

pH	Vol. hydrogenfosfat	Vol. 0.1M HCl	Vol. 0.1M NaOH
7	756.0 ml	244 ml	
8	955.1 ml	44.9 ml	
9	955.0 ml	45.0 ml	
10	966.4 ml		33.6
11	965.3 ml		34.7

*Gjennomføring av kalibrering:*

1. Ta en kolbe og en magnetrører og fyll den med bufferløsning med pH7. Sett den til røring.
2. Monter pH-sensoren i et laboratoriestativ og la den stå neddyppet i bufferen over noe tid slik at måleverdien har stabilisert seg. La sensoren være oppkoblet til mikrokontrolleren og observer pH-verdien.



3. Les av avviket fra pH7 og legg dette inn i programmet  
`#define Offset 0.0 //Kompenser for avviket`
4. Dersom målingene viser verdier over 7,0 så legges offset inn med negativt fortegn  
Dersom den er under pH7,0 så legges korreksjonen inn med positivt fortegn
5. Tøm ut bufferen og skyll kolben med litt av den nye bufferen som skal brukes i kalibreringen, f.eks. pH 9,18. Bruk den samme bufferen og skyll proben, og tøm ut.
6. Fyll opp med buffer med pH 9,18, rør godt og stikk proben ned i kolben og vent til måleverdien har stabilisert seg.
7. Korriger verdien med å skru på forsterkningen (potensiometeret).

Dersom det er store avvik, kan det være nødvendig å gå flere runder ved at man går frem og tilbake mellom bufferne pH7 og pH 9,18. Selv om det anbefales å bruke 9,18 så kan men like godt benytte pH9,0, hensikten er at man kalibrerer proben i det området der man ønsker å bruke den, for oss nær pH8, sjøvann. Vi er nå klare til å måle i sjøvann.



## Måling av PH i sjøvann

Med denne prosedyren målte vi sjøvann til **pH8,12**. Målingen er gjort ved 23,5°C.

### pH som funksjon av temperaturen

Når temperaturen i en løsning stiger, stiger de molekylære vibrasjonene i løsningen, noe som resulterer i ionisering og dannelse av  $H^+$ -ioner. Flere  $H^+$  ioner fører til surere oppførsel. På grunn av temperaturendringer endres pH-verdien til løsningen slik at pH synker ved økende temperatur.

Det er viktig å være klar over at enhver løsning vil gjennomgå en endring i pH-verdien som funksjon av temperaturen. En forskjell i pH-målinger ved forskjellige temperaturer er imidlertid IKKE en feil! Det nye pH-nivået forteller ganske enkelt om den sanne pH-verdien for den løsningen ved den spesifikke temperaturen, som vi ser av tabellen<sup>114</sup> under.

T (°C)	$K_w$ (mol <sup>2</sup> dm <sup>6</sup> )	pH
0	$0.114 \times 10^{14}$	7.47
25	$1.008 \times 10^{14}$	7.00
50	$5.476 \times 10^{14}$	6.63
100	$51.3 \times 10^{14}$	6.14

Variasjonen med temperaturen er også avhengig av pH-nivået. Påvirkningen av temperaturen er mindre ved lave pH-verdier (sure) enn ved høye pH-verdier (basiske). Tabellen<sup>115</sup> til høyre viser dette.

	0°C	25°C	50°C
Acid	2.01	2.00	2.00
Neutral (Water)	7.47	7.00	6.63
Basic	13.80	12.83	12.15

Dersom vi ønsker å inkludere en temperaturkorreksjon i programmet, så vil det være en fordel om sammenhengen mellom pH og temperatur kan uttrykkes matematisk.

Siden variasjonen av pH i sjøvann av andre årsaker enn temperatur er små og ganske langsomme, kan det være aktuelt å foreta slike korreksjoner. På nettsiden:

<https://www.hamzasreef.com/Contents/Calculators/PhTempCorrection.php>

... finner vi en kalkulator som kan vise oss hvilket variasjonsområde det her er snakk om.

<p>pH Reading: <input type="text" value="8.1"/> pH Units</p> <p>Solution Temperature: <input type="text" value="20"/> Celsius/Centigrade ▾</p>	<p><input type="button" value="Calculate"/></p> <p>Corrected pH: <input type="text" value="8.1188"/> pH Units</p>
--	---

114. <https://www.westlab.com/blog/2017/11/15/how-does-temperature-affect-ph>

115. <https://www.westlab.com/blog/2017/11/15/how-does-temperature-affect-ph>





Kalkulatoren beregner pH ved 25°C når målingene er gjort ved lavere temperaturer, hvilket vil være aktuelt for vårt tilfelle. Som det framgår av tabellen ser vi at en måling nær en pH på pH8,1 så vil variasjonen være på 0,08 over et temperaturområde på 25°C.

pH-målt ved ...	Temperatur	Omregnet til pH @ 25°C
pH 8,1	0°C	pH 8,20
pH 8,1	2°C	pH 8,19
pH 8,1	5°C	pH 8,18
pH 8,1	10°C	pH 8,16
pH 8,1	15°C	pH 8,14
pH 8,1	20°C	pH 8,12

### Slik virker måleprogrammet

Som omtalt foran utføres målinger av potensialforskjellen over glasselektroden. Det gjøres et sett med inntil 40 målinger. Deretter beregnes et gjennomsnitt av de målte forskjellene i potensialverdier. Den midlere spenningsverdien ( $\overline{V}_{pH}$ ) brukes så til å beregne en verdi for pH etter følgende formel:

$$pH = 3,5 * \overline{V}_{pH} + \text{Offset} \quad (6.18)$$

Der

$\overline{V}_{pH}$  er den midlede potensialforskjellen målt mellom elektrodene

pH er den beregnede pH-verdien

Offset er avviket registrert ved kalibrering med nøytral bufferløsning

3,5 er den beregnede stigningskoeffisienten mellom potensialforskjell og pH under forutsetning av at forsterkningen er justert riktig under kalibreringen

Vi legger spesielt merke til at det ikke er tatt hensyn til at pH endres som funksjon av temperaturen. Dette må man evt. legge inn selv, eller sørge for å utføre kalibreringen ved den aktuelle temperaturen.

### Eksempelkode

Eksempelkoden finnes i vedlegg: G.5 side 279

### Oppbevaring

Når sensoren ikke er i bruk skal toppen settes på proben. Toppene fylles med 3,3 mol/L KCL slik at en hindrer at glasskula i tuppen av proben får anledning til å tørke ut.

Unngå langvarig neddykking i destillert vann, proteinholdige og "acid fluoride"-holdige væsker og silisium baserte oljer.

For flere råd om vedlikehold av proben se [https://wiki.dfrobot.com/Gravity\\_Analog\\_pH\\_-\\_Sensor\\_Meter\\_Kit\\_V2\\_SKU\\_SEN0161-V2#More\\_Documents](https://wiki.dfrobot.com/Gravity_Analog_pH_-_Sensor_Meter_Kit_V2_SKU_SEN0161-V2#More_Documents)



## 6.8 Sensor for måling av ORP (SEN-0165)<sup>116</sup>

ORP (Oksidasjon-Reduksjons-Potensial) er et mål for evnen til oksidasjon og reduksjon i en oppløsningen. I motsetning til en pH-måling som følger en logaritmisk kurve og derfor krever flere kalibreringsjusteringer, følger ORP et lineært forløp. *ORP har vist seg å være en pålitelig metode for å måle vannkvalitet og gir en enkelt måleverdi uavhengig av hvilket produkt eller desinfiseringsmiddel som er brukt, den er også uavhengig av varierende feltforhold.*

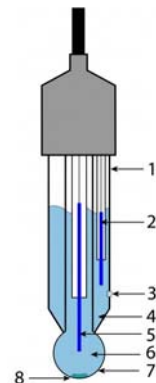


Firmaet LanLang skriver på hjemmesiden sin:<sup>117</sup>

Oksidasjonsreduksjonspotensialet (ORP) måler en innsjø eller elves evne til å rense seg selv eller bryte ned avfallsprodukter, for eksempel forurensninger og døde planter og dyr. Når ORP-verdien er høy, er det mye oksygen tilstede i vannet. Dette betyr at bakterier som bryter ned døde vev og forurensninger, kan fungere mer effektivt. Generelt er jo høyere ORP verdien, desto sunnere er sjøen eller elven. Men selv i friske innsjøer og elver er det mindre oksygen (og dermed lavere ORP-verdier) når du kommer nærmere bunnsedimentene (leire, se bildet under en sjøbunn). Dette skyldes at det er mange bakterier som arbeider hardt i sedimentene for å dekomponere dødt vev, og de bruker mye av det tilgjengelige oksygen. Faktisk forsvinner oksygen veldig raskt i bunnslammet (ofte innenfor en centimeter eller to), og ORP faller raskt. ORP måles i tillegg til oppløst oksygen fordi ORP kan gi forskere tilleggsinformasjon om vannkvaliteten og forurensningsgraden, hvis den er til stede. Det er også andre elementer som kan fungere som oksygen (når det gjelder kjemi) og bidrar til økt ORP.

Måleenheten for ORP er i mV. Hvis oksidasjons-reduksjonspotensialet har en høy verdi, er kjemisk oksidasjon sterk, mens hvis potensialet er lavere, er oksidasjonen svakere. *Et positivt potensial betyr at løsningen viser en viss grad av oksidasjon, mens et negativt potensial betyr at løsningen viser en viss grad av reduksjon*<sup>118</sup>.

Selve måleelementet er en ORP-kompositelektrode laget av gull eller platina, som brukes til å måle oksidasjons-reduksjons-potensialet til oppløsningen.



116. [https://wiki.dfrobot.com/Analog\\_ORP\\_Meter\\_SKU\\_SEN0165\\_](https://wiki.dfrobot.com/Analog_ORP_Meter_SKU_SEN0165_)

117. <http://no.waterfiltermediasupplier.com/info/what-is-orp-36070712.html>

118. <https://www.emerson.com/documents/automation/application-data-sheet-fundamentals-of-orp-measurement-rosemount-en-68438.pdf>



## Sensorens virkemåte

Sensoren virker omtrent på samme måte som en pH-sensor, og består av to elektroder, en måle- og en referanseelektrode, som er plassert i hver sine elektrolytter som vist på figuren til høyre<sup>119</sup>. Til sammen danner disse et lite “batteri”. Spenningen til dette batteriet vil endre seg avhengig av egenskapene til væsken som hele proben dyppes ned i. På grunnlag av denne spenningen bestemmes ORP-verdien. Med henvisning til figuren til høyre:

1. ... er ikke-ledende glass eller plastikk som er ment å holde proben
2. ... referanseelektroden, vanligvis framstilt av det samme materialet som hovedelektroden.
3. ... er en overgangssone i glasset mellom oppløsningen som skal måles og området som inneholder referanseelektroden.
4. ... er referanseoppløsningen som inneholder referanseelektroden og som ofte er 0,1 mol/L av KCl.
5. ... er måleelektroden laget av sølvklorid eller tidligere også av kvikksølvklorid.
6. ... er oppløsningen rundt måleelektroden som er en buffer av 0,1 mol/L KCl.
7. ... er glasset omkring måleelektroden som er av en spesiell type.
8. Dersom måleelektroden er laget av sølvklorid, kan det lett oppstå litt bunnfall av AgCl i bunnen av glasskulen.

Som nevnt er pH- og ORP-sensorene ganske like, men elektrodene står gjerne i forskjellig elektrolytt (se også 6.7 side 202). Mens pH-sensoren er logaritmisk er ORP-sensoren lineær.

## Spesifikasjoner:

- Spenningsforsyning: +5.00V
- Måleområde: -2000mV~2000mV
- Operativ temperatur: 5~70°C (Denne kan være kritisk i havvann om vinteren)

---

119.[https://www.phidgets.com/docs/PH/ORP\\_Sensor\\_Primer#How\\_they\\_work](https://www.phidgets.com/docs/PH/ORP_Sensor_Primer#How_they_work)

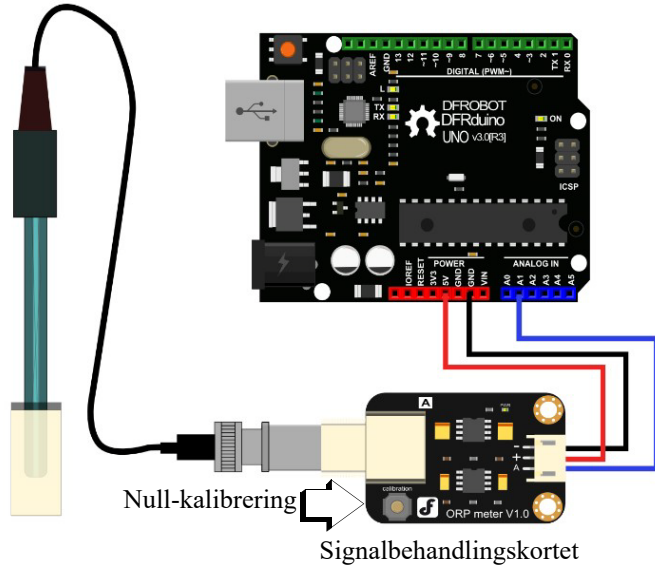


- Nøyaktighet:  $\pm 10\text{mV}$  ( $25^\circ\text{C}$ )
- Responstid:  $\leq 20\text{sec}$
- ORP Probe med BNC kontakt
- PH2.0 Interface (3 pin)
- Knapp for null-kalibrering

### Oppkobling

Eksempelet under viser hvordan proben kan kobles opp til en Arduino UNO.

Signalbehandlingskortet har en knapp for null-kalibrering.



### Måleverdier i standardløsning med forskjellige temperaturer

Tabellen til høyre viser hvordan den målte spenningen varierer med temperaturen ved bruk av en standard løsning på  $3,5\text{mol/L KCl}$ .

$222\text{mV} \pm 15\text{mV}$  ( $25^\circ\text{C}$ )  
( $3,5\text{mol/L KCl}$ )

### Framstilling av standard løsning

Vi vet at en 1 mol oppløsning av et stoff er molekylvekten av stoffet i gram oppløst i 1 liter destillert vann. Molekylvekten av  $\text{K} + \text{Cl} = 39,1 + 35,5 = 74,6\text{ g}$ . 1 mol =  $74,6\text{ g}$  i 1 liter vann. Eller  $3,5\text{ mol} = 261,1\text{ g}$  i 1 liter vann. Dvs.  $26,1\text{ g}$  i 1 dL destillert vann.

$^\circ\text{C}$	mV	$^\circ\text{C}$	mV
10	242	30	215
15	235	35	209
20	227	38	205
25	222	40	201

### Brukerveiledning

Vær oppmerksom på følgende:

- Det anbefales på det sterkeste å bruke en stabil spenningskilde på signalkortet. Det antas at variasjoner i forsyningsspenningen gir seg direkte utslag i levert spenning på signalutgangen.
- Normalt kan man bruke sensoren uten kalibrering. Dersom det er mistanke om avvik bør man bruke standard løsningen for å sjekke at spenningen er som forventet. Husk å kompensere for temperatur.
- Ved bruk bør proben vaskes med ionebyttet vann (deionized).
- **NB!** Ikke trykk knappen for kalibrering når proben er tilkoblet, da dette kan ødelegge proben.



1. Koble signalkortet opp til Arduino-kortet som vist på figuren over. **NB!** Vent med å koble til selve proben. Når signalkortet har spenning skal det lyse en blå diode. Signalutgangen kobles til A1. Bruker man en annen analog inngang må dette endres tilsvarende i programmet.
2. Kompiler og last opp eksempelkode til mikrokontrolleren. Legg spesielt merke til kodelinjen:

```
#define OFFSET 0
```

3. Åpne monitoren i Arduino IDE og du vil se den målte ORP-verdien bli skrevet ut. Trykk på kalibreringsknappen på signalkortet. Kalibreringsknappen legger inngangen fra måleproben til jord. Dette er årsaken til at dette ikke bør gjøres mens proben er tilkoblet. Registrer utskriften i monitoren og bruk denne til å korrigere Offset i programmet. Viser monitoren en ORP på 8mV så går man inn i programmet og endrer:

```
#define OFFSET 8
```

Kompiler og last opp programmet på nytt etter at du har gjort endringen.

4. Nå er sensoren kalibrert og vi kan koble til proben ved hjelp av BNC-pluggen og lese av ORP-verdien i serie monitoren. Husk å sett monitoren til 9600 Baud som er den datahastigheten mikrokontroller sender på.

Før kalibrering kan det være ganske store avvik fra 0mV når man trykker kalibreringsknappen.

Vi er nå klare til å måle ORP-verdien i sjøvann og f.eks. destillert vann:

ORP-verdien for sjøvann er **+340mV ved 24°C** etter flere minutters måling (verdien driver fra ca. 400mV og ned til 340mV)

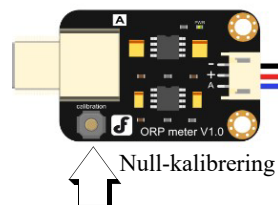
ORP-verdien for destillert vann varierer ganske tilfeldig fra ca. 200 – 700mV ved 23,5°C

### Eksempelprogram

Eksempelkode finnes G.6 side 282

### Behandling av proben

- Vask proben i ionebyttet vann (destillert vann) flere ganger etter bruk. Om nødvendig bruk lunkent vann.
- Etter lengre tids bruk kan proben bli passivisert. Dette merkes ved at følsomheten går ned, responsen blir langsom og sensoren mister nøyaktighet. Sett proben i en 0,5 mol hydroclorid oppløsning i 24 timer.
- Passivering kan også skje dersom sensoren blir sterkt forurenset. I så fall må forurensningen fjernes med dertil egnet renevæske.
- Etter et år med bruk kan det være på tide å skifte probe.





## 6.9 Sensor for måling av vanntrykk (SEN-0257)<sup>120</sup>

Vanntrykksmåleren SEN-0257 måler vanntrykk fra 0 – 1,6 Mpa. Vanntrykket øker med ca. 100kPa pr. 10 meter og kan da i prinsippet måle dybder fra 0 – 150 meter og noe mindre i saltvann siden egenvekten for saltvann er noe høyere (1,020 to 1,029 kg/L).

Som figuren under viser er det en lineær sammenheng mellom vanntrykket og spenningen ut. Kretsen skal ha 5V inn og vil da levere en spenning på fra 0,5V ved 0 Pa til 4,5V ved 1,6MPa (150m).

Normalt vil det derfor ikke være behov for en nivåjustering da spenningen ut av sensoren ikke vil overskride grensen på ca. 3,0V før ved en dybde på nærmere 90 meter.

Fra figuren til høyre ser vi at en spenning på 4000mV gir en variasjon fra 0 – 1,6MPa. Dette gir en omregningsformel fra spenning, U, i Volt til trykk, p, i kPa lik:

$$p = 1600/4 U \text{ [kPa]} = 400 \cdot U \text{ [kPa]} \quad (6.19)$$

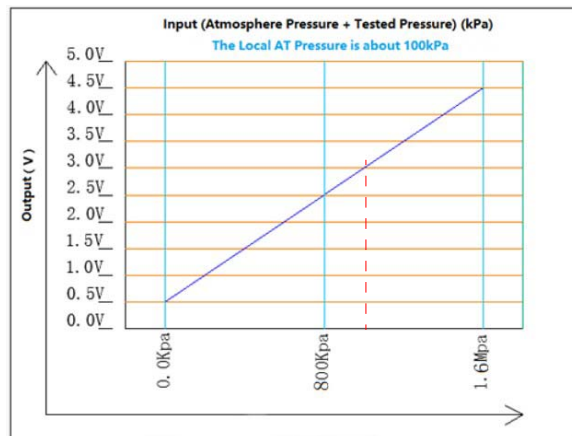
### Virkemåte

Sensoren består av en tynn skive av et pezoresistivt silisiummateriale som, når det utsettes for en kraft vil endre resistivitet. Ved hjelp av en Wheatstones målebro vil en liten endring i resistivitet omdannes til en endring i spenning. Sensoren er dessuten styrt med elektronikk som forsterker og lineariserer spenningen slik at resultatet blir som vist på figuren over.

### Spesifikasjoner

Under er listet opp de viktigste spesifikasjonene:

- Medium: Ikke korroderende væsker eller gasser
- Oppkobling: 3 pinner Gravity-plugg (Signal-VCC-GND)
- Måleområde trykk: 0~1.6 Mpa
- Supplyspenning: +5 VDC



<sup>120</sup><https://www.dfrobot.com/product-1675.html>

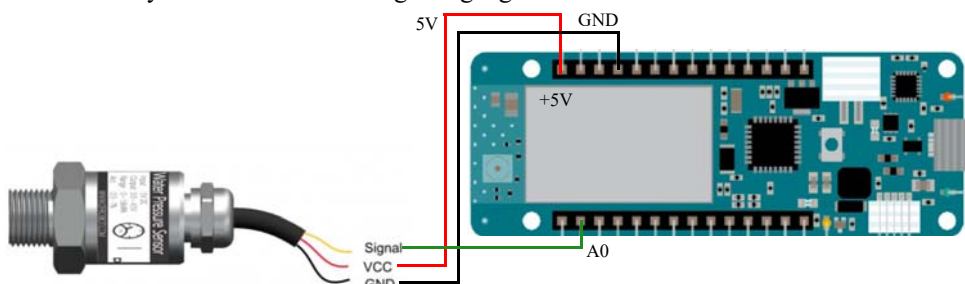


- Utgangsspenning: 0.5~4.5 V
- Målenøyaktighet: 0.5%~1% av fullskala (FS) (0.5%, 0~55°C)
- Gjenger: G1/4
- Mekanisk adapter: G1/2 to G1/4
- Nivå for vanntetthet: IP68
- Temperaturområde: -20~85°C
- Responstid: <2.0 ms
- Midlere strømtrekk: 2.8 mA
- Mulig operativt område for trykk:  $\leq 2.0$  Mpa
- Maksimalt trykk før ødeleggelse:  $\geq 3.0$  Mpa
- Levetid mht. antall målinger: 10'000'000 ganger

### Oppkobling

Sensoren er normalt utstyrt med hylsekontakter og med forskjellig fargete ledninger: Rød (+5V), Sort (GND) og Gul (Signal). Her har vi valgt å føre den gule videre som grønn ledning, for å gjøre den mer synlig i figuren. For å kunne koble oss til et koblingsbrett så bruker vi en liten stiftlist med tre pinner. Dermed kan vi koble hylsekontakten til koblingbrettet.

For enkelhetsskyld bruker vi den analoge inngangen A0 hos mikrokontrolleren.



Siden trykksensoren har et stort måleområde i forhold til hva vi har av mulighet til å teste, så dermed vil bare en liten del av AD-konverterens dynamiske område bli utnyttet.

La oss gjøre et enkelt overslag:

Vi velger å sette AD-konverteren til 12 bit som utgjør 4096 nivåer. Dersom vi antar at det dynamiske området er fra 0,5 – 4,5V, så vil hver bit utgjøre ca. 1 mV. Dersom vi antar at vår målesylinder er 1 meter og det dynamiske område er på 150 meter (0 – 1,6 MPa) så vil hvert bit utgjøre ca. 3,7 cm. En målesylinder på ca. 1 meter skulle derfor gi utslag med ca. 27 mV eller en tilsvarende endring i bit-nivå på 27.

For måling på grunt vann kan det være verdt å vurdere å sette en forsterker i signalgangen mellom sensoren og A0 for å øke utslaget.





## Omregning fra trykk til dybde

Vi vet at trykk i en væske måles med benevnningen  $\text{N/m}^2$ :

$$1\text{N/m}^2 = 1\text{Pa}$$

$$100000\text{Pa} = 100\text{kPa} = 1\text{Bar} = 1000\text{mBar}$$

$$1\text{atm} = 101,325\text{kPa}$$

Vi vet også at trykket,  $p$  [ $\text{N/m}^2$ ], mot en flate,  $A$  [ $\text{m}^2$ ], i en dybde,  $d$  [ $\text{m}$ ] nede i vannet, er gitt av følgende sammenheng [11]:

$$p = \frac{Ad\rho g}{A} = d\rho g \quad (6.20)$$

Der

$$\rho = \text{væskens tetthet} [\text{kg/m}^3]$$

$$g = \text{tyngdeakselerasjonen} [\text{m/s}^2]$$

Vi registrerer at benevnningen stemmer:

$$p = d \cdot \rho \cdot g = [\text{m}] \cdot [\text{kg/m}^3] \cdot [\text{m/s}^2] = [\text{kg m/s}^2]/[\text{m}^2] = [\text{N/m}^2] = [\text{Pa}] \quad (6.21)$$

Normalt vil trykket ved overflata av væsken,  $p_0$ , adderes til trykket som oppstår pga. tyngden av væskesøylen, slik at vi kan skrive:

$$p = p_0 + d \cdot \rho \cdot g \quad (6.22)$$

Siden vi kun er interessert i trykkforskjellen mellom overflata og ved den dybden vi måler, så trekker vi fra lufttrykket ved overflata. Siden tettheten,  $\rho$ , av ferskvann ved denne temperaturen er  $1\text{kg/L}$  eller  $1000\text{kg/m}^3$ , og tyngdeakselerasjonen er tilnærmet  $10\text{ m/s}^2$  så ser vi at:

$$p = d \cdot \rho \cdot g = 10\text{ m} \cdot 1000\text{ kg/m}^3 \cdot 10\text{ m/s}^2 = 100\,000\text{ Pa} = 100\text{ kPa} = 1\text{ Bar} \quad (6.23)$$

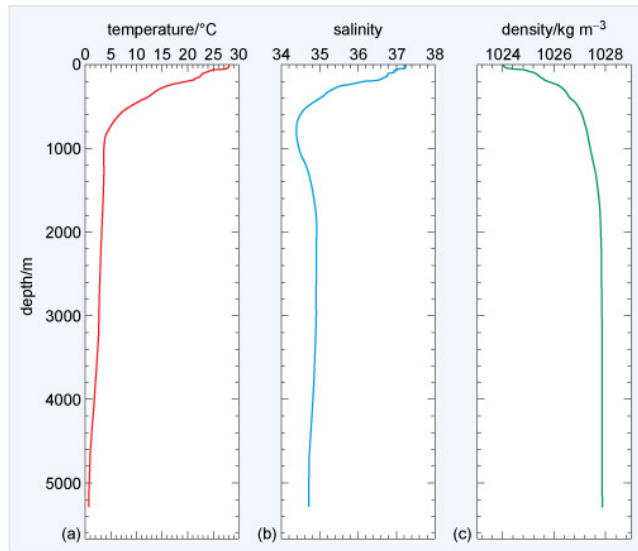
Grovt sett kan vi si at trykket i vann øker med 1 Bar for hver 10. dybdemeter.

Eller mer nøyaktig:

$$p = d \cdot \rho \cdot g = 10\text{ m} \cdot 1000\text{ kg/m}^3 \cdot 9,81\text{m/s}^2 = 98\,100\text{ Pa} = 98,1\text{kPa} = 0,981\text{Bar} \quad (6.24)$$



Tettheten i sjøvann kan imidlertid variere både med dybden som vist til høyre på figuren under <sup>121</sup>. Den vil også variere med saltholdighet og temperatur som vi også ser variere med dybden (til høyre på figuren under). Typisk er tettheten hos sjøvann i området fra 1,020 – 1,029 kg/L:



Vi ønsker å finne dybde fra overflata,  $d$ , som funksjon av vannets tetthet,  $\rho$ , og forskjell i vanntrykk,  $p$ , mellom overflata og den aktuelle dybden:

Vi kan nå beregne dybden på grunnlag av trykkmålinger med følgende uttrykk:

$$d = p / (\rho_s \cdot g) = p / (1025 \text{ kg/m}^3 \cdot 9,81 \text{ m/s}^2) = p / 10055,25 \text{ [m]} \quad (6.25)$$

Der:

$\rho_s = 1,025 \text{ kg}$  (Midlere tetthet for sjøvann)

$g = 9,81 \text{ m/s}^2$  (Tyngdeakselerasjonen)

Vi ser da bort fra endringer i tetthet som funksjon av temperaturen samtidig som vi velger en midlere verdi for tettheten til vannet.

### Kalibrering

Normalt skal utgangsspenningen fra sensoren være 0,5V når den befinner seg rett over vann flata. På grunn av drift i elektronikken og høytrykk/lavtrykk i atmosfæren, vil dette kunne avvike noe fra 0,5V. Det er derfor nødvendig å gjøre en innledende måling over vann før vi senker den ned i vann (f.eks. en målesylinder).

---

<sup>121</sup><https://www.open.edu/openlearn/science-maths-technology/the-oceans/content-section-3.2> (Målinger gjort i Atlanterhavet)



Vi kobler opp kretsen, installerer et program for å avlese spenningen fra sensoren og legger verdien tilbake i programmet der det står: `const float OffSet = 0.767;` I dette eksempelet var spenningen på utgangen av sensoren 0,767V ved overflata.

### Program for kalibrering og måling

```
// Program for kalibrering og måling av trykk og vanndybde
// Programmet tar utgangspunkt i et program skrevet for DFRobot og modifisert
// av:
// Nils Kr. Rossing 05.10.23

const float OffSet = 0.767;          // See Calibration Note!
const float waterDensity = 1.000; // kg/L Ferskvann 1,000 kg/dm3 Saltvann 1,020
- 1,029 kg/dm3
const float g = 9.81;                // Gravitation constant 9,81 m/s2
const int averageNo = 500;          // Antall målinger for midling

float V, P, D;                       // Voltage, Pressure and Depth
void setup() {
  Serial.begin(115200);
  Serial.println("/** Water Pressure Sensor Data **/");
  analogReadResolution(12);         // 12Bits
}
void loop() {
  //Connect sensor's output (SIGNAL) to Analog 0
  V = 0;
  for(int i=0; i<averageNo; i++ ) V = V + analogRead(0) * 5.00 / 4096;

  V = V/averageNo;
  P = (V - OffSet) * 400;
  D = P/(waterDensity*g); // P in kPa and waterDensity in kg/dm3 and g in m/s2

  Serial.print("Voltage:");
  Serial.print(V, 3);
  Serial.println("V");
  Serial.print("Pressure:");
  Serial.print(P, 1);
  Serial.println("kPa");
  Serial.print("Water depth:");
  Serial.print(D, 2);
  Serial.println("m");
  Serial.println();
}
```



---

```
    delay(1000);  
}
```



## 7 Referanser

- [1] Ervik, H. “Miljøovervåking i tareskogen”, SLserien nr. 19, Skolelaboratoriet NTNU 2019
- [2] Rossing, N.K. “MauSea – trykk- og temperaturmåling under vann”, Skolelaboratoriet, Blå hefteserie (<https://www.ntnu.no/skolelab/blå-hefteserie>). Under overskriften “ROV (Mau-Sea)” og fanen “ROV med trykk- og temperaturmåling, 2018”
- [3] Haugen, F.A. og Lysaker, M., “Python for REALFAG”, Utgave 1 Fagbokforlaget , Bergen 2020.
- [4] Sadar, M. “Turbidity standards - Technical information booklet No. 12”, <https://www.hach.com/asset-get.download.jsa?id=7639984474>
- [5] Rossing, N.K “Grunnopløring Arduino - Trafikklys”, Skolelaboratoriet 2021 <https://www.ntnu.no/documents/2004699/12108297/Grunnoppl%C3%A6ring+arduino+-+trafikklys.pdf/c4de7673-b114-d2e1-25e9-2490a656ee7e?t=1604657195148>
- [6] Rossing, N.K. “Måling av partikkeltetthet i væske med Arduino”, Skolelaboratoriet 2019 <https://www.ntnu.no/documents/2004699/12108297/M%C3%A5ling+av+partikkeltetthet+i+v%C3%A6ske+med+Arduino.pdf/2b60dea3-dde3-7ab0-d635-a81d97f93db8?t=1574244235236>
- [7] Rossing, N.K. Stausland, C “Lag en barometrisk høydemåler”, Skolelaboratoriet 2022 <https://www.ntnu.no/documents/2004699/12108297/Grunnkurs+programmering+Arduino+-+lag+en+h%C3%B8ydem%C3%A5ler.pdf/4ce3a7a7-d0f8-4832-8e57-954953c42331?t=1643110194763>
- [8] Rossing, N.K. “Videregående oppløring, Arduino: Værstasjon”, Skolelaboratoriet 2022 <https://www.ntnu.no/documents/2004699/12108297/V%C3%A6rstasjon+-+videreg%C3%A5ende+oppl%C3%A6ring+Arduino.pdf/80c10c12-2e19-d474-94d1-fbf137e73f60?t=1643110308768>
- [9] Rossing, N.K. Hansen, T.I “Miljøovervåking med havbøye, Arduino MKR NB 1500 – Data til server ved Inst. for marin teknikk”, Rev. 4.2, Skolelaboratoriet 2023
- [10] PICAXE, “Tutorials”, <https://picaxe.com/software/picaxe/blockly-for-picaxe/>
- [11] Bruun S. Devik Olaf, Råstad H., Viervoll H., “Fysikk fra gymnaset – Mekanikk”, Olaf Norlis Forlag, Oslo 1970



## Vedlegg A Komponentliste

Tabellen under inneholder noen av komponentene som er brukt og hvor de kan skaffes fra. Lista kan brukes som underlag for å sende ut komponenter til kursdeltagerne:

**Tabell A.1 Komponentliste**

Typebetegnelse	Type komponent	Leverandør	Pris	Nettadresse
<b>Komponenter i kurspakken</b>				
MKR NB 1500	Mikrokontroller	RS-online (inkl. MVA)	1095,38 (31.05.23)	<a href="https://no.rs-online.com/web/p/arduino/1763642">https://no.rs-online.com/web/p/arduino/1763642</a>
MKR GPS	GPS kort	RS-online (inkl. MVA)	471,06 (31.05.23)	<a href="https://no.rs-online.com/web/p/shields-for-arduino/1894522/">https://no.rs-online.com/web/p/shields-for-arduino/1894522/</a>
MKR SD Proto Shield	Kort for lagring av data på SD-kort	RS-online	201,58 (21.06.23)	<a href="https://no.rs-online.com/web/p/shields-for-arduino/1697589">https://no.rs-online.com/web/p/shields-for-arduino/1697589</a>
GSM antenna	m/ UFL connector (868 Mhz)	RS-online (inkl. MVA)	65,62 (31.05.23)	<a href="https://no.rs-online.com/web/p/shields-for-arduino/1697591">https://no.rs-online.com/web/p/shields-for-arduino/1697591</a>
USB kabel	USB A til micro USB B kabel	RS-online (inkl. MVA)	87,85 (31.05.23)	<a href="https://no.rs-online.com/web/p/usb-cables/8762403">https://no.rs-online.com/web/p/usb-cables/8762403</a>
DS18B20	Temperatursensor	Mouser Electronics (inkl. mva)	94,59 (31.05.23)	<a href="https://no.mouser.com/ProductDetail/DFRobot/DFR0198?qs=Zcin8yvlnh-POt9ZkO3roFQ%3D%3D">https://no.mouser.com/ProductDetail/DFRobot/DFR0198?qs=Zcin8yvlnh-POt9ZkO3roFQ%3D%3D</a>
CR1216	Knappebatteri	ELFA Distrelec	26,50 (02.06.23)	<a href="https://www.elfadistrelec.no/no/knappecellebatteri-litium-cr1216-3v-25mah-varta-cr1216/p/16924681">https://www.elfadistrelec.no/no/knappecellebatteri-litium-cr1216-3v-25mah-varta-cr1216/p/16924681</a>
Minnekort SD	32 Gbyte minnekort Kingston m/adapter	Komplett	72,00 (21.06.23)	<a href="https://www.komplett.no/product/1146043/hjem-fritid/sport-fritid/kameratilbehoer/minnekort-til-foto-video/kingston-canvas-select-plus-microsd-32gb">https://www.komplett.no/product/1146043/hjem-fritid/sport-fritid/kameratilbehoer/minnekort-til-foto-video/kingston-canvas-select-plus-microsd-32gb</a>
Totalt for pakke inkl. MVA			2114,58 (21.06.23)	Frakt kommer i tillegg
<b>Komponenter til tilleggspakke for oppbygging av kretskort</b>				
Mini Micro JST 2.0 PH	Batteriplugg (100 stk)	Banggood	kr. 27,05 (31.05.23)	<a href="https://www.banggood.com/Excellway-100Pcs-Mini-Micro-JST-2_0-PH-2Pin-Connector-Plug-With-120mm-Wires-Cables-p-1147298.html">https://www.banggood.com/Excellway-100Pcs-Mini-Micro-JST-2_0-PH-2Pin-Connector-Plug-With-120mm-Wires-Cables-p-1147298.html</a>
Li-Po batteri 2950 mAh	Batteri	Biltema	kr. 100,00 (31.05.23)	<a href="https://www.biltema.no/kontor-teknikk/batterier/oppladbare-batterier/oppladbart-icr18650-batteri-2950-mah-2000037909">https://www.biltema.no/kontor-teknikk/batterier/oppladbare-batterier/oppladbart-icr18650-batteri-2950-mah-2000037909</a>
Li-Po batteri lader	Batterilader	Biltema	kr. 70,00 (31.05.23)	<a href="https://www.biltema.no/kontor-teknikk/batterier/batteriladere/batterilader-til-batteri-18650-2000046111">https://www.biltema.no/kontor-teknikk/batterier/batteriladere/batterilader-til-batteri-18650-2000046111</a>
Skrukontakt til DS18B20	3 pin skrukontakt (10 stk)	Aliexpress	kr. 18,95 (31.05.23)	<a href="https://www.aliexpress.com/item/1005001677869988.html">https://www.aliexpress.com/item/1005001677869988.html</a>
DS18B20	1 stk. Temperatursensor, vannrett	Aliexpress	kr. 23,30 (31.05.23)	<a href="https://www.aliexpress.com/item/1005002719614657.html">https://www.aliexpress.com/item/1005002719614657.html</a>



Typebetegnelse	Type komponent	Leverandør	Pris	Nettadresse
Kontakter	10 stk hylsekontakt 10 stk stifflist	Aliexpress	kr. 19,50 (31.05.23)	<a href="https://www.aliexpress.com/item/1005002577212594.html">https://www.aliexpress.com/item/1005002577212594.html</a>
Picaxe 08M2	1 stk mikrokontroller	PICAXE	£ 2,0 (31.05.23)	<a href="https://picaxe.com/hardware/picaxe-chips/picaxe-08m2-microcontroller/">https://picaxe.com/hardware/picaxe-chips/picaxe-08m2-microcontroller/</a>
	1 stk kabel for programmering		£ 12,50 (31.05.23)	<a href="https://picaxe.com/hardware/cables/picaxe-usb-download-cable/">https://picaxe.com/hardware/cables/picaxe-usb-download-cable/</a>
	1 stk Proto board for programmering		£ 2,50 (31.05.23)	<a href="https://picaxe.com/hardware/project-boards/picaxe-08-proto-board/">https://picaxe.com/hardware/project-boards/picaxe-08-proto-board/</a>
	10 stk 8 pin sokler		kr. 7,02 (31.05.23)	<a href="https://www.aliexpress.com/item/1005003256812123.html">https://www.aliexpress.com/item/1005003256812123.html</a>
Booster	Spennings-booster 5V	Aliexpress	kr. 22,07 (31.05.23)	<a href="https://www.aliexpress.com/item/32899720887.html">https://www.aliexpress.com/item/32899720887.html</a>
TP 4056	5 stk. USB C Lader	Aliexpress	kr. 17,61 (31.05.23)	<a href="https://www.aliexpress.com/item/1005004332269949.html">https://www.aliexpress.com/item/1005004332269949.html</a>
Holder 18650	1 stk Batteriholder (for 2 batterier)	Aliexpress	kr. 8,58 (31.05.23)	<a href="https://www.aliexpress.com/item/1005001861065786.html">https://www.aliexpress.com/item/1005001861065786.html</a>
USB A (Hunn)	10 stk strøm-forsyningsutgang	Aliexpress	kr. 4,46 (31.05.23)	<a href="https://www.aliexpress.com/item/1005004556799784.html">https://www.aliexpress.com/item/1005004556799784.html</a>
Micro slide switch	25 stk brytere	Aliexpress	kr. 47,80 (31.05.23)	<a href="https://www.aliexpress.com/item/1005003938407295.html">https://www.aliexpress.com/item/1005003938407295.html</a>
2N7000	100 stk MOSFET	Aliexpress	kr. 27,08 (31.05.23)	<a href="https://www.aliexpress.com/item/1005002977107756.html">https://www.aliexpress.com/item/1005002977107756.html</a>
Kretskort	1 stk Kretskort	Vers. 1.4		Thor Inge Hansen Skien vgs
<b>Aktuelle sensorer for innkjøp til kurset</b>				
SEN0161	pH-sensor	DFRobot	\$ 29,5	<a href="https://www.dfrobot.com/product-1025.html">https://www.dfrobot.com/product-1025.html</a>
		Mouser	Kr.432,0	<a href="https://no.mouser.com/">https://no.mouser.com/</a>
SEN0237-A	Okygeninnhold	DFRobot	\$ 169,0	<a href="https://www.dfrobot.com/product-1628.html">https://www.dfrobot.com/product-1628.html</a>
		Mouser	Kr. 1859,0	<a href="https://no.mouser.com/">https://no.mouser.com/</a>
SEN0189	Turbiditet	DFRobot	\$ 9,9	<a href="https://www.dfrobot.com/product-1394.html">https://www.dfrobot.com/product-1394.html</a>
		Mouser	Kr. 108,55	<a href="https://no.mouser.com/">https://no.mouser.com/</a>
DFR0300-H	Salinitet	DFRobot	\$ 79,9	<a href="https://www.dfrobot.com/product-1797.html">https://www.dfrobot.com/product-1797.html</a>
		Mouser	Kr. 875,78	<a href="https://no.mouser.com/">https://no.mouser.com/</a>
SEN0244	Ferskvannskvalitet (TDS)	DFRobot	\$ 11,8	<a href="https://www.dfrobot.com/product-1662.html">https://www.dfrobot.com/product-1662.html</a>
		Mouser	Kr. 141,44	<a href="https://no.mouser.com/">https://no.mouser.com/</a>
SEN0257	Vanndrykk	DFRobot	\$ 15,9	<a href="https://www.dfrobot.com/product-1675.html">https://www.dfrobot.com/product-1675.html</a>
		Mouser	Kr. 199,50	<a href="https://no.mouser.com/">https://no.mouser.com/</a>
SEN0554	Non contact Turbidisensor	DFRobot	\$ 9,9	<a href="https://www.dfrobot.com/product-2652.html">https://www.dfrobot.com/product-2652.html</a>
<b>Andre komponenter</b>				



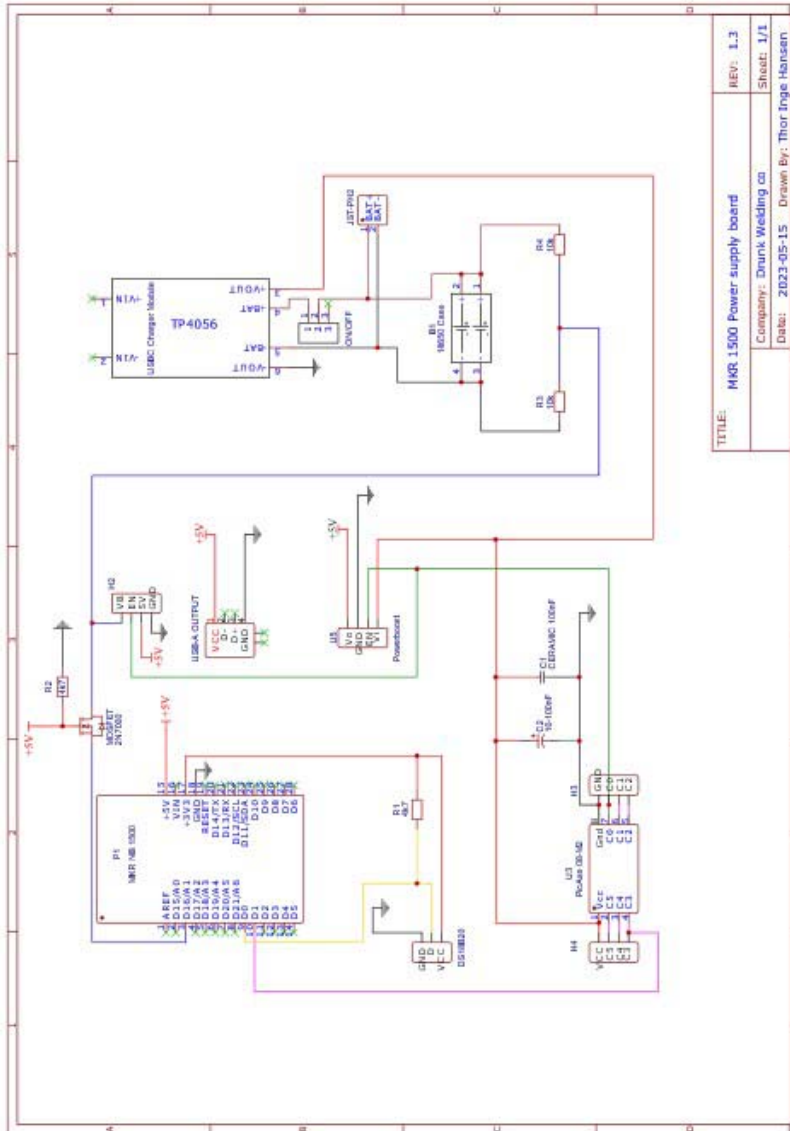


Typebetegnelse	Type komponent	Leverandør	Pris	Nettadresse
MKR Mem Shield	Kort for lagring av data på SD-kort	RS-online	286,21 (19.06.23)	<a href="https://no.rs-online.com/web/p/shields-for-arduino/1763644">https://no.rs-online.com/web/p/shields-for-arduino/1763644</a>
MKR ENV	Miljø sensorkort	RS-online (inkl. MVA)	487,14 (31.05.23)	<a href="https://no.rs-online.com/web/p/shields-for-arduino/2198445/">https://no.rs-online.com/web/p/shields-for-arduino/2198445/</a>
MKR Proto Shield	Protypkort for monterings sensorer	RS-online	115,96 (19.06.23)	<a href="https://no.rs-online.com/web/p/shields-for-arduino/1697586">https://no.rs-online.com/web/p/shields-for-arduino/1697586</a>
MKR Proto Shield, large	Prototype kort, stort	RS-online	132,31 (19.06.23)	<a href="https://no.rs-online.com/web/p/shields-for-arduino/1697587">https://no.rs-online.com/web/p/shields-for-arduino/1697587</a>



## Vedlegg B Kretsskjema og PCB-layout av kretskort<sup>122</sup>

Denne kretsen inkluderer MKR NB 1500 og forsynes av to Li-Po-batterier som kan plasseres på undersiden. I tillegg benyttes en liten mikrokontroller som en eksternt "vakthund" og en spenningsbooster for å heve spenningen til ca. 5 V. Kortet inkluderer også koblingsplint for tilkobling av stemperensoren DS18B20. En ladekrets er inkludert for lading av batteriene.



122.Krettskjemaet og layout er gjort av Thor Inge Hansen ved Skien videregående skole





## Vedlegg C Feilfinning og problemer

I dette vedlegget skal vi samle lure triks for å komme rundt feil som kan oppstå.

### C.1 Problemer med opplasting av programmet<sup>123</sup>

Mens jeg arbeidet med både MKR NB 1500, påmontert MKR ENV og MKR GPS, ble det etter hvert umulig å laste opp programmet. Dvs. at den forsøkte, men oppførte seg annerledes enn tidligere og avsluttet opplastingen etter lang tid med en feilmelding.

Mange mikrokontrollerkort har egne driverkretser for å kommunisere med USB-bussen, slik er det ikke med alle. For MKR NB 1500 ligger USB programvaren sammen med skisseprogrammene og kjøres dermed av den samme prosessoren. Det kan derfor skje at skisseprogramvaren kan ødelegge og blokkere kommunikasjonen over USB-bussen slik at f.eks. opplasting blir umulig.

Imidlertid er ikke situasjonen umulig. Det finnes nemlig en egen bootloader som er skjernet fra skisseprogramvaren, og som har sin egen USB-kode. Ved å aktivere denne kan man komme rundt problemet og normalisere opplastingen. Denne koden kan aktiveres på følgende måte:

1. Trykk resett-knappen (RST) på kortet to ganger i rask rekkefølge. Du vil da se at den grønne lysdioden på kanten av kortet begynner å pulsere langsomt. Denne prosedyren gjør at bootloaderen kjører kontinuerlig slik at timingen ved opplastingen ikke er kritisk.
2. Velg Tools > Port, og velg riktig port som kan være en annen enn den som tidligere er brukt av kortet.
3. Så lastes skissen opp på vanlig måte. Neste gang du laster opp programmet, kan dette gjøres på vanlig måte.

### C.2 Problemer med opplasting av programmet pga dvale<sup>124</sup>

Dersom kretsen legges i dvale i lengre perioder, med relativt korte perioder hvor den er våken, kan man lett komme i en situasjon der man ikke får kontakt med mikrokontrolleren. I dvale (deep sleep) vil også kommunikasjonen med USB-porten legges i dvale slik at man ikke får kontakt med kortet og ikke får lastet inn et nytt program.

For å komme ut av situasjonen gjør man som under C.1:

1. Trykk resett-knappen på kortet to ganger i rask rekkefølge. Du vil da se at den grønne lysdioden på kanten av kortet begynner å pulsere langsomt. Denne prosedyren gjør at kortet våkner opp og bootloaderen begynner å kjøre.
2. Velg Tools > Port, og velg riktig port som kan være en annen enn den som tidligere er brukt av kortet.
3. Så lastes skissen opp på vanlig måte.

---

123.En takk til *Pert* <https://forum.arduino.cc/t/unbricking-an-audrino-mkr1500/878247>

124.En takk til *Pert* <https://forum.arduino.cc/t/mrk-1500-stuck-in-deep-sleep/690513>



### C.3 Problemer med å lese verdier fra MKR GPS sammen med MKR ENV<sup>125</sup>

Når både shield-kortene MKR GPS og MKR ENV er koblet til samtidig, fikk jeg problemer med å lese av GPS-kortet. Jeg kombinerte eksempelfilen fra de to kortene rått og fikk problemer når void loop()-funksjonen ble fylt opp med flere funksjoner, da uteble leste data fra GPS-kortet.

Under er vist den opprinnelige void loop()-funksjonen fra GPS-eksemplet:

```
void loop() {
  if (GPS.available()) {
    // les GPS verdier
    float latitude   = GPS.latitude();
    float longitude  = GPS.longitude();
    float altitude   = GPS.altitude();
    float speed      = GPS.speed();
    int  satellites  = GPS.satellites();

    // skriv utGPS verdier
    Serial.print("Location: ");
    Serial.print(latitude, 7);
    Serial.print(", ");
    Serial.println(longitude, 7);

    Serial.print("Altitude: ");
    Serial.print(altitude);
    Serial.println("m");

    Serial.print("Ground speed: ");
    Serial.print(speed);
    Serial.println(" km/h");

    Serial.print("Number of satellites: ");
    Serial.println(satellites);

    Serial.println();
  }
  //delay(1000);
}
```

---

125.En takk til *mstepanek1976* <https://forum.arduino.cc/t/env-shield-gps-shield-not-working/657837>



```
}
```

Dette var hele void loop()-funksjonen i eksempelkoden. Legg spesielt merke til //delay(1000); i nest siste linje. Denne er her kommentert ut. Så snart denne ble lagt inn i programmet, så sluttet det å skrive ut GPS-måleverdier. Det samme skjedde dersom andre kommandoer ble lagt inn på slutten av funksjonen.

Innhenting og utskrift av måleverdiene er lagt inn i en if-setning som har som betingelse at nye data skal være tilgjengelige:

```
if (GPS.available()) { ... }  
// les GPS verdier  
...
```

Dvs. at returnen fra GPS.available()==1 for at dataene kan hentes og skrives ut. Dersom de ikke er det, hopper programmet over innhenting og utskrift.

Etter hvert oppdaget jeg at for å hente ut måledata måtte programmet gå mange runder ja flere hundre eller tusen før dataene var klare. Så snart en delay() ble lagt inn så tok det jo mye lengre tid før data kunne hentes ut. Så datene kom sikkert med det tok så lang tid at det opplevdes som de uteble.

Problemet ble løst ved å kjøre mange raske runder med følgende kommando:

```
while (!GPS.available()) {}
```

Her vil programmet gå en svært raskt while-loop helt til dataene er klare. Dermed oppleves det som om de kommer umiddelbart og programmet kan gå videre.

Problemet løst.

## C.4 Problemer med resetting av programmet

Mikrokontrolleren er tilkoblet PC'en via USB-kabelen og datainnsamlingen startes ved å laste programmet over til mikrokontrolleren. Samtidig åpnes monitoren slik at vi kan se meldinger som returneres fra programmet, som er praktisk under utvikling og testing.

Vi har erfart følgende:

- Når vi trykker på resett-knappen på kortet slutter programmet å samle inn og overføre data
- Nærmere undersøkelser har vist at programmet henger fordi det ikke får opprettet kontakt med monitoren via USB-kabelen. Det vi ser er at den blir stående å "spinne" i while(!Serial)-loopen og vente på DTR (Data Terminal Ready) fra PC'en som ikke kommer:

```
while (!Serial)  
{  
    ; // wait for serial port to connect. Needed for native USB port only  
}
```

Dersom vi fjerner while-loopen beskrevet foran, så vil det samme skje neste gang vi møter en Serial.print(...) kommando. Den vil stoppe programmet fordi seriekommunikasjonen med monitoren ikke er opprettet.



Når man trykker resett-knappen så brytes serie-kommunikasjonen med monitoren via USB-kabelen. Denne kan først opprettes ved at PC'en laster opp programmet eller når monitoren åpnes.

- **Løsning:** Løsningen blir derfor å fjerne all serie-kommunikasjon i det endelige programmet før det lastes over til bøya. En serie-kommunikasjon vi likevel ikke trenger.
- For å få en indikasjon på hvor i programmet mikrokontrolleren befinner seg så kan man, istedet for tekst, legge inn blinksekvenser med den interne lysdioden. Disse bør fjernes i den endelige versjonen av programmet da LED'en trekker strøm, som vist under:

```
for(int i=0; i<8; i++)
{
  delay(50);
  digitalWrite(LED_BUILTIN,HIGH);
  delay(50);
  digitalWrite(LED_BUILTIN,LOW);
}
```

## C.5 Programmet stopper å samle inn data

Vi har kjørt programmet over lengre perioder og har opplevd at det har stoppet, i en serie av innsamlede data. Dersom en eller flere sensorer ikke gir data, vil programmet gi en feilmelding og gå videre. Slik programmet nå er skrevet så vil det i enkelte tilfeller kunne ende opp i en uendelig sløyfe. Dette gjelder blant annet oppkobling til 4G-nettet:

```
void connectToGPRS()
{
  // Koble opp til GPRS nettverket med APN, login og passord
  Serial.println("Starter Arduino web-client.");

  while (!connected)
  {
    if ((nbAccess.begin("", true, true) == NB_READY) &&
        (gprs.attachGPRS() == GPRS_READY))
    {
      connected = true;
      Serial.println("Tilkoblet GPRS");
    }
    else
    {
      Serial.println("Ikke tilkoblet GPRS");
      delay(1000);
    }
  }
}
```





Det er naturlig at programmet vil befinne seg i while()-loopen en stund. En kan f.eks. begrense antall runder i loopen og på den måten unngå at programmet låser seg. Det samme gjelder dersom den ikke klarer å låse til tilstrekkelig antall GPS-satellitter:

```
while (!GPS.available()) {}
```

## C.6 Problemer med å lese riktig lufttrykk fra ENV-kort når GPS er tilkoblet

Når både GPS-kortet og ENV-kortet er tilkoblet, ser det ut til at lufttrykksverdien blir alt for høy. Vi leser typisk 926mBar uten GPS-kortet tilkoblet, og 2591mBar når GPS-kortet er tilkoblet. Vi har likevel observert at dette ikke er helt konsekvent. Det er en sjelden gang registrert riktige verdier selv med GPS-kortet tilkoblet, men dette er unntaket. Vi har også observert det samme problemet med ulike GPS-kort.

Ser vi næyere på oppkoblingen av MKR ENV og MKR GPS, så legger vi merke til at EXTINT er lagt til samme pinne på mikrokontrolleren som DRDY Pressure fra MKR ENV-kortet, dvs. det ligger an til en konflikt. Den samme konflikten vil inntreffe enten vi bruker I<sup>2</sup>C eller UART-tilkoblingen mellom MKR GPS og mikrokontrollerkortet.

Forslag til løsning:

*Endring av oppkobling:*

1. Ta ut I<sup>2</sup>C ledningen som forbinder MKR GPS med MKR NB 1500
2. Koble opp MKR GPS til MKR NB 1500 via UART'en ved å forbinde:

D14 (Tx), MKR NB 1500 → D14 (Rx), MKR GPS  
D13 (Rx), MKR NB 1500 ← D13 (Tx), MKR GPS  
**D7, MKR GPS → D6, MKR NB 1500**  
med jumpere.

Siden vi tidligere har fått konflikt ved at både MKR GPS og MKR ENV bruker D7 i forbindelse med kommunikasjonen, så velger vi istedet å bruke D6 til å motta interrupt fra MKR GPS (EXTINT).

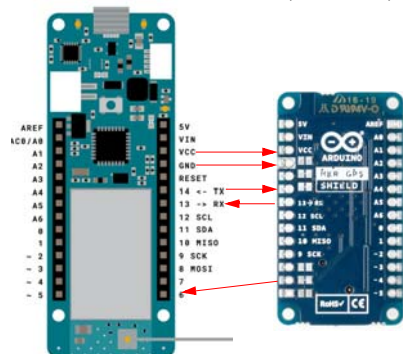
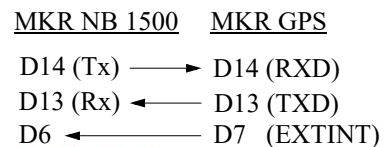
I tillegg må vi forbinde GND og VCC (3,3V) på begge kretsene.

*Endring i programmet:*

3. Man skifter kommunikasjonsport fra I<sup>2</sup>C til UART Rx/Tx (D13/D14) ved å initialisere GPS-kretsen med:

```
GPS.begin(GPS_MODE_SHIELD);
```

i setup()-funksjonen i programmet.





### Endring i biblioteket:

4. For at programmet i MKR NB 1500 skal kommunisere via port D6 istedet for port D7, må vi endre siste linje i biblioteket fra

```
GPSClass GPS(Serial1, 9600, serialDDC, 400000, 7);
```

til:

```
GPSClass GPS(Serial1, 9600, serialDDC, 400000, 6);
```

Biblioteket finner man her ved bruk av PC:

 > Denne PCen > Dokumenter > Arduino > libraries > Arduino\_MKRGPS > src

under navnet: GPS.cpp

### Tester synes å vise:

5. Ved å bruke UART (Rx/Tx) istedet for I<sup>2</sup>C bussen for å kommunisere GPS data til mikrokontrolleren, ser det ut til at vi unngår konflikten og både GPS-data og lufttrykk fra ENV-kortet synes å gi riktige verdier.
6. Man trenger ikke å krysse Tx/Rx mellom MKR NB 1500 og MKR GPS. Dvs. Tx MKR NB 1500 går til Tx MKR GPS og Rx MKR NB 1500 går til Rx MKR GPS. Dette må bety at merkingen av MKR GPS er feilmerket på hylsekontakten. Vi antar da at hylsekontakten er riktig merket på mikrokontrolleren.
7. Bruk av D6 istedet for D7 for å overføre interruptsignal, synes ikke å spille noen rolle. Det er derfor unødvendig å gjøre endringer i biblioteket.
8. *Konklusjonen er at GPS-kortet kan plugges rett ned i mikrokontrolleren, ev. at MKR ENV og MKR GPS-kortene kan stables direkte opp på hverandre.*

## C.7 Programmet klarer ikke å skrive til monitoren etter reset

Når programmet resettes enten manuelt eller ved hjelp av en reset-puls på RESET-inngangen, så skrives det ikke lengre til monitoren.

Dette er et kjent problem som lar seg løse ved at man lukker og åpner monitorvinduet på nytt. Det samme skjer ikke dersom programmet lastets opp på nytt.

Vi har også erfart at programmet stopper opp dersom det inneholder Serial.print-kommandoer, men har ingen serie linje å levere data til. Dette vil være tilfelle dersom mikrokontroller-kortet kjører på batteri og er uten USB-forbindelse til PC'en. I så fall vil man i prinsippet heller ikke ha noe behov for seriekommunikasjon. Det vil derfor være aktuelt å lage en egen versjon av programmet som er ment å brukes når mikrokontrolleren skal operere på egen hånd, og som er uten Serial.print-kommandoer.

I et slikt tilfelle kan man legge inn kommandoer som styrer kompilatoren, kommandoer av typen:

```
#define DEBUG // Kommenter bort denne dersom Serial.print ikke skal inkluderes  
//#define LED // Kommenter bort denne dersom LED_BUILTIN
```



```
.
.
.
#ifdef LED
  for(int i=0; i < 5; i++)
  {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(100);
    digitalWrite(LED_BUILTIN, LOW);
    delay(100);
  }
#endif

.
.
.
#ifdef DEBUG
  // Koble opp til GPRS nettverket med APN, login og passord
  Serial.println("Starter Arduino web-client.");
#endif
```

I dette eksempelet kan vi velge å skrive tekst til monitoren ved å definere en konstant `DEBUG`, dersom vi istedet for skriftlige beskjeder til monitoren vil vise hvor programmet befinner seg ved hjelp av blink i den innebygde lysdioden `LED_BUILTIN`, så kan vi definere konstanten `LED`.

Dersom konstanten benyttet i en `#ifdef` setning *ikke* er definert, men er kommentert bort, så vil ingen av kommandoene mellom `#ifdef` .... og `#endif` bli kompilert og bli med i programmet. Dermed er det lett å svitsje mellom ulike hjelpemidler for debugging.

## C.8 Programmet henger seg opp og kommer ikke videre

Vi har opplevd at programmet stopper opp dersom det skal koble seg til 4G (GPRS) eller serveren. I dette tilfellet hjelper det å gi programmet en hard reset. Dvs. å legge `RESET`-pinnen lavt i f.eks. 500 ms. Programmet vil i så fall starte på nytt.

For å være istand til dette kreves en ekstern “vakhund” som normalt blir “matet” regelmessig når programmet går normalt. Dersom “maten” ikke kommer i tide, sender den en reset-puls. En slik løsning synes å kreve at “vakhunden” får spenning også under dvale.

## C.9 Forsøk på oppkobling mot server synes å utebli

Vi har erfart at oppkoblingen mot serveren uteblir, dvs. programmet er innom, men synes ikke å gjøre noe alvorlig forsøk på oppkobling, men går raskt videre i programmet. I så måte synes det nødvendig å slå av og på mikrokontrolleren og starte helt på nytt. Dette krever at spenningen til kortet brytes og slås på, på nytt. Dette er en øvelse som kan tillegges “vakhunden”.

I et slikt tilfelle må “vakhunden” få beskjed om at den må respondere med en av-på-kommando. En slik kan legges inn i programmet når det erkjenner at oppkoblingen til serveren var mislykket. Et signal fra “vakhunden” må i så fall kunne styre en elektronisk av-på-knapp, f.eks. en MOSFET i power-ledningen.

Vi har prøvd å legge inn kommandoen som er en soft reset:



---

```
NVIC_SystemReset(); // Resett programmet
```

men det synes ikke å være tilstrekkelig.

### **C.10 Programmet klarer ikke å restarte etter dvale**

Vi har erfart at programmet ikke kommer tilbake etter dvale, selv om alt tyder på at det vekkes opp og synes å starte.

Vi har funnet ut at problemet synes å forsvinne dersom følgende kommando legges inn i oppstart-funksjonen (dummy):

```
NVIC_SystemReset(); // Resett programmet
```

som vist under:

```
void dummy()
{
  digitalWrite(PowerWDPin, LOW); // Vekk opp vakthunden
  GPS.wakeup();
  NVIC_SystemReset(); // Resett programmet
}
```



## Vedlegg D Sensor laboratoriejournal

Vedlegget beskriver prosedyren for karakterisering og kalibrering av noen aktuelle sensorer. Teorien er omtalt i kapittel 6 side 155.

- Måling av temperatur med NTC – vedlegg D.1 side 236
- Måling av temperatur med DS18B20 – vedlegg D.2 side 238
- Måling av oppløst oksygen med SEN-237A – vedlegg D.3 side 240
- Måling av vannets ledningsevne, saltholdighet med DFR0300-H – vedlegg D.4 side 245
- Måling av turbiditet, partikkeltetthet med SEN-0189 – vedlegg D.5 side 251
- Måling av pH med SEN-161 – vedlegg D.6 side 255
- Måling av oksidasjon og reduksjons potensial med SEN-0165 – vedlegg D.7 side 260
- Måling av trykk og beregning av dybde (SEN-0257) – vedlegg D.8 side 263



## D.1 Måling av temperatur i vann med NTC

Dette er en sensor som kan brukes for måling av temperatur i både luft og vann. Se detaljert beskrivelse i avsnitt 6.3.2 side 160.

### D.1.1 Utstyr

Følgende utstyr trengs:

- PC/MAC med programvare
- USB-kabel m/mikro USB
- MKR NB 1500 – mikrokontroller
- Koblingsbrett m/jumpere
- Seriemotstand ( $R_S = 10k\Omega$ )
- Sensor – NTC-motstand ( $10k\Omega$ )
- Kalibrert termometer
- Ett 100mL begerglass m/vann med ulik temperatur
- Sjøvann, varmt og kaldt

### Programvare

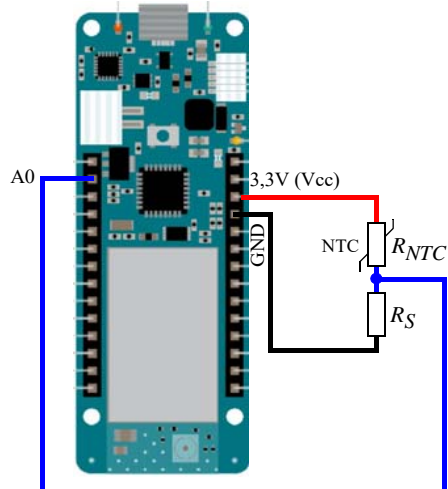
- Program for avlesning av temperatur Testprogram-MKR-Temp-Kalibrering-NTC-1.ino (se vedlegg D.1 side 236, eller last ned fra nettstedet: [www.ntnu.no/skolelab/bla-hefteserie](http://www.ntnu.no/skolelab/bla-hefteserie)).

### D.1.2 Oppkobling

Til dette formålet velger vi å koble opp kretsen på et lite koblingsbrett

### D.1.3 Kalibrering

1. Les teorien i avsnitt 6.3.2 side 160 før du gjennomfører kalibreringen
2. Koble opp kretsen i henhold til koblings skjemaet omtalt foran
3. Hent programmet for kalibrering av temperatur-sensoren: Testprogram-MKR-Temp-Kalibrering-NTC-1.ino.
4. Installer og kjør programmet.





5. Dypp sensoren i kaldt vann (f.eks. 6°C) sammen med et ordinært “kalibrert” termometer. Åpne monitoren og les av den digitale verdien og temperaturen på termometeret.

Kalibrert temperatur: \_\_\_\_\_ °C (T\_Lav)  
Digital verdi; \_\_\_\_\_ (D\_T\_Lav)

6. Dypp sensoren i lunkent vann (f.eks. 16°C) sammen med et ordinært termometer. Åpne monitoren og les av den digitale verdien og temperaturen på termometeret.

Kalibrert temperatur: \_\_\_\_\_ °C (T\_Hoy)  
Digital verdi; \_\_\_\_\_ (D\_T\_Hoy)

7. Sett inn de digitale verdiene og temperaturene i programmet:

```
// ----- Legg inn verdier for kalibrering av temperatur -----//  
  
int D_T_Lav = 339; // Digital verdi for spenningen på A0, registrert lav temperatur)  
int D_T_Hoy = 710; // Digital verdi for spenningen på A0, registrert ved høy temperatur)  
int D_T_Var = 0; // Digital variabel temperatur  
float T_Lav = 6; // Lav kalibreringstemperatur i grader C avlest på kalibrert insturment  
float T_Hoy = 16; // Høy kalibreringstemperatur i grader C avlest på kalibrert insturment
```

8. Last opp og kjør programmet på nytt med de kalibrerte verdiene. Dypp sensoren i vann med kjent temperatur. Åpne monitoren og sjekk om målt temperatur er i overensstemmelse med temperaturen i vannet.
9. Testprogrammet er nå klart til å bli inkludert i hovedprogrammet.





## D.2 Måling av temperatur i vann med DS18B20

Dette er en sensor som egner seg godt for måling av temperatur i vann.

### D.2.1 Utstyr

Følgende utstyr trengs:

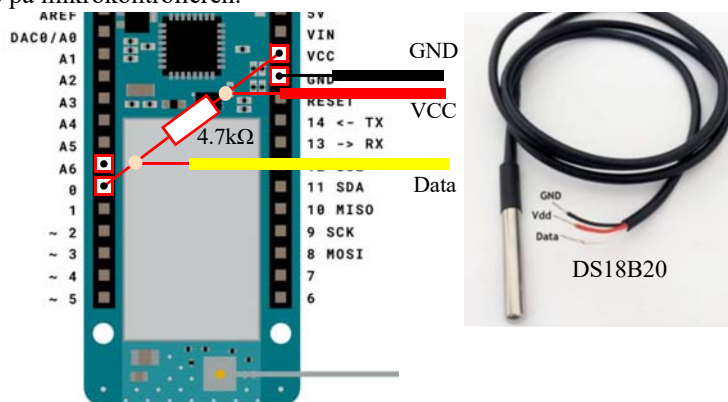
- PC/MAC med programvare
- USB-kabel m/mikro USB
- MKR NB 1500 – mikrokontroller
- Koblingsbrett m/jumper
- Kalibrert termometer
- Sensor DS18B20 m/4,7k $\Omega$  motstand
- Ett 100mL begerglass m/vann med ulik temperatur

### Programvare

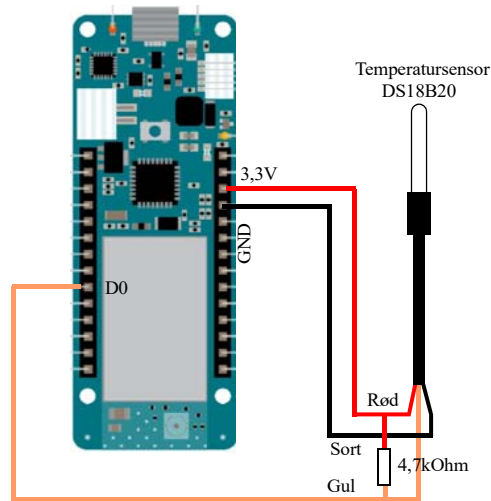
- Program for avlesning av temperatur: Temperatur-DS18B20.ino (se vedlegg D.2 side 238, eller last ned fra nettstedet: [www.ntnu.no/skollelab/bla-hefteserie](http://www.ntnu.no/skollelab/bla-hefteserie)).
- To biblioteker:  
#include <DS18B20.h> og  
#include <OneWire.h>  
Se avsnitt 2.3 side 39 for installasjon av disse bibliotekene om det ikke alt er gjort

### D.2.2 Oppkobling

Siden sensoren DS18B20 er inkludert i settet som er delt ut, velger vi å bruke denne enkle tilkoblingen direkte på mikrokontrolleren.



Alternativt kan man koble opp sensoren med koblingsbrett og jumper-ledninger. Koblingen kan da bli som vist under:



### D.2.3 Kalibrering

1. Varm opp vann eller bruk varmt vann fra varmtvannskranen på ca. 30°C
2. Fyll begerglasset med det varme vannet
3. Stikk temperatursensoren og et kalibrert termometer ned i begerglasset og rør rundt
4. Kjør programmet og les av temperaturen, les av når den er stabil og sammenlign med verdien hos det kalibrerte termometeret.
5. Sjekk temperaturen ved forskjellige temperaturer
6. Legg evt. inn en korleksjon i programmet om det er avvik mellom målt temperatur med sensor og temperaturen målt med det kalibrerte termometer et.



### D.3 Måling av oppløst oksygen i vann (SEN0237-A)

Sensoren for oppløst oksygen i vann er omtalt i kapittel 6.4 side 170. Les gjerne gjennom teorien i kapittelet før du går i gang med kalibrering og måling av sensoren.

#### D.3.1 Utstyr

Følgende utstyr trengs:

- PC/MAC med programvare
- USB-kabel m/mikro USB
- MKR NB 1500 – mikrokontroller
- Termometer, evt. temperatursensoren DS18B20
- Sensor SEN0237-A med signalbehandlingskort
- Stiftlist (3 stifter)
- Koblingsbrett m/jumpere
- NaOH oppløsning 0,5 mol
- Pipette
- 2 stk. 100mL erlenmeyerkolber m/vann
- 1 stk. luftpumpe for metting av oksygen
- Magnetrører
- Labstativ med klemme for å holde sensoren
- Kjøleskap, eller mulighet for nedkjøling

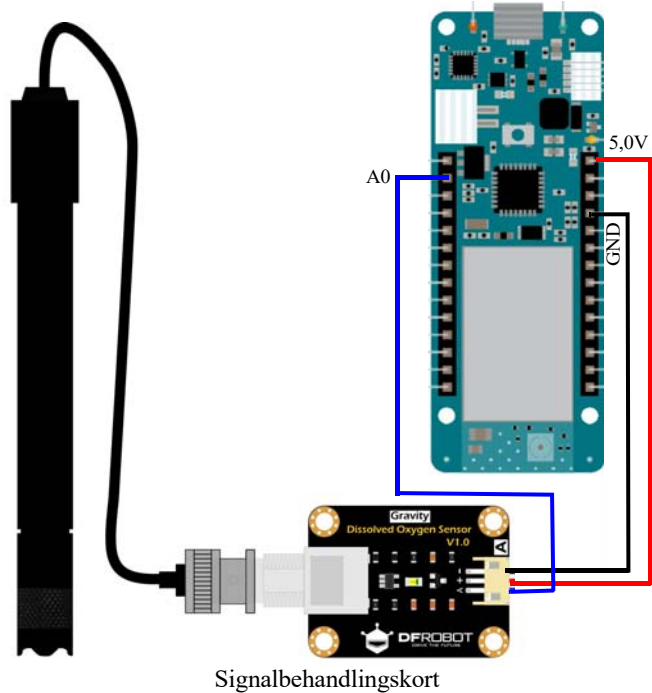
#### Programvare

- Program for avlesning av spenning og temperatur: `Avlesning_av_spenning_temp.ino` (se vedlegg G.2.1 side 271, eller last ned fra nettstedet: [www.ntnu.no/skolelab/bla-hefteserie](http://www.ntnu.no/skolelab/bla-hefteserie)).
- Program for måling av oksygeninnhold i væske: `Oksygen_Maaling.ino` (se vedlegg G.2.2 side 273, eller last ned fra nettstedet: [www.ntnu.no/skolelab/bla-hefteserie](http://www.ntnu.no/skolelab/bla-hefteserie)).



### D.3.2 Oppkobling

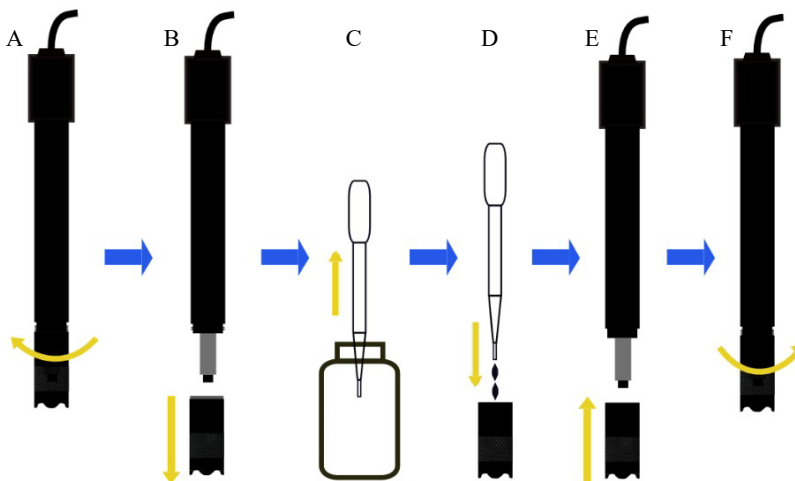
Sensoren med signalbehandlingskort kobles til mikrokontrolleren som vist på figuren til høyre. Legg merke til at signalbehandlingskortet skal kobles til 5,0 V på MKR NB 1500. Så lenge signalspenningen til signalbehandlingskortet ikke overskrider 3,3 V, burde det gå greit å sende spenningen rett til en analog inngang hos mikrokontrolleren.



### D.3.3 Forberedelser

#### Tilsett NaOH oppløsningen til proben

Figuren under viser hvordan vi fyller proben (beholderen) med NaOH oppløsningen. Vær forsiktig da NaOH er etsende, bruk hansker og vernebriller:





- A. Skru av beholderen med membranen foran på proben ...
- B. ... og ta den helt av
- C. Bruk en pipette for å fylle ...
- D. ... membranbeholderen 2/3 full med 0,5mol/L NaOH
- E. Skru membranbeholderen på proben
- F. Det er bra om litt av oppløsningen kommer ut idet beholderen skrur på, dermed vet vi at den er full

Pass på å sette lokk på resten av NaOH oppløsningen slik at man unngår at  $\text{CO}_2$  i lufta forringer oppløsningen. Skyll av overflødig NaOH.

### D.3.4 Kalibrering

Vi velger å utføre en to-punkts kalibrering for å kunne gjøre målinger over et litt større temperaturområde.

Bruk måleprogrammet: "Avlesning\_av\_spenning\_temp.ino"

1. Tilbered to erlenmeyerkolber med rent springvann. Sett den ene kolben i kjøleskapet og den andre til oppvarming i et vannbad. Sørg for at temperaturen ikke overskrider  $40^\circ\text{C}$ .
2. Bruk en av følgende metoder for å lage vann med mettet oksygen. Ta først glasset med høyest temperatur:
  - A. Bruk en magnetrører og la den gå i ca. 10 minutter for å mette vannet med oksygen, eller rør kraftig i vannet med en glass-stav.
  - B. Eller bruk en luftpumpe og pump luft ned i rent vann i 10 min.





3. Stopp røringen eller pumpingen og dypp sensoren sammen med et termometer ned i begeret etter at boblene er forsvunnet og fortsett å røre langsomt i vannet for å unngå at det dannes bobler.

4. Når spenningen er stabil, les av spenningen og temperaturen

Avlest spenning og temperatur:

CAL1\_V = \_\_\_\_\_ mV

CAL1\_T = \_\_\_\_\_ °C

5. Gjør det samme med glasset med det kalde vannet.

Avlest spenning og temperatur:

CAL2\_V = \_\_\_\_\_ mV

CAL2\_T = \_\_\_\_\_ °C

Resultatet kan f.eks. være:

CAL1\_V = 1600mV

CAL1\_T = 25°C

og

CAL2\_V = 1300mV

CAL2\_T = 15°C

6. Sett `#define TWO_POINT_CALIBRATION 1` slik at det gjøres en topunkts kalibrering.

7. Sett kalibreringsverdiene inn i måleprogrammet som vist under i rødt.

Merk at ikke hele programmet er vist:

```
// Oksygen_Maaling inkluderer kalibrering og måling av temperatur
// Programmet bygger på programmet fra DFRobot:
// https://wiki.dfrobot.com/
Gravity__Analog_Dissolved_Oxygen_Sensor_SKU_SEN0237#More
// men er skrevet om for å passe bedre for MKR NB 1500
// Nils Kr. Rossing 16.12.22
.
.
.
//Single-point calibration Mode=0
//Two-point calibration Mode=1
#define TWO_POINT_CALIBRATION 1 // Vi velger to-punkts kalibrering

//Single point calibration needs to be filled CAL1_V and CAL1_T
#define CAL1_V 1600 //mv
#define CAL1_T 25 //C
//Two-point calibration needs to be filled CAL2_V and CAL2_T
//CAL1 High temperature point, CAL2 Low temperature point
#define CAL2_V 1300 //mv
#define CAL2_T 15 //C
```



.  
. .  
.

### D.3.5 Måling av oppløst oksygen

1. Ta så en prøve av f.eks. havvann.
2. Kjør programmet og studer utskriften.

Etter at kalibreringsverdiene fra målingene er lagt inn i programmet og det er kompilert og kjørt, kan vi få ut et resultatet som ligner på denne tabellen:

Temperatur:	25°C	ADC RAW:	215	ADC Voltage:	1049	DO:	5413
Temperatur:	25°C	ADC RAW:	214	ADC Voltage:	1044	DO:	5387
Temperatur:	25°C	ADC RAW:	214	ADC Voltage:	1044	DO:	5387
Temperatur:	25°C	ADC RAW:	213	ADC Voltage:	1040	DO:	5362
Temperatur:	25°C	ADC RAW:	213	ADC Voltage:	1040	DO:	5362

Her er temperaturen i °C, dernest avlest ADC digital verdi, så avlest spenningen i milliVolt, og tilslutt den beregnende oppløste mengden oksygen (DO) i µg/L. Ved å dele på 1000 får vi mg/L.

3. Les av resultatet  
Mål temperaturen: \_\_\_\_\_ °C. Vent til verdien har stabilisert seg og les av målt oksygeninnhold: \_\_\_\_\_ mg/L.
4. Ta gjerne flere målinger



## D.4 Måling av vannets ledningsevne og saltholdighet (DFR0300-H)

### D.4.1 Utstyr

Følgende utstyr trengs:

- PC/MAC med programvare
- USB-kabel m/mikro USB
- MKR NB 1500 – mikrokontroller
- Temperatursensor DS18B20 m/4,7 kOhm motstand
- Sensor DFR0300-H med signalbehandlingskort
- Stiftlist (3 stifter)
- 1 x Koblingsbrett (halv størrelse) m/jumpere
- 1 x Glasskar
- 1 x 100mL erlenmeyerkolber
- 5 x 250mL erlenmeyerkolber
- 1 x 1000mL erlenmeyerkolbe
- Vanlig koksalt (NaCl)
- Vekt (nøyaktighet 0,1 eller 0,01 gram)
- Magnetrorer
- Labstativ med klemme for å holde sensoren

### Programvare

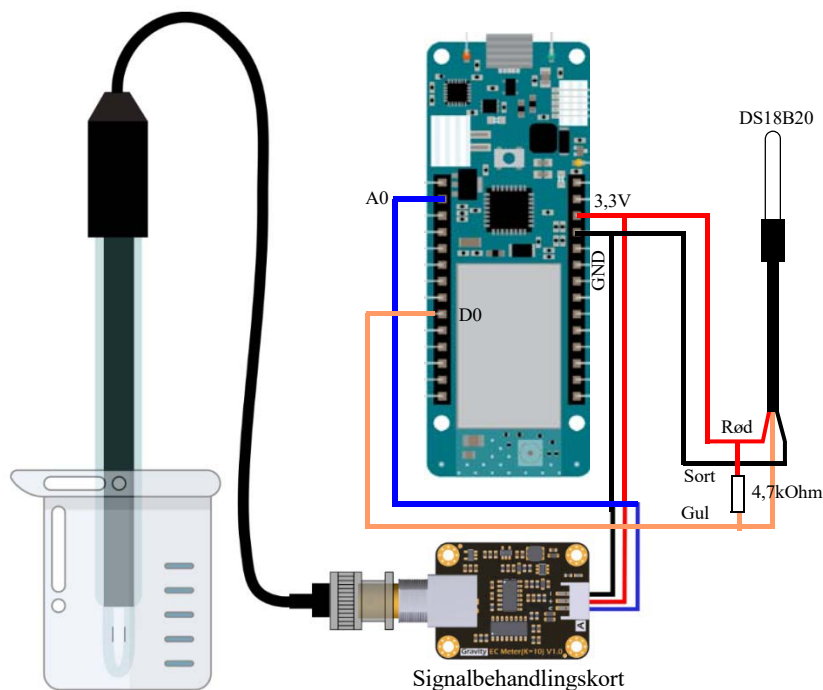
- Program for avlesning av saltholdighet: `Salinitetsmåler_DFR0300_H.ino` (se vedlegg G.3 side 275, eller last ned fra nettstedet: [www.ntnu.no/skolelab/bla-hefteserie](http://www.ntnu.no/skolelab/bla-hefteserie)).





## D.4.2 Oppkobling

Sensoren benytter en drivspenning på 3,3V hvilket passer godt for bruk sammen med MKR NB 1500. I tillegg til sensoren som måler saltholdigheten, vil vi benytte temperatursensoren DS18B20.



NB! Tape gjerne temperatursensoren inntil konduktivitetsensoren slik at temperaturmålingen blir stabil. Dette er særdeles viktig når man skal ta opp temperaturkarakteristikken.

## D.4.3 Forberedelser

### Framstilling av standard blandinger

Lag ca. 200mL av følgende blandinger: 0 %, 8,75 %, 17,5 %, 35 % og 70 %.

Vi tar utgangspunkt i 500mL med 70 % salinitet. En slik kan vi lage på følgende måte:

- Natriumklorid/salt, (NaCl)
- Destillert vann eller milliporevann (vi kan sannsynligvis også bruke springvann)
- Magnetrorer
- Erlenmeyer kolber (5 stk. 250 mL)
- Erlenmeyer kolbe (1 stk. 1000 ml)



Fyll kolben med 500 ml *oppvarmet* destillert vann (eller springvann). Tilsett 33,03 g natriumklorid (NaCl) og bland godt til saltet er oppløst. *Det er viktig å blande ut saltet i varmt vann, dermed løser deg seg lettere opp. Vi har erfart at det kan se oppløst ut, likevel kan saliniteten bli for lav.* Denne løsningen har salinitet på 70 ‰ (ppt) ved 25 °C.

Dernest lager vi de andre konsentrasjonene slik:

1. 500 mL (70 ‰) → 200mL 70,0 ‰
2. 500 mL (70 ‰) → 100 mL + 100 mL rent vann → 200 mL 35,0 ‰ – tilsvarer sjøvann
3. 500 mL (70 ‰) → 50 mL + 150 mL rent vann → 200 mL 17,5 ‰ – tilsvarer brakkvann
4. 500 mL (70 ‰) → 25 mL + 175 mL rent vann → 200 mL 8,75 ‰
5. 200 mL rent vann

Vi har nå de løsningene vi trenger for å utføre kalibrering av sensoren. I tillegg har vi behov for å varme opp eller avkjøle prøvene slik at vi får kalibrert sensoren mht. temperaturvariasjoner.

#### D.4.4 Salinitet som funksjon av målt spenning

Innledende målinger:

1. Last opp måleprogrammet: Salinitetsmåler\_DFR0300\_H.ino (programmet kan lastes ned fra nettsiden [www.ntnu.no/skolelab/bla-hefteserie](http://www.ntnu.no/skolelab/bla-hefteserie)), åpne monitoren slik at spenningen kan leses av.
2. Stikk sensoren ned i rent vann fra springen (eller destilert vann). En slik løsning skal ha en salinitet på 0 ‰. Noter temperaturen i væsken les av og noter spenningen.
3. Mål spenning som funksjon av salinitet for de framstilte løsningene og før resultatet inn i tabell 4 side 247. Husk og noter temperaturen i væsken. Den bør være den samme i alle målingene.

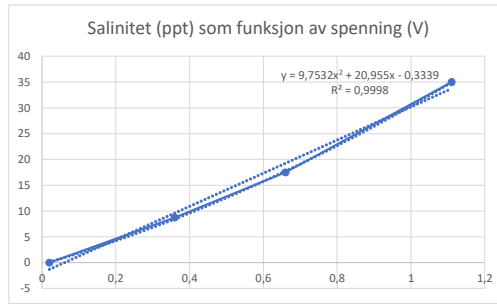
Tabell 4: Spenning målet med forskjellig salinitet.

Salinitet [‰]	Spenning [V]
Temperatur: °C	
0 ‰	V
8,75 ‰	V
17,50 ‰	V
35,00 ‰	V
70,00 ‰	V

4. Legg målingene inn i et regneark og finn en 1. eller 2. grad regresjonsformel for sammenhengen mellom spenning og salinitet.



Eksempelvis som vist i figuren under



Følgende sammenheng ligger som default i programmet:

$$\text{Salinitet} = 9,7532 * \text{Spenning}^2 + 20,955 * \text{Spenning} - 0,3339 \text{ [‰]} \quad (\text{D.1})$$

Sjekk om den stemmer i ditt tilfelle.

Min sammenheng er:

$$\text{Salinitet} = \text{_____} * \text{Spenning}^2 + \text{_____} * \text{Spenning} - \text{_____} \text{ [‰]} \quad (\text{D.2})$$

5. Juster om nødvendig parametrene i formelen i programmet. Husk at denne sammenhengen kun gjelder for temperaturen under målingene. Du finner den på linje 54 og 55 i programmet:

```
// Beregner Salinitet
Salinitet = 9.7532 * UD_mid_volt * UD_mid_volt + 20.955 * UD_mid_volt - 0.3339;
```

6. Gjør en måling av en sjøvannsprøve ved den aktuelle temperaturen og se om den ser rimelig ut. Denne verdien er foreløpig ikke korrigert for variasjoner mht temperatur. Derfor bør denne måles ved samme temperatur som benyttet under kalibreringen foran.

Målt ukorrigert salinitet: \_\_\_\_\_ ‰

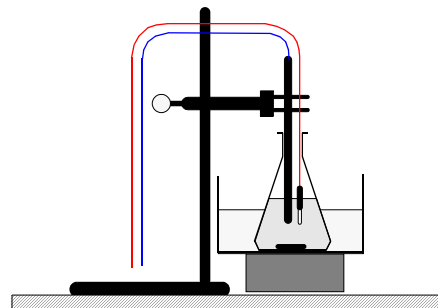
Målt temperatur: \_\_\_\_\_ °C

La oss se nærmere på hvordan målt salinitet varierer med temperaturen.

#### D.4.5 Korrigerings av målt salinitet som funksjon av temperaturen

Vi antar at saltholdigheten i sjøvann er uforandret med temperaturen, og at en evt. variasjon skyldes egenskaper ved sensoren vår og er en uønsket effekt vi gjerne vil korrigere for. Vi må derfor måle ukorrigert salinitet som funksjon av temperaturen.

1. **Sett opp apparaturen** som vist på figuren til høyre og koble den til MKR NB 1500 som vist i avsnitt D.4.2. Tape temperatursensoren inntil salinitetssensoren slik at vi får en stabil måling nær





sentrum av kolben. Løft sensorene litt opp fra bunnen slik at de ikke kommer i konflikt med magnetrøreren. Er man uheldig vil magnetrøreren feste seg til temperatursensoren siden denne har en metallkappe.

- Fyll karet med varmt vann fra springen (ca. 40°C). NB! Det er viktig at vannet ikke er for varmt, da vil oppvarmingen av sjøvannet gå for fort.

### 3. Salinitet som funksjon av temperatur

Fyll sjøvann med lav temperatur (hentet fra kjøleskap ca. +5°C) i erlenmeyer-kolben. Ha minst mulig sjøvann i kolben slik at oppvarmingen skjer raskest mulig.

Gjør målinger hvert 20 sek. (eller oftere etter behov) og skriv resultatet ut i tekstmonitoren hos Arduino IDE'en. Til dette formålet kan det være lurt å organisere utskriften slik at den er lett å hente inn i et regneark.

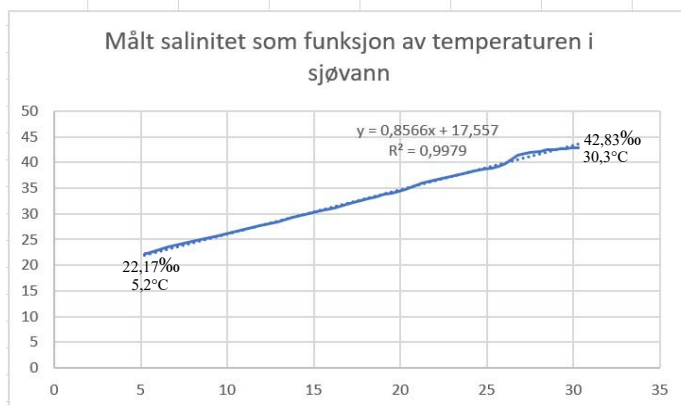
```
//Serial.print(", Midlere temperatur: "); // Skriv temperaturen:
//Serial.print(";");
Serial.print(Temperature,1);

//Serial.print(", Ukorrigert salinitet: "); // Skriv ut den avleste verdien:
Serial.print(";");
Serial.println(Salinitet,2);

//Serial.print(", Korrigert salinitet: "); // Skriv ut den avleste verdien:
//Serial.print(";");
//Serial.println(korrSalinitet,2);
```

La den gå til den har nådd ca. 25°C, evt. mer om tiden tillater det.

- Legg resultatet over i et regneark og lag en graf. Eksempelvis som vist under:



- Gjør en regresjon og finn en optimal funksjon som beskriver salinitet som funksjon av temperaturen. En lineær sammenheng synes å være tilstrekkelig.

Eksempelvis som vist på figuren over, der sammenhengen kan beskrives som:

$$\text{Salinitet (T)} = 0,8566 * \text{Temperatur [C]} + 17,557 [\text{‰}] \text{ for } [5,2 - 30,3^{\circ}\text{C}] \quad (\text{D.3})$$



Det er også denne som er default i programmet. Siden alle målingene er gjort ved samme temperatur (eksempelvis 20°C) og vi har tidligere målt en kalibrert saliniteten ved denne temperaturen (eksempelvis 34,3‰), så finner vi korreksjonen ved å trekke fra den kalibrerte målingen for saliniteten (eksempelvis 34,3‰). I vårt eksempel får vi da:

$$\text{Korreksjon (T)} = 0,8566 * \text{Temperatur [C]} + 17,557 - 34,3 \text{ [‰]} \quad (\text{D.4})$$

Sett opp din egen korrigerende sammenheng:

$$\text{Korreksjon (T)} = \text{_____} * \text{Temperatur [°C]} + \text{_____} - \text{_____} \text{ [‰]} \quad (\text{D.5})$$

## 6. Beregn korrigert salinitet

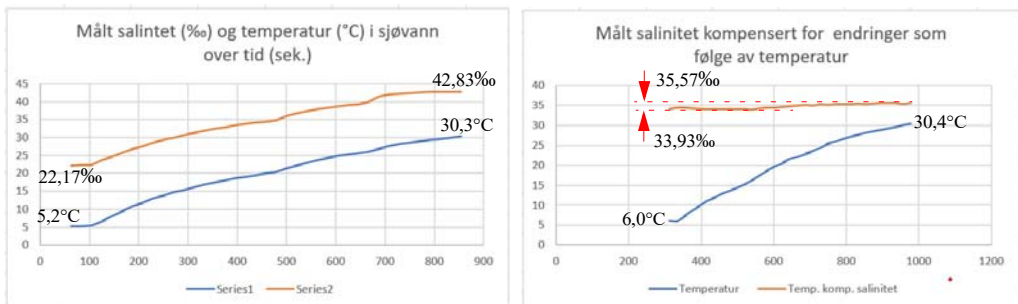
Salinitet korrigert mht temperaturen finner vi da på følgende måte:

Korrigert salinitet = Salinitet – Korreksjon

I programmet ser dette slik ut:

```
// Beregner Salinitet
Salinitet = 9.7532 * UD_mid_volt * UD_mid_volt + 20.955 * UD_mid_volt - 0.3339;
korreksjon = 0.8566*(Temperature) + 17.557 - 34.3;
korrSalinitet = Salinitet - korreksjon;
```

I vårt eksempel fikk vi følgende resultater før og etter temperaturkompensasjon:



7. Kjør programmet og se om korreksjonen fungerer for standardverdiene saltholdig vann.



## D.5 Måling av turbiditet i vann (SEN-0189)

Måling av turbiditet handler om å måle tettheten av partikler i vann. Les gjerne teorien knyttet til denne sensoren som er omtalt avsnitt 6.6 side 193.

### D.5.1 Utstyr

Følgende utstyr trengs:

- PC/MAC med programvare
- USB-kabel m/mikro USB
- MKR NB 1500 – mikrokontroller
- Turbiditetssensor SEN-0189
- Stiftlist (3 stifter)
- 2 x 10kOhm 1% motstander
- 1 x koblingsbrett (mini)
- Div. jumpere
- Kyvettekammer for kalibrering
- Formazin for kalibrering (500 NTU)
- Destillert vann
- 3 x 100mL erlenmeyerkolber

### Programvare

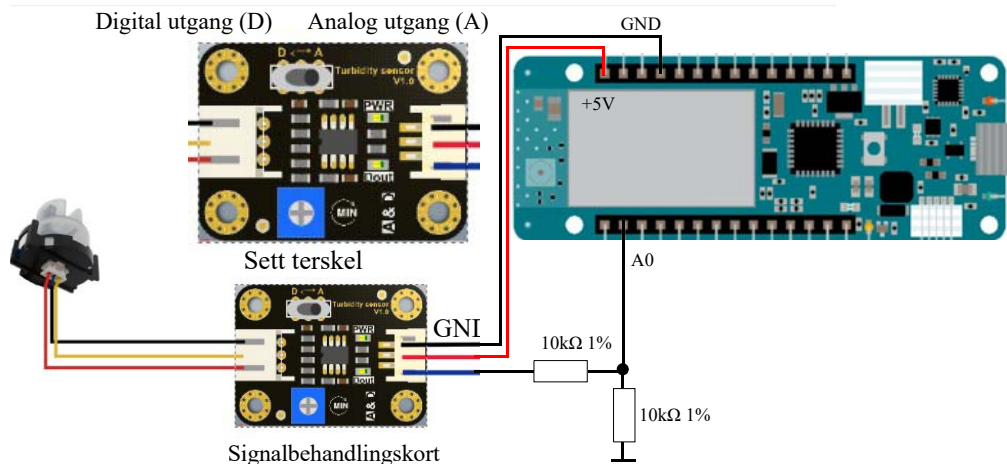
- Program for avlesning av turbiditet: `Turbidimeter-SEN0189_Kal-1.ino` (se vedlegg G.4.3 side 279, eller last ned fra nettstedet: [www.ntnu.no/skolelab/bla-hefteserie](http://www.ntnu.no/skolelab/bla-hefteserie)).

### D.5.2 Oppkobling

Figuren under viser oppkoblingen til en MKR NB 1500. Siden MKR NB 1500 benytter en arbeidsspenning på 3,3 V må man sørge for at analoginngangen til MKR ikke overskrider 3,3V.

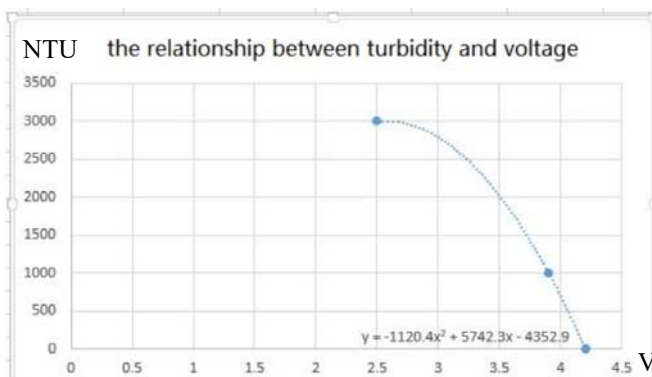


Dette gjør vi ved hjelp av spenningsdeler som halverer spenningen. Deretter må vi passe på å doble spenningen når vi bruker den i programmet vårt.



Ved hjelp av en bryter D/A kan vi velge mellom analogt og digitalt signal på utgangen.

A – Sett bryteren i posisjon A, vil utgangen levere en spenning som faller med økende turbiditet (økende tetthet av partikler). Grafen i figuren under antyder sammenhengen mellom turbiditet og utgangsspenning.



Vi merker oss at spenningen i diagrammet kan være i området fra ca. 4,2 til 2,5V for verdier av NTU fra 0 – 3000, som er i meste laget for Arduino MKR, hvilket betyr at vi må bruke en spenningsdeler. Vi velger å halvere spenningen med to motstander a 10kΩ.

Som vi ser også at sammenhengen mellom spenning og turbiditet er temmelig ulineær. Det er kun for lave turbiditetsverdier, 0 – 1500 NTU, at sammenhengen er tilnærmet lineær.

Måleenheten NTU står for *Nephelometric*<sup>126</sup> *Turbidity Units* og er vanlig brukt i USA. 1 NTU tilsvarer ca. 1 mg/L. Vi har valgt å bruke den her.

126.Nephel kommer av gresk og betyr tåke eller tåkete. Vi finner det samme ordet i Nebula, stjernetåke



### D.5.3 Forberedelser

#### Framstilling av standarder

Vi ønsker å bruke standarder på 0, 200 og 400 NTU. Disse må vi lage fra 500 NTU. 0 NTU får ved å bruke destillert vann eller rent springvann.

Vi har en flaske med Farmazin med en turbiditet på 500 NTU. Vi ønsker å tilsette vann slik at vi får en NTU på henholdsvis 400 og 200 NTU. Vi bruker Farmazin som inneholder små partikler. Her må merke oss at partiklene har en tendens til å falle mot bunnen. *Det er derfor viktig at vi rister den godt før vi heller over i mindre kolber eller begerglass.*

Vi ønsker en sluttmengde på  $T_{mL} = 50$  mL av 200 og 400NTU. Spørsmålet er da hvor mye vi må tilsette av Farmazin ( $F_{mL}$ ) og destillert vann ( $V_{mL}$ ).

Vi kan sette opp følgende sammenheng:

$$\frac{\text{Ønsket volum fortynnet standard} \times \text{Ønsket NTU-verdi}}{\text{NTU for standard som skal blandes ut}} = \text{Volum tilsatt av opprinnelig standard}$$

I vårt tilfelle blir beregningen for å lage en standard på 400 NTU lik:

$$\frac{50\text{mL} \times 400}{500} = 40\text{mL}$$

Dvs. at oppløsningen på inneholde 40 mL av standarden på 500 NTU og 10 mL destillert vann.

Tilsvarende kan vi gjøre følgende beregning for å lage en standard på 200 NTU:

$$\frac{50\text{mL} \times 200}{500} = 20\text{mL}$$

Dvs. at oppløsningen må inneholde 20 mL av standarden på 500 NTU for så å tilsette 30 mL destillert vann.

### D.5.4 Kalibrering

Vi bruker følgende metode for å kalibrere og måle med sensoren:

1. Bruk programmet: *Turbidimeter-SEN0189.ino*
2. Vi bruker en 3D-printet ugjennomsiktig boks for å hindre lekkasje av lys. Boksen passer akkurat sensoren som vist på figuren til høyre.
3. Fyll boksen med kalibreringsvæske med NTU 400. *Unngå at det kommer væske inn i sensor-kammeret.* Mål spenningen og noter verdien:

\_\_\_\_\_ V

4. Fyll boksen med kalibreringsvæske med NTU 200. Mål spenningen og noter verdien:

\_\_\_\_\_ V







5. Fyll boksen med destillert vann og noter måleverdien:

\_\_\_\_\_ V.

6. Skriv verdiene inn i tabellen:

Kalibrert oppløsninger [NTU]	Spenning [V]
Under kalibrering	
Kalibrerings medium	Spenning
0 NTU (rent vann)	_____ V
200 NTU	_____ V
400 NTU	_____ V
Måling i etterkant	
Prøve	Målt verdi (NTU)
spring vann	_____ NTU
200 NTU	_____ NTU
400 NTU	_____ NTU
500 NTU	_____ NTU

7. Åpne måleprogrammet: *Turbidimeter\_SEN0189\_Kal-1.ino*

8. Skriv de verdiene (NTU og spenningene) inn i programmet og gjør målinger på NTU 200 og NTU 400 og sjekk at verdiene blir som under kalibreringen. Verdiene som ligger i programmet skrives over:

```
float NTU      = 0;    // Målt og beregnet NTU-verdi
float NTU_200  = 200;  // Kalibrert NTU-verdi 200 NTU.
float NTU_400  = 400;  // Kalibrert NTU-verdi 400 NTU.
float U_0      = 4.36; // Målt spenningsverdi ved NTU-verdi 400
float U_200    = 4.14; // Målt spenningsverdi ved NTU-verdi 200
float U_400    = 3.97; // Målt spenningsverdi ved NTU-verdi 400
```

9. Mål standardene på nytt og se hvor nær de kommer de virkelige verdiene

10. Mål en uutblandet prøve fra Fermazin flaska (500 NTU) og se om det virker rimelig.



## D.6 Måling av pH (SEN-0161)

Måling av pH er blitt mer og mer viktig da man observerer at økende opptak av CO<sub>2</sub> gir et surere hav. Det er imidlertid snakk om små endringer. Les gjerne teorien knyttet til denne sensoren som er omtalt avsnitt 6.7 side 202.

### D.6.1 Utstyr

Følgende utstyr trengs:

- PC/MAC med programvare
- USB-kabel m/mikro USB
- MKR NB 1500 – mikrokontroller
- pH-sensor SEN-0161
- Signalbehandlingskort med kabel
- Stiftlist (3 stifter)
- 2 x 10kOhm motstander
- 1 x koblingsbrett (mini)
- Div. jumpere
- 1 x Standard Buffer-løsning på pH 7,0
- 1 x Standard Buffer-løsning på pH 4,0
- 1 x Standard Buffer-løsning på pH 9,18
- Destillert vann i spruteflaske
- 1 x 100mL erlenmeyerkolber

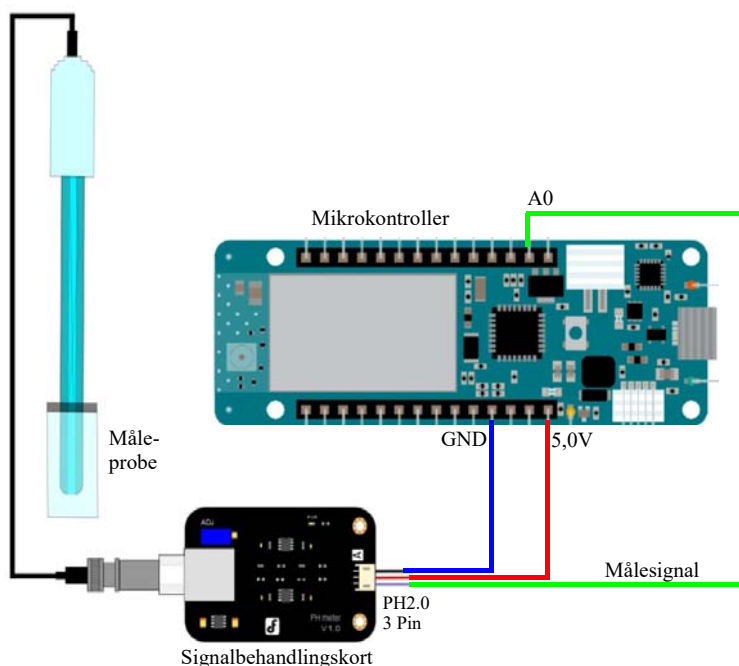
### Programvare

- Program for avlesning av pH: `pH-sensor_SEN161_utvidet-1.ino` (se vedlegg G.5 side 279, eller last ned fra nettstedet: [www.ntnu.no/skolelab/bla-hefteserie](http://www.ntnu.no/skolelab/bla-hefteserie)).



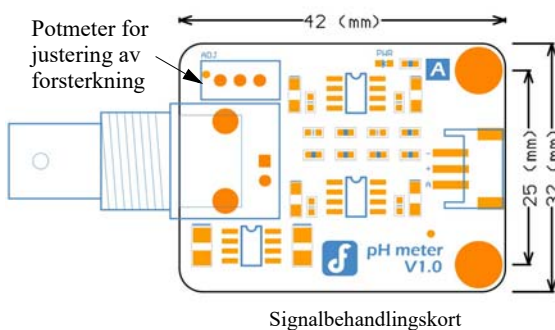
## D.6.2 Oppkobling

pH-sensoren skal ha en forsyningsspenning på 5,0V dermed må vi sørge for at spenningen inn på A0 ikke overskrider 3,3V. Det må brukes koblingsbrett mellom signalbehandlingskortet og mikrokontrolleren..



### Pass på følgende:

- Unngå at signalbehandlingskortet blir fuktig. Et fuktig kort vil endre målebetingelsene og gi avvik i målingene.
- Foran i proben sitter en liten glasskule som ikke må skades, noe som lett kan skje dersom kula støtes mot harde overflater.
- Pass på at proben ikke er tilkoblet signalbehandlingskortet over lengre tid uten at det står spenning på kortet.
- Når sensoren ikke brukes skal proben beskyttes av en hette fylt med 3,3 mol/L KCl.



## D.6.3 Forberedelser

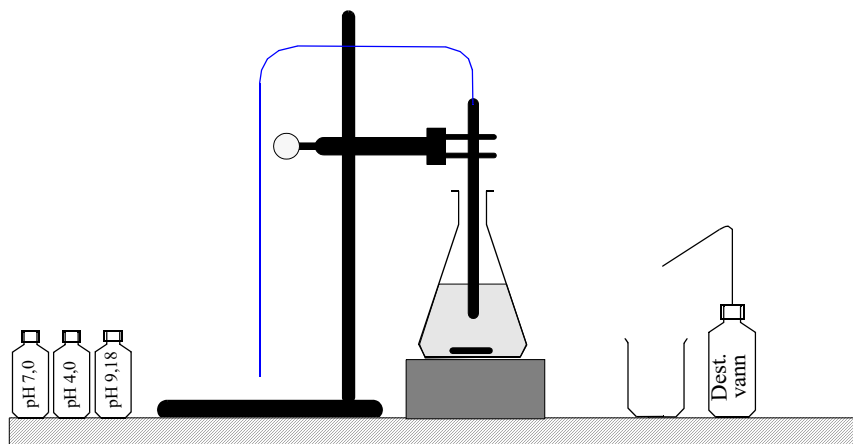
1. Les gjennom det som står om måling med pH-sensoren i kapittel 6.7 side 202



2. Vi har laget fire løsninger som skal brukes ved kalibrering:
  - 3,3 mol kaliumklorid oppløsning (KCl)
  - 1 x bufferløsning med pH 4,0
  - 1 x bufferløsning med pH 7,0
  - 1 x bufferløsning med pH 9,18

3. Sett opp måleutstyret

Sett opp måleutstyret som vist på figuren under. Husk å ta av hetta som sitter på sensoren.



Pass på å bruk magnetrøreren flittig. Likeså destillert vann til å skylle sensoren mellom hvert måling.

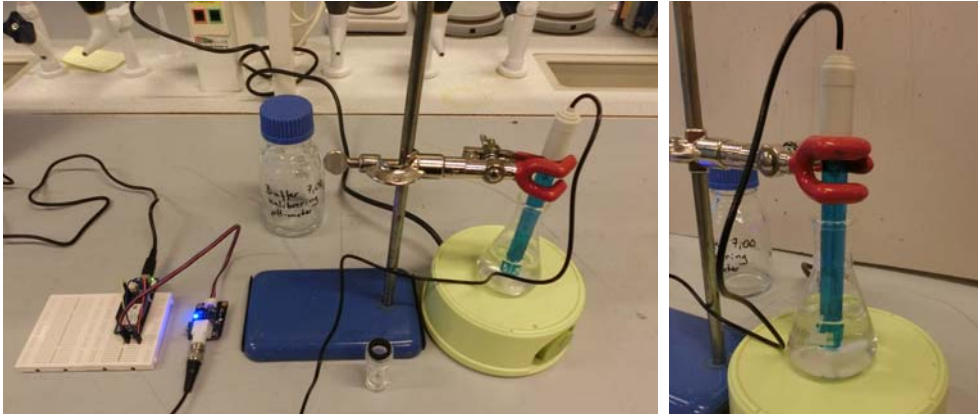
#### D.6.4 Kalibrering

Vi skal nå gå gjennom kalibrering av sensoren.

1. Ta en kolbe og en magnetrører og fyll den med bufferløsning med pH7. Sett den til røring.



2. Monter pH-sensoren i laboratoriestativet og la den stå neddyppet i bufferen over noe tid slik at måleverdien stabiliserer seg. La sensoren være oppkoblet til mikrokontrolleren og observer pH-verdien.



3. Åpne monitoren og les av pH-verdien. Denne skal være innenfor  $\text{pH } 7,0 \pm 0,3$ . La oss anta at den viser  $\text{pH } 6,88$ . Dvs. vi registrerer et avvik på  $-0,12$ .
4. Korriger for dette avviket ved å sette verdien  $0,12$  inn i programmet der det står:  

```
#define Offset 0.0 //Kompenser for avviket
```
5. Dersom målingene viser verdier over  $7,0$  så legges offset inn med negativt fortegn  
Dersom den er under  $\text{pH } 7,0$  så legges korreksjonen inn med positivt fortegn
6. Tøm ut bufferen og skyll kolben med litt av den nye bufferen som skal brukes i kalibreringen, f.eks.  $\text{pH } 9,18$ . Bruk den samme bufferen og skyll proben, og tøm ut.
7. Fyll opp med buffer med  $\text{pH } 9,18$ , rør godt og stikk proben ned i kolben og vent til måleverdien har stabilisert seg.
8. Korriger verdien med å skru på forsterkningen (potensiometeret).



### D.6.5 Måling av pH i sjøvann

Bruk den kalibrerte sensoren til å måle pH til sjøvann, husk å registrer temperaturen:

pH sjøvann: pH \_\_\_\_\_

Temperatur: \_\_\_\_\_ °C



## D.6.6 Variasjon av pH som funksjon av temperaturen

Når temperaturen i en løsning stiger, stiger de molekylære vibrasjonene i løsningen, noe som resulterer i ionisering og dannelse av  $H^+$ -ioner. Flere  $H^+$  ioner fører til surere oppførsel. På grunn av temperaturoendringer endres pH-verdien til løsningen slik at pH synker ved økende temperatur.

Det er viktig å være klar over at enhver løsning vil gjennomgå en endring i pH-verdien som funksjon av temperaturen. En forskjell i pH-målinger ved forskjellige temperaturer er imidlertid IKKE en feil! Det nye pH-nivået forteller ganske enkelt om den sanne pH-verdien for den løsningen ved den spesifikke temperaturen, som vi ser av tabellen<sup>127</sup> under.

T (°C)	$K_w$ ( $mol^2 dm^6$ )	pH
0	$0.114 \times 10^{14}$	7.47
25	$1.008 \times 10^{14}$	7.00
50	$5.476 \times 10^{14}$	6.63
100	$51.3 \times 10^{14}$	6.14

Bruk kalkulatoren på følgende nettside og beregn variasjonen i pH over et passende temperaturområde:

<https://www.hamzasreef.com/Contents/Calculators/PhTempCorrection.php>

Kalkulatoren beregner pH ved 25°C når målingene er gjort ved lavere temperaturer, hvilket vil være aktuelt for vårt tilfelle. Som det framgår av tabellen ser vi at en måling nær en pH på 8,1 så vil variasjonen være på 0,08 over et temperaturområde på 25°C

Skriv verdiene inn i tabellen:

pH-målt ved ...	...temperatur	Omregnet til pH @ 25°C
pH 8,1	0°C	pH
pH 8,1	2°C	pH
pH 8,1	5°C	pH
pH 8,1	10°C	pH
pH 8,1	15°C	pH
pH 8,1	20°C	pH

Gjør en kontrollmåling av sjøvann ved en temperatur som er vesentlig forskjellig fra den du opprinnelig hadde.

<sup>127</sup><https://www.westlab.com/blog/2017/11/15/how-does-temperature-affect-ph>



## D.7 Måling av ORP (SEN-0165)

ORP (Oksidasjon-Reduksjons-Potensial) er et mål for væskens evnen til oksidasjon og reduksjon. I motsetning til en pH-måling som følger en logaritmisk kurve og derfor krever flere kalibreringsjusteringer, følger ORP et lineært forløp.

*ORP har vist seg å være en pålitelig metode for å måle vannkvalitet og gir en enkelt måleverdi uavhengig av hvilket produkt eller desinfiseringsmiddel som er brukt, den er også uavhengig av varierende feltforhold.*

Vi har tatt med denne sensoren da det kan være aktuelt å måle kvaliteten i ferskvann. Les gjerne teorien knyttet til denne sensoren som er omtalt avsnitt 6.8 side 211.

### D.7.1 Utstyr

Følgende utstyr trengs:

- PC/MAC med programvare
- USB-kabel m/mikro USB
- MKR NB 1500 – mikrokontroller
- ORP-sensor SEN-0165
- Signalbehandlingskort med kabel
- Stiftlist (3 stifter)
- Koblingsbrett (halv størrelse) m/jumpere
- Destillert vann i spruteflaske
- 1 x 100mL erlenmeyerkolber

#### Programvare

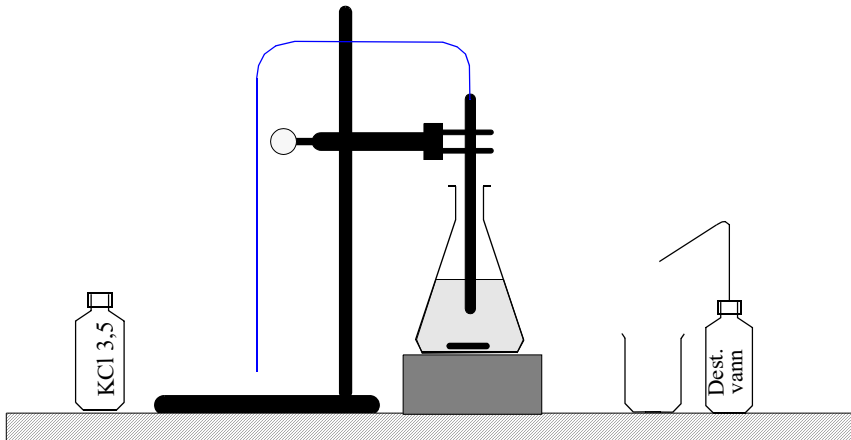
- Program for avlesning av oksidasjon-reduksjons-potensialet (ORP): ORP-sensor\_-SEN0165.ino (se vedlegg G.6 side 282, eller last ned fra nettstedet: [www.ntnu.no/skolelab/bla-hefteserie](http://www.ntnu.no/skolelab/bla-hefteserie)).



## D.7.2 Forberedelser

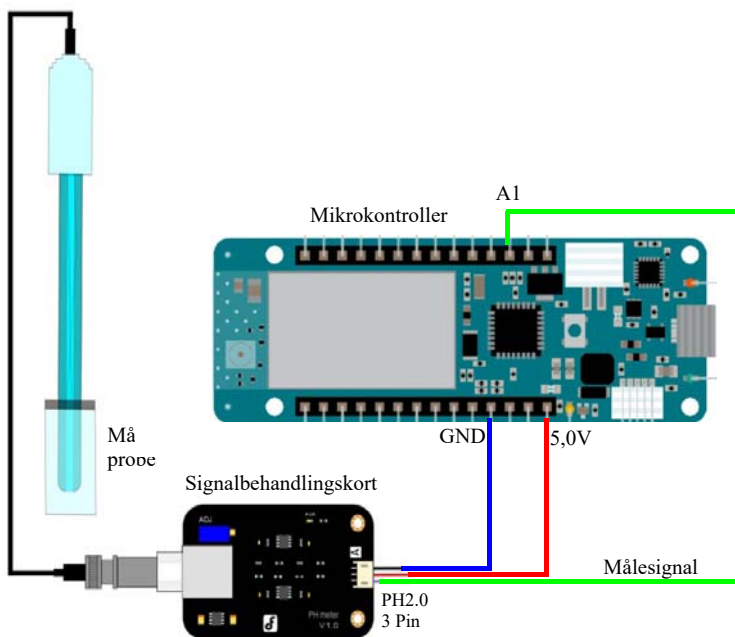
### Monter måleoppsett

Monter sensoren i et laboratoriestativ med magnetrører som vist i figuren under.



## D.7.3 Oppkobling

ORP-sensoren skal ha en forsyningsspenning på 5,0V, dermed må vi sørge for at spenningen inn på A0 ikke overskrider 3,3V. Det må brukes koblingsbrett mellom signalbehandlingskortet og mikrokontrolleren..







## D.7.4 Kalibrering og måling

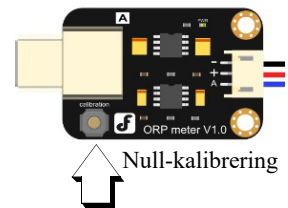
Normalt kan man bruke sensoren uten kalibrering. Dersom det er mistanke om avvik bør man bruke standard løsningen for å sjekke at spenningen er som forventet. Husk å kompensere for temperatur.

**NB!** Ikke trykk knappen for kalibrering når proben er tilkoblet, da dette kan ødelegge proben.

1. Vask proben med ionebyttet vann (deionized).
2. Koble signalkortet opp til Arduino-kortet som vist i avsnitt D.7.3 side 261. **NB!** Vent med å koble til selve proben. Når signalbehandlingskortet har spenning skal det lyse en blå diode.
3. Kompiler og last opp programkoden: *ORP-sensor\_SEN0165.ino* til mikrokontrolleren. Legg spesielt merke til kodelinjen:

```
#define OFFSET 0
```

4. Åpne monitoren i Arduino IDE og du vil se den målte ORP-verdien bli skrevet ut. Trykk på kalibreringsknappen på signalbehandlingskortet. Kalibreringsknappen legger inngangen fra måleproben til jord. Dette er årsaken til at dette ikke bør gjøres mens proben er tilkoblet. Registrer utskriften i monitoren og bruk denne til å korrigere offset i programmet. Viser monitoren en ORP på 8mV, så går man inn i programmet og endrer:



```
#define OFFSET 8
```

Kompiler og last opp programmet på nytt etter at du har gjort endringen.

5. Nå er sensoren kalibrert og vi kan koble til proben ved hjelp av BNC-pluggen og lese av ORP-verdien i seriemonitoren. ORP-verdien måles i mV. Husk å sett monitoren til 9600 Baud som er den datahastigheten mikrokontroller sender på.

Før kalibrering kan det være ganske store avvik fra 0mV når man trykker kalibreringsknappen.

### 6. Utfør målinger

Vi er nå klare til å måle ORP-verdien i sjøvann og f.eks. destillert vann:

ORP-verdi destillert vann: \_\_\_\_\_ mV. Kompensert for temp: \_\_\_\_\_ mV  
Temperatur: \_\_\_\_\_ °C ved 25°C

Så måler vi sjøvann:

ORP-verdi sjøvann vann: \_\_\_\_\_ mV Kompensert for temp: \_\_\_\_\_ mV  
Temperatur: \_\_\_\_\_ °C ved 25°C



Tabellen til høyre viser hvordan den målte spenningen varierer med temperaturen ved bruk av en standard løsning på 3,5mol/L KCl.

222mV ± 15mV (25°C)  
(3.5mol/L KCL)

°C	mV	°C	mV
10	242	30	215
15	235	35	209
20	227	38	205
25	222	40	201

### D.7.5 Behandling av proben etter bruk

Vask proben i ionebyttet vann (destillert vann) flere ganger etter bruk. Om nødvendig bruk lunkent vann.

### D.8 Måling av trykk og beregning av dybde (SEN-0257)<sup>128</sup>

Vanntrykksmåleren SEN-0257 måler vanntrykk fra 0 – 1,6 Mpa. Vanntrykket øker med ca. 100kPa pr. 10 meter og kan da i prinsippet måle dybder fra 0 – 150 meter og noe mindre i saltvann siden egenvekten for saltvann er noe høyere (1,020 to 1,029 kg/L).

Les gjennom teorien for sensoren på side 6.9 side 215.



#### Utstyr

- PC/MAC med programvare
- USB-kabel m/mikro USB
- MKR NB 1500 – mikrokontroller
- ORP-sensor SEN-0165
- Signalbehandlingskort med kabel
- Koblingsbrett (halv størrelse) m/jumpere
- Stiftlist (3 stifter)
- Stor sylinder av acryl
- Springvann
- Målestav (målbånd)

<sup>128</sup><https://www.dfrobot.com/product-1675.html>



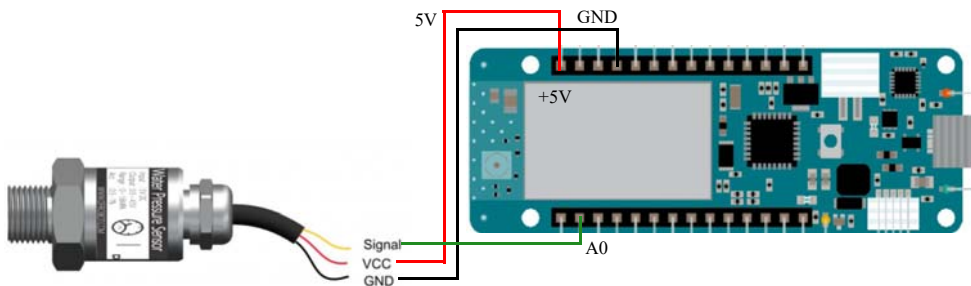
## Programvare

Program for kalibrering, trykkmåling og beregning av dybde: Trykk-sensor\_SEN0257.ino (se vedlegg G.7 side 287, eller last ned fra nettstedet: [www.ntnu.no/skolelab/bla-hefteserie](http://www.ntnu.no/skolelab/bla-hefteserie)).

## Oppkobling

Koble sensoren opp til mikrokontrolleren som vist på figuren under. Sensoren er normalt utstyrt med hylskontakter og med forskjellig fargete ledninger: Rød (+5V), Sort (GND) og Gul (Signal). Her har vi valgt å føre den gule videre som grønn ledning på tegningen, for å gjøre den mer synlig. For å kunne koble oss til et koblingsbrett, så bruker vi en liten stiftlist med tre pinner. Dermed kan vi koble hylsekontakten til koblingbrettet.

For enkelhetsskyld bruker vi den analoge inngangen, A0, hos mikrokontrolleren.



## Last opp måleprogrammet og kalibrer sensoren

Vi skal nå kalibrere sensoren og lese av dybden:

- Fyll sylindere opp med springvann
- Kompiler, last opp og kjør måleprogrammet
- Dypp sensoren så vidt under vann. Sørg for at all luft er ute av sensoren
- Åpne monitorvinduet og les av spenningen (Voltage).

```
12:20:42.987 -> Voltage:0.781V
12:20:42.987 -> Pressure:2.7kPa
12:20:42.987 -> Water depth:0.27m
```

Dette er spenningen ut av sensoren når den ligger i vannskorpa og er stort sett bestemt av lufttrykket over vannet.

Målt spenning: \_\_\_\_\_ V.

- Legg den avleste spenningen inn i programmet på følgende linje  
`const float OffSet = 0.774; // See Calibration Note!`  
På denne måten korrigeres for lufttrykket over vannflata og avvik i elektronikken. Legg programmet med den nye verdien, inn i mikrokontrolleren på nytt.
- Senk sensoren ned så langt som ledningen tillater.  
Gjør nye målinger og noter den målte dybden:  
Dybde målte med sensoren: \_\_\_\_\_ cm



- 
- Mål avstanden fra overflata ned til sensoren: \_\_\_\_\_ cm
  - Sjekk om dybden målt med målebånd stemmer med målingen gjort med sensoren. Juster følsomheten Sens slik at det blir overensstemmelse mellom målt verdi med sensoren og med målebånd: \_\_\_\_\_ kPa/V
  - Studer hvordan måleverdien endrer seg over tid og diskuter ev. avvik og drift:  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_
  - Gå grundig gjennom programmet. Vær sikker på at du skjønner hvert tinn i programmet.
  - Lag en kort oppsummering av kalibrering og måling for presentasjon på slutten av dagen.



## Vedlegg E Omregning fra ledningsevne til saltholdighet

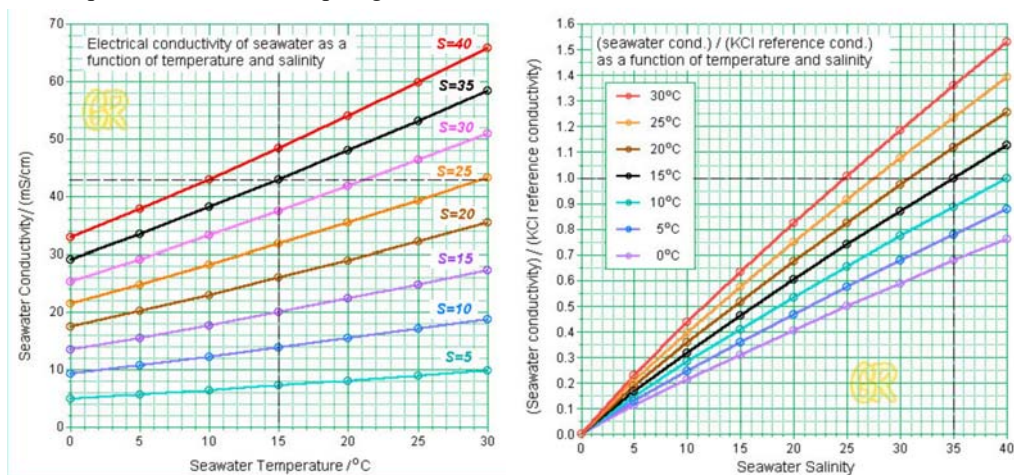
### E.1 Ledningsevne som funksjon av saltholdighet, temperatur og trykk

For ev.- å lette utregningen av saltholdighet (salinitet) på bakgrunn av ledningsevnen er det utviklet flere kalkulatorer.

Conductivity: <input type="text"/> (uS/cm)	Conductivity (mS/cm) = <input type="text" value="0"/>
Water temperature: <input type="text"/> (C)	Temperature (ITS-90) = <input type="text" value="0"/>
<input type="button" value="Calculate"/>	Sea Pressure (dBar) = <input type="text" value="0"/>
Salinity: <input type="text"/> (ppt)	Practical Salinity (PSS-78) = <input type="text"/>
	<input type="button" value="Calculate"/>

Figuren over viser to eksempler på slike kalkulatorer. Mens den til venstre har ledningsevne og temperatur som parametere<sup>129</sup>, så har den til høyre også med trykk (dBar)<sup>130</sup>.

Ved bruk av kalkulatorer som disse kan man lage kurveskarer som viser sammenhengen mellom de ulike parametrene som vist på figuren under<sup>131</sup>:



Grafene til venstre viser ledningsevne som funksjon av temperaturen i °C for ulike verdier av salinitet. Grafene til høyre viser relativ ledningsevne som funksjon av salinitet for ulike temperaturer. Referanseløsningen er laget av KCl og har en salinitet på 35 ppt ved 15°C<sup>132</sup>.

For ytterligere diskusjon av salinitet, se N.K. Rossing, "MauSea – trykk- og temperaturmåling under vann" rev. 4.6

129.<http://salinometry.com/ctd-salinity-calculator/>

130.<http://www.fivecreeks.org/monitor/sal.shtml>

131.<http://www.grabovrat.com/handbook/handbookH08g1.html>

132. Standard vannprøven er laget ved å blande ut 32.4356 gram KCl i 1 liter destillert vann som gir akkurat 35‰ ved 15°C og 1 atm.



## Vedlegg F Python-programmer for presentasjon av måleresultater

### F.1 Grunnleggende program for presentasjon av data<sup>133</sup>

Programmet viser hvordan Python kan brukes til å hente måleresultater fra en file, legge dataene inn i lister og plote dem i grafiske diagrammer. Filen ligger på en server på Inst. for marin teknikk. I Python er innrykk viktig for at programmet skal fungere som tiltenkt, derfor er det viktig at dette blir riktig. Det beste er derfor å hente fila fra:

[www.ntnu.no/skolelab/bla-hefteserie](http://www.ntnu.no/skolelab/bla-hefteserie)

Gå ned til overskriften: *Programmering av Tingenes internett (IoT)*

Åpne fanen: *EVU-kurs: Miljøovervåking med havbøye*

Fila: *grafisk fremstilling av septtest\_5.py*

```
import numpy as np
import numpy as np
import matplotlib.pyplot as plt
import datetime as dt
import pandas as pd

### Valg av tidsvindu for plotting av data

vintertid=False

### import og lagning av arrays

#Henter inn fila som en dataframe i pandas
data = pd.read_csv('http://sensor.marin.ntnu.no/logs/SeptKursTest_5.txt',
delimenter=',')

# Tar ut de kolonnonene som jeg vil bruke fra data og lager egne arrays (numpy
lister) av dem data blir lagret som float
# Kolonnene hentes ut ved overskrift (første data i kolonnen).

epochTime = data["1686171837"].values
templuft = data['23.7'].values
tempvann = data['19.7'].values
tempfukt = data['39.1'].values
light = data['193.55'].values
```

---

133. Programmet er skrevet av Thor Inge Hansen ved Skien vgs.



```
volt = data["2.52"].values
runTime = data['26485'].values/1000 #gjør ms om til s
sumRunTime = sum(runTime)

### Funksjon for å lage datetime-objeter fra epochTime

def epochToDatetime(x):
    liste=[]
    for i in range(len(x)):
        if vintertid:
            t=dt.datetime.utcfromtimestamp(x[i]) + dt.timedelta(hours=1)
        else:
            t=dt.datetime.utcfromtimestamp(x[i]) + dt.timedelta(hours=2)
        liste.append(t)
    return liste

### Anvender funksjonen ovenfor

timelist=epochToDatetime(epochTime)

### Regning av gjennomsnittlig intervall

interval=np.zeros(len(timelist))

for i in range(len(epochTime)-1):
    interval[i]=int((epochTime[i+1]-epochTime[i]))

interval_mean=np.mean(interval[0:len(interval)-1])

### Finner antall ganger intervall mellom sending er over 150 s
count = 1

for i in range (len(interval)):
    if interval[i]>600: #1200sek =20min = feilsending
        count+=1

### Plotfunk

plt.figure(1,figsize=(11,8))
plt.plot(timelist,volt,'b.')
```



```
plt.ylabel('Spenning[V]')
plt.xlabel('Tid')
plt.title('Spenning (1S3P 2500mah liPo)')
plt.ylim(3.2,4.2)
plt.grid()

plt.figure(2,figsize=(11,8))
plt.plot(timelist,light,'b.')
plt.ylabel('Lux')
plt.xlabel('Tid')
plt.title('Lysstyrke')
plt.ylim(0,800)
plt.grid()

plt.figure(3,figsize=(11,8))
plt.plot(timelist,templuft,'r.')
plt.plot(timelist,tempvann,'b.')
plt.xlabel('Tid')
plt.ylabel('Temperatur')
plt.title('Temperaturmålinger')
plt.grid()

plt.figure(4,figsize=(11,8))
plt.plot(timelist,tempfukt,'b.')
plt.xlabel('Tid')
plt.ylabel('Luftfuktighet i %')
plt.title('Relativ luftfuktighet')
plt.grid()

plt.figure(5,figsize=(11,8))
plt.plot(timelist,runTime,'b.')
plt.ylim(0,200)
plt.title('runTime')
plt.xlabel('Sendingsnr.')
plt.ylabel('RunTime for programmet [s]')
plt.grid()

print('Antall målinger så langt',len(volt),'#')
print('Antall manglende målinger',count,'#')
print('Prosentvis manglende målinger',round(100*count/len(volt),1),'%')
print('Gjennomsnittlig runtime er',round(np.mean(runTime),1),'s')
```





---

```
print('Gjennomsnittlig intervall er',round((np.mean(interval)/60),1),'min')
print('Sum runtime',round(sumRunTime/3600,1),'timer')
```



## Vedlegg G Programmer for bruk av sensorer

Programmene i dette vedlegget er testprogrammer, programmer for kalibrering og bruk av sensorer for måling av væskeparametre.

### G.1 Måling av temperatur med DS18B20

```
// Temperatur-DS18B20
// Programmet måler og beregner temperaturen målt med DS18B20
// Supply-spenning 3,3 V
// Nils Kr. Rossing 14.08.23

#include <MKRNB.h>
#include <DS18B20.h>
#include <OneWire.h>

DS18B20 ds(0); //Sensoren er koblet til pinne D0
uint8_t address[] = {40, 250, 31, 218, 4, 0, 0, 52};
uint8_t selected;

void setup() {
  Serial.begin(9600);
  selected = ds.select(address);
}

void loop()
{
  Serial.print("Temperature is: ");
  Serial.print(ds.getTempC());
  Serial.println(" C");

  delay(2000);
}
```

### G.2 Kalibrering og måling av sensor for oppløst oksygen SEN0237-A

#### G.2.1 Avlesning av spenning og temperatur

```
// Avlesning av spenning og temperatur for kalibrering av oppløstoksygen
```



```
// Nils Kr. Rossing 16.12.22

#include <Arduino.h>
#include <DS18B20.h>
#include <OneWire.h>

DS18B20 ds(0); //Sensoren er koblet til pinne D0
uint8_t address[] = {40, 250, 31, 218, 4, 0, 0, 52};
uint8_t selected;

#define VREF    3300//VREF(mv)
#define ADC_RES 4096//ADC Resolution

void setup()
{
    Serial.begin(115200);
    selected = ds.select(address);
    analogReadResolution(12); // Sett oppløsningen til AD-konverteren
    12 bit
}

void loop()
{
    float Spenning = 0;
    // Les av og skriv ut spenning
    /*
    for (int i=0; i<100; i++)
    {
        Spenning = Spenning + analogRead(A0);
    }
    Spenning = Spenning/100;
    */
    Spenning = analogRead(A0);

    Serial.print("Spenning: ");
    Serial.print(float(Spenning*VREF/ADC_RES));
    Serial.println("V");

    // Les av og skriv ut temperatur
```



```
float Temperature = ds.getTempC();
Serial.print("Temperatur: ");
Serial.print(Temperature);
Serial.println(" C");

Serial.println();
delay(1000);
}
```

## G.2.2 Måling av oksygeninnhold i vann

```
// Oksygen_Maaling inkluderer kalibrering og måling av temperatur
// Programmet bygger på programmet fra DFRobot:
// https://wiki.dfrobot.com/
Gravity__Analog_Dissolved_Oxygen_Sensor_SKU_SEN0237#More
// men er skrevet om for å passe bedre for MKR NB 1500
// Nilos Kr. Rossing 16.12.22

#include <Arduino.h>
#include <DS18B20.h>
#include <OneWire.h>

DS18B20 ds(0); //Sensoren er koblet til pinne D0
uint8_t address[] = {40, 250, 31, 218, 4, 0, 0, 52};
uint8_t selected;

#define DO_PIN A0

#define VREF 3300 //VREF (mv)
#define ADC_RES 4096 //ADC Resolution

//Single-point calibration Mode=0
//Two-point calibration Mode=1
#define TWO_POINT_CALIBRATION 0

//Single point calibration needs to be filled CAL1_V and CAL1_T
#define CAL1_V 2430 //mv
#define CAL1_T 33 //?
```



```
//Two-point calibration needs to be filled CAL2_V and CAL2_T
//CAL1 High temperature point, CAL2 Low temperature point
#define CAL2_V 1080 //mv
#define CAL2_T 12    ///  
  
const uint16_t DO_Table[41] = {  
    14460, 14220, 13820, 13440, 13090, 12740, 12420, 12110, 11810, 11530,  
    11260, 11010, 10770, 10530, 10300, 10080, 9860, 9660, 9460, 9270,  
    9080, 8900, 8730, 8570, 8410, 8250, 8110, 7960, 7820, 7690,  
    7560, 7430, 7300, 7180, 7070, 6950, 6840, 6730, 6630, 6530, 6410};  
  
uint8_t Temperaturet;  
uint16_t ADC_Raw;  
uint16_t ADC_Voltage;  
uint16_t DO;  
  
int16_t readDO(uint32_t voltage_mv, uint8_t temperature_c)  
{  
#if TWO_POINT_CALIBRATION == 0  
    uint16_t V_saturation = (uint32_t)CAL1_V + (uint32_t)35 * temperature_c  
    - (uint32_t)CAL1_T * 35;  
    return (voltage_mv * DO_Table[temperature_c] / V_saturation);  
#else  
    uint16_t V_saturation = (int16_t)((int8_t)temperature_c - CAL2_T) *  
    ((uint16_t)CAL1_V - CAL2_V) / ((uint8_t)CAL1_T - CAL2_T) + CAL2_V;  
    return (voltage_mv * DO_Table[temperature_c] / V_saturation);  
#endif  
}  
  
void setup()  
{  
    Serial.begin(115200);  
    analogReadResolution(12); // Sett oppløsningen til AD-konverteren 12  
bit  
    selected = ds.select(address);  
  
    Serial.println("Oksygen_Maaling");
```



```
}

void loop()
{
  Temperatur = (uint8_t) ds.getTempC();
  ADC_Raw = analogRead(DO_PIN);
  ADC_Voltage = uint32_t(VREF) * ADC_Raw / ADC_RES;

  Serial.print("Temperatur:\t" + String(Temperatur) + "\t");
  Serial.print("ADC RAW:\t" + String(ADC_Raw) + "\t");
  Serial.print("ADC Voltage:\t" + String(ADC_Voltage) + "\t");
  Serial.println("DO:\t" + String(readDO(ADC_Voltage, Temperatur)) +
"\t");

  delay(1000);
}
```

### **G.3 Kalibrering og måling av sensor for måling av salinitet DFR0300-H**

```
// Salinitetsmåler_DFR0300_H
// Programmet måler og beregner Salinitets verdien til en væske på bak-
grunn av
// kalibreringsverdier for ferskvann 0, 17,5 og 35 promille
// Supply-spenning 3,3 V
// Nils Kr. Rossing 15.12.22

#include <DS18B20.h>
#include <OneWire.h>

DS18B20 ds(0); //Sensoren er koblet til pinne D0
uint8_t address[] = {40, 250, 31, 218, 4, 0, 0, 52};
uint8_t selected;

// Deklarasjon av variabler
float Salinitet = 0; // Målt og beregnet salinitets-verdi
float korreksjon = 0; // Korrigeringsverdi mht temperatur
float korrSalinitet = 0; // Korrigert salinitet
float UD_35 = 0.82; // Målt digital spenningsverdi ved 35 promille
```



```
float UD_175 = 0.49; // Målt digital spenningsverdi ved 17,5
promille

void setup() {
  Serial.begin(115200); //Baud rate: 115 200
  selected = ds.select(address);
  analogReadResolution(12); // Sett oppløsningen til AD-konverteren 12
bit
}

void loop() {
  float Temperature = 0; // Målt temperatur ved hjelp av DS18B20
  float korrSalinitet = 0; // Korreksjonsverdi av salinitet mht
  float korreksjon = 0; // Korreksjonsverdi mht temperaturendringer
  long sensorValue = 0; // Les av analog inngang fra pinne A0:
  float UD_mid = 0; // Middelvei av målt spenning
  float UD_mid_volt = 0; // Middelvei av målt spenning

  for(int i=0; i<100; i++)
  {
    sensorValue = sensorValue + analogRead(A0);
  }
  UD_mid = float(sensorValue/100); // Finner middelveidien

  UD_mid_volt = UD_mid * (3.32 / 4096.0); //
  Temperature = ds.getTempC(); // Måler temperaturen

  // Konverterer den analoge avlesningen (som går fra 0-4095) til en spen-
ning 0 - 3,0V:
  Serial.print(millis()/1000,1);
  Serial.print(";");
  //Serial.print("Midlere spenning: "); // Skriv ut den avleste verdien
for kalibrering:
  Serial.print(UD_mid_volt);

  //Serial.print("Midlere temperatur: "); // Skriv ut den avleste verdien
for kalibrering:
  Serial.print(";");
  Serial.print(Temperature,1);
```



```
// Beregner Salinitet
Salinitet = 9.7532 * UD_mid_volt * UD_mid_volt + 20.955 * UD_mid_volt
- 0.3339; // Polinomtilpasset av 2 grad R^2 0,9998
korreksjon = 0.8566*(Temperature) + 17.557 - 34.3;
korrSalinitet=Salinitet-korreksjon;

//Serial.print("Korrigert salinitet: "); // Skriv ut den avleste
verdien:
Serial.print("");
Serial.println(korrSalinitet,2);

delay(20000);
//int minutter = 1;
//for (int j=0; j<minutter; j++) delay(60000);
}
```

## **G4 Kalibrering og måling Turbiditet SEN-0189**

### **G4.1 Eksempelprogram for måling av spenningen (Turbidimeter-SEN0189.ino)**

```
void setup() {
  Serial.begin(9600); //Baud rate: 9600
  analogReadResolution(12); // Sett oppløsningen til AD-konverteren 12
  bit
}

void loop() {
  int sensorValue = 0;
  float voltage = 0;

  for (int i=0; i<100; i++)
  {
    sensorValue = analogRead(A0); // Les av analog inngang fra pinne A0:
    voltage = voltage + 2 * sensorValue * (3.3 / 4096.0);
  }

  voltage = voltage/100;
```





```
// Konverterer den analoge avlesningen (som går fra 0-4095) til en spenning (0 - 4.1 +/- 0,3 V ved 0 NTU (Rent vann)):  
Serial.println(voltage); // Skriv ut den avleste verdien:  
delay(500);  
}
```

## G.4.2 Eksempelprogram for deteksjon av passering av et terskelnivå

```
int ledPin = LED_BUILTIN; // Connect an LED on pin 13, or use the onboard one  
int sensor_in = 0; // Connect turbidity sensor to Digital Pin 2  
  
void setup(){  
  pinMode(ledPin, OUTPUT); // Set ledPin to output mode  
  pinMode(sensor_in, INPUT); //Set the turbidity sensor pin to input mode  
}  
  
void loop(){  
  if(digitalRead(sensor_in)==LOW){ //read sensor signal  
    digitalWrite(ledPin, HIGH); // if sensor is LOW, then turn on  
  }  
  else{  
    digitalWrite(ledPin, LOW); // if sensor is HIGH, then turn off the led  
  }  
}  
  
// Deklarasjon av variabler  
float NTU = 0; // Målt og beregnet NTU-verdi  
float NTU_200 = 0; // Kalibrert NTU-verdi 200 NTU.  
float NTU_400 = 0; // Kalibrert NTU-verdi 400 NTU.  
float UD_200 = 0; // Målt digital spenningsverdi ved NTU-verdi 200  
float UD_400 = 0; // Målt digital spenningsverdi ved NTU-verdi 400  
  
void setup() {  
  Serial.begin(115200); //Baud rate: 115 200  
  analogReadResolution(12); // Sett oppløsningen til AD-konverteren 12 bit
```



```
}
```

### G.4.3 Eksempelprogram for kalibrering og måling (Turbidimeter\_SEN0189\_Kal-1)

Programmet kan brukes til å avlese verdier for kalibrering slik at det kan gjøre en lineær beregning av NTU i henhold til de standarder som er brukt.

```
// Turbidimeter_SEN0189_Kal-1
// Programmet måler og beregner NTU verdien til en væske på bakgrunn av
// kalibreringsverdier for NTU-verdi 200 og NTU-verdi 400
// Nils Kr. Rossing 03.12.22
void loop() {
  long sensorValue = 0; // Les av analog inngang fra pinne A0:
  float UD_mid      = 0; // Middelerdi av målt spenning
  float UD_mid_volt = 0; // Middelerdi av målt spenning

  for(int i=0; i<100; i++)
  {
    sensorValue = sensorValue + analogRead(A0);
  }

  UD_mid = 2.0 * float(sensorValue/100); // Finner middelerdien
  UD_mid_volt = UD_mid * (3.3 / 4096.0); //
  // Konverterer den analoge avlesningen (som går fra 0-4095) til en spenning (0 - 4.1 +/- 0,3 V ved 0 NTU (Rent vann)):

  Serial.print("Maalt midlere spenning: "); // Skriv ut den avleste verdien for kalibrering:
  Serial.println(NTU);

  // Beregner NTU-verdien
  NTU = (NTU_400 - NTU_200)/(UD_200 - UD_400)*(UD_mid_volt-4.1);
  Serial.print("Maalt NTU: "); // Skriv ut den avleste verdien:
  Serial.println(NTU);
  delay(500);
}
```

### G.5 Kalibrering og måling med pH-sensoren SEN-0161

Programmet brukes for kalibrering og for måling av pH-verdien.



```
/*
 # This sample code is used to test the pH meter V1.0.
 # Editor : YouYou
 # Modifier : Nils Kr. Rossing 01.12.22
 # Ver      : 1.0
 # Product: analog pH meter
 # SKU      : SEN0161
*/
#define SensorPin A0          //pH meter Analog output to Arduino Analog
Input 0
#define Offset -3.50          //deviation compensate
#define LED LED_BUILTIN
#define samplingInterval 20
#define printInterval 800
#define ArrayLenth 40        //times of collection
int pHArray[ArrayLenth];     //Store the average value of the sensor
feedback
int pHArrayIndex=0;
void setup(void)
{
  pinMode(LED,OUTPUT);
  Serial.begin(9600);
  Serial.println("pH meter experiment!");    //Test the serial monitor
}
void loop(void)
{
  static unsigned long samplingTime = millis();
  static unsigned long printTime = millis();
  static float pHValue,voltage;
  if(millis()-samplingTime > samplingInterval)
  {
    pHArray[pHArrayIndex++]=analogRead(SensorPin);
    if(pHArrayIndex==ArrayLenth)pHArrayIndex=0;
    voltage = avergearray(pHArray, ArrayLenth)*5.0/1024;
    pHValue = 3.5*voltage+Offset;
    samplingTime=millis();
  }
  if(millis() - printTime > printInterval) //Every 800 milliseconds,
  print a numerical, convert the state of the LED indicator
```



```
{
  Serial.print("Voltage:");
  Serial.print(voltage,2);
  Serial.print("    pH value: ");
  Serial.println(pHValue,2);
  digitalWrite(LED,digitalRead(LED)^1);
  printTime=millis();
}
}
double avergearray(int* arr, int number){
  int i;
  int max,min;
  double avg;
  long amount=0;
  if(number<=0){
    Serial.println("Error number for the array to avraging!/n");
    return 0;
  }
  if(number<5){ //less than 5, calculated directly statistics
    for(i=0;i<number;i++){
      amount+=arr[i];
    }
    avg = amount/number;
    return avg;
  }else{
    if(arr[0]<arr[1]){
      min = arr[0];max=arr[1];
    }
    else{
      min=arr[1];max=arr[0];
    }
    for(i=2;i<number;i++){
      if(arr[i]<min){
        amount+=min; //arr<min
        min=arr[i];
      }else {
        if(arr[i]>max){
          amount+=max; //arr>max
        }
      }
    }
  }
}
```



```
        max=arr[i];
    }else{
        amount+=arr[i]; //min<=arr<=max
    }
} //if
} //for
avg = (double)amount/(number-2);
} //if
return avg;
}
```

## G.6 Kalibrering og måling med ORP-sensoren SEN-0165

```
/*
# This sample codes is for testing the ORP meter V1.0.
# Editor : YouYou
# Date   : 2013.11.26
# Product: ORP meter
# SKU    : SEN0165
*/
#define VOLTAGE 5.00 //system voltage
#define OFFSET -1122 //zero drift voltage
#define LED LED_BUILTIN //operating instructions

double orpValue;

#define ArrayLenth 40 //times of collection
#define orpPin 0 //orp meter output,connect to Arduino controller
ADC pin

int orpArray[ArrayLenth];
int orpArrayIndex=0;

double avergearray(int* arr, int number){
    int i;
    int max,min;
    double avg;
    long amount=0;
    if(number<=0){
        printf("Error number for the array to avraging!\n");
    }
}
```



```
    return 0;
}
if(number<5){ //less than 5, calculated directly statistics
    for(i=0;i<number;i++){
        amount+=arr[i];
    }
    avg = amount/number;
    return avg;
}else{
    if(arr[0]<arr[1]){
        min = arr[0];max=arr[1];
    }
    else{
        min=arr[1];max=arr[0];
    }
    for(i=2;i<number;i++){
        if(arr[i]<min){
            amount+=min; //arr<min
            min=arr[i];
        }else {
            if(arr[i]>max){
                amount+=max; //arr>max
                max=arr[i];
            }else{
                amount+=arr[i]; //min<=arr<=max
            }
        }
    }
}
return avg;
}
```

```
void setup(void) {
    Serial.begin(9600);
    pinMode(LED,OUTPUT);
    analogReadResolution(10); // Sett oppløsningen til AD-konverteren 12
bit
```



```
}

void loop(void) {
    static unsigned long orpTimer=millis(); //analog sampling interval
    static unsigned long printTime=millis();
    if(millis() >= orpTimer)
    {
        orpTimer=millis()+20;
        orpArray[orpArrayIndex++]=analogRead(orpPin); //read an analog
value every 20ms
        if (orpArrayIndex==ArrayLenth) {
            orpArrayIndex=0;
        }
        orpValue=((30*(double)VOLTAGE*1000)-(75*avergearray(orpArray,
ArrayLenth)*VOLTAGE*1000/1024))/75-OFFSET;

        //convert the analog value to orp according the circuit
    }
    if(millis() >= printTime) //Every 800 milliseconds, print a numerical,
convert the state of the LED indicator
    {
        printTime=millis()+800;
        Serial.print("ORP: ");
        Serial.print((int)orpValue);
        Serial.println("mV");
        digitalWrite(LED,1-digitalRead(LED));
    }
}

LED
    flash(1);
    #endif
}
else
{
    #ifdef DEBUG
        Serial.println("Mislykket tilkobling til GPRS");
    #endif
}
```



```
    #ifdef LED
        flash(5);
    #endif

    break;
}
}
}

void sdPrint(){

    myFile = SD.open("DataFile.txt", FILE_WRITE);
    delay(1000);

    if (myFile)
    {
        myFile.println(buffer);
        myFile.close();
    }
}

// Funksjonen kobler opp til server og skriver databuffer til serveren
void connectToServer()
{
    if (client.connect(server, port))
    {
        client.print("GET "); // Gjør et HTTP request:
        client.print(buffer);
        client.println(" HTTP/1.1");
        client.print("Host: ");
        client.println(server);
        client.println("Connection: close");
        client.println();

        #ifdef DEBUG
            Serial.print("Vellykket overføring til server av datasett nr.: ");

```





```
        Serial.println(num);
    #endif

    #ifdef LED
        flash(2);
    #endif
}
else
{
    #ifdef DEBUG
        Serial.print("Mislykket overføring til server av datasett nr.: ");
        Serial.println(num);
    #endif

    #ifdef LED
        flash(10);
    #endif
}
}

// Funksjonen blinker et angitt antall ganger som en hjelp til debugging
void flash(int number)
{
    delay(1000);
    for(int i=0; i<number; i++)
    {
        digitalWrite(LED_BUILTIN, HIGH);
        delay(100);
        digitalWrite(LED_BUILTIN, LOW);
        delay(100);
    }
    delay(1000);
}

// Funksjonen legger kretsen i dvale
void GoToSleep() // Legg mikrokontrolleren og GPS'en i dvale
```



```
{
  #ifdef DEBUG
    Serial.println("Går i dvale");
  #endif

  #ifdef LED
    flash(3);
  #endif

  GPS.standby();
  LowPower.deepSleep(SleepTime);
}

// Returfunksjon etter endt dvale
void dummy()
{
  GPS.wakeup();
  NVIC_SystemReset(); // Resett programmet
}

void myshutdown()
{
  #ifdef DEBUG
    Serial.println("Resetter MKR");
  #endif
}
```

## G.7 Kalibrering og måling av trykk og dybde med SEN-0257

```
// Program for kalibrering og måling av trykk og vanndybde
// Programmet tar utgangspunkt i et program skrevet for DFRobot og modifisert
// av:
// Nils Kr. Rossing 27.11.23

const float OffSet = 0.774; // See Calibration Note!
const float waterDensity = 1.000; // kg/L Ferskvann 1,000 kg/dm3 Saltvann 1,020
- 1,029 kg/dm3
const float g = 9.81; // Gravitation constant 9,81 m/s2
const int averageNo = 500; // Antall målinger for midling
```



```
float V, P, D; // Voltage, Pressure and Dept
float Sens = 400; // Antall kPa pr. volt

void setup() {
  Serial.begin(115200);
  Serial.println("** Water Pressure Sensor Data **/");
  analogReadResolution(12); // 12Bits
}
void loop() {
  //Connect sensor's output (SIGNAL) to Analog 0
  V = 0;
  for(int i=0; i<averageNo; i++ ) V = V + analogRead(0) * 5.00 / 4096;

  V = V/averageNo;
  P = (V - OffSet) * Sens;
  D = P/(waterDensity*g); // P in kPa and waterDensity in kg/dm3 and g in m/s2

  Serial.print("Voltage:");
  Serial.print(V, 3);
  Serial.println("V");
  Serial.print("Pressure:");
  Serial.print(P, 1);
  Serial.println("kPa");
  Serial.print("Water depth:");
  Serial.print(D, 2);
  Serial.println("m");
  Serial.println();
  delay(1000);
}
```



## Vedlegg H Testprogrammer for MKR NB 1500

### H.1 Testprogram for å legge i dyp søvn og vekkes fra en port<sup>134</sup>

```
/*
  ExternalWakeup
  This sketch demonstrates the usage of External Interrupts (on pins) to wakeup a
  chip in sleep mode.

  Sleep modes allow a significant drop in the power usage of a board while it does
  nothing waiting for an event to happen. Battery powered application can take advantage
  of these modes to enhance battery life significantly.

  In this sketch, shorting pin 8 to a GND will wake up the board.

  Please note that, if the processor is sleeping, a new sketch can't be uploaded.
  To overcome this, manually reset the board (usually with a single or double tap
  to the RESET button)

  This example code is in the public domain.
*/

#include "ArduinoLowPower.h"

// Blink sequence number
// Declare it volatile since it's incremented inside an interrupt
volatile int repetitions = 1;

// Pin used to trigger a wakeup
const int pin = 8;

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  // Set pin 8 as INPUT_PULLUP to avoid spurious wakeup
  pinMode(pin, INPUT_PULLUP);
  // Attach a wakeup interrupt on pin 8, calling repetitionsIncrease when the
  device is woken up
  LowPower.attachInterruptWakeup(pin, repetitionsIncrease, CHANGE);
}

void loop() {
  for (int i = 0; i < repetitions; i++) {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(500);
```

---

<sup>134</sup><https://github.com/arduino-libraries/ArduinoLowPower/blob/master/examples/ExternalWakeup/ExternalWakeup.ino>



```
digitalWrite(LED_BUILTIN, LOW);
delay(500);
}
// Triggers an infinite sleep (the device will be woken up only by the registered
wakeUp sources)
// The power consumption of the chip will drop consistently
LowPower.sleep();
}

void repetitionsIncrease() {
  // This function will be called once on device wakeup
  // You can do some little operations here (like changing variables which will
be used in the loop)
  // Remember to avoid calling delay() and long running functions since this
functions executes in interrupt context
  repetitions ++;
}
```

## H.2 Testprogram for å legge i dyp søvn og vekkes av intern “time out”<sup>135</sup>

```
/*
  TimedWakeup
  This sketch demonstrates the usage of Internal Interrupts to wakeup a chip in
sleep mode.
  Sleep modes allow a significant drop in the power usage of a board while it does
nothing waiting for an event to happen. Battery powered application can take advan-
tage of these modes to enhance battery life significantly.
  In this sketch, the internal RTC will wake up the processor every 2 seconds.
  Please note that, if the processor is sleeping, a new sketch can't be uploaded.
To overcome this, manually reset the board (usually with a single or double tap
to the RESET button)
  This example code is in the public domain.
*/

#include "ArduinoLowPower.h"

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  // Uncomment this function if you wish to attach function dummy when RTC wakes
up the chip
  // LowPower.attachInterruptWakeup(RTC_ALARM_WAKEUP, dummy, CHANGE);
}
```

---

<sup>135</sup><https://github.com/arduino-libraries/ArduinoLowPower/blob/master/examples/TimedWakeup/TimedWakeup.ino>



```
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(500);
  digitalWrite(LED_BUILTIN, LOW);
  delay(500);
  // Triggers a 2000 ms sleep (the device will be woken up only by the registered
  wakeup sources and by internal RTC)
  // The power consumption of the chip will drop consistently
  LowPower.sleep(2000);
}

void dummy() {
  // This function will be called once on device wakeup
  // You can do some little operations here (like changing variables which will
  be used in the loop)
  // Remember to avoid calling delay() and long running functions since this
  functions executes in interrupt context
}
```



# Vedlegg I Testprogrammer for shield-kort

Her har vi samlet testprogrammet for uttesting av ulike shield-kort.

## I.1 Testprogram for MKR ENV<sup>136</sup>

```
#include <Arduino_MKRENV.h>

void setup() {
  Serial.begin(9600);
  while (!Serial);

  if (!ENV.begin()) {
    Serial.println("Failed to initialize MKR ENV shield!");
    while (1);
  }
  Serial.println("Testprogram-MKR-ENV-1");
}

void loop() {
  // read all the sensor values
  float temperature = ENV.readTemperature();
  float humidity    = ENV.readHumidity();
  float pressure    = ENV.readPressure();
  float illuminance = ENV.readIlluminance();
  //float uva       = ENV.readUVA();
  //float uvb       = ENV.readUVB();
  //float uvIndex   = ENV.readUVIndex();

  // print each of the sensor values
  Serial.print("Temperature = ");
  Serial.print(temperature);
  Serial.println(" °C");

  Serial.print("Humidity    = ");
  Serial.print(humidity);
  Serial.println(" %");

  Serial.print("Pressure    = ");
  Serial.print(pressure);
```

---

136.<https://docs.arduino.cc/tutorials/mkr-env-shield/mkr-env-shield-basic>



```
Serial.println(" kPa");

Serial.print("Illuminance = ");
Serial.print(illuminance);
Serial.println(" lx");
/*
Serial.print("UVA          = ");
Serial.println(uva);

Serial.print("UVB          = ");
Serial.println(uvb);

Serial.print("UV Index    = ");
Serial.println(uvIndex);
*/

// print an empty line
Serial.println();

// wait 1 second to print again
delay(1000);
}
```

## I.2 Testprogram for MKR ENV skiving til SD-kort<sup>137</sup>

```
/*
  02 Sensors to SD v0001
  Read information from the all of the sensors
  on the MKR ENV Shield and store it on a
  CSV file inside an SD card. Connect the SD
  card to the slot on the ENV Shield
  (c) 2019 D. Cuartielles for Arduino
  This code is Free Software licensed under GPLv3
*/

#include <Arduino_MKRENV.h>
#include <SPI.h>
#include <SD.h>

// chip select for SD card
```

---

137.<https://docs.arduino.cc/tutorials/mkr-env-shield/mkr-env-shield-basic>





```
const int SD_CS_PIN = 4;

// variables
float temperature = 0;
float humidity = 0;
float pressure = 0;
/*
float UVA = 0;
float UVB = 0;
float UVIndex = 0;
*/

// file object
File dataFile;

void setup() {
  Serial.begin(9600);

  // init the ENV Shield
  if (!ENV.begin()) {
    Serial.println("Failed to initialize MKR ENV shield!");
    while (1);
  }

  // init SPI
  SPI.begin();
  delay(100);

  // init SD card
  if (!SD.begin(SD_CS_PIN)) {
    Serial.println("Failed to initialize SD card!");
    while (1);
  }

  // init the logfile
  dataFile = SD.open("log-0001.csv", FILE_WRITE);
  delay(1000);

  // init the CSV file with headers
  //dataFile.println("temperature,humidity,pressure,UVA,UVB,UVindex");
  dataFile.println("temperature,humidity,pressure");
}
```



```
// close the file
dataFile.close();
delay(100);
Serial.println("Testprogram-MKR-ENV-SD-1");
}

void loop() {
  // init the logfile
  dataFile = SD.open("log-0001.csv", FILE_WRITE);
  delay(1000);

  // read the sensors values
  temperature = ENV.readTemperature();
  humidity = ENV.readHumidity();
  pressure = ENV.readPressure();
  /*
  UVA = ENV.readUVA();
  UVB = ENV.readUVB();
  UVIndex = ENV.readUVIndex();
  */

  // print each of the sensor values
  dataFile.print(temperature);
  dataFile.print(",");
  dataFile.print(humidity);
  dataFile.print(",");
  dataFile.println(pressure);
  /*
  dataFile.print(",");
  dataFile.print(UVA);
  dataFile.print(",");
  dataFile.print(UVB);
  dataFile.print(",");
  dataFile.println(UVIndex);
  */

  // close the file
  dataFile.close();

  // wait 1 second to print again
}
```



```
    delay(1000);  
}
```

### I.3 Testprogram for MKR GPS<sup>138</sup>

```
/*  
 * Testprogram for MKR GPS  
 * Programmet er hentet fra eksempelsamlingen fra MKR GPS  
 * men endret med hensyn til lesing av data fra kretsen  
 * Endret av Nils Kr. Rossing 21.03.22  
 */  
  
#include <Arduino_MKRGPS.h>  
  
void setup() {  
    // initialize serial communications and wait for port to open:  
    Serial.begin(9600);  
    while (!Serial) {  
        ; // wait for serial port to connect. Needed for native USB port only  
    }  
  
    // If you are using the MKR GPS as shield, change the next line to pass  
    // the GPS_MODE_SHIELD parameter to the GPS.begin(...)  
    if (!GPS.begin()) {  
        Serial.println("Failed to initialize GPS!");  
        while (1);  
    }  
    Serial.println("Testårogram-MKR-GPS-1");  
}  
  
void loop() {  
    // check if there is new GPS data available  
    while(!GPS.available()) {}  
    // Les GPS verdier  
    float latitude = GPS.latitude();  
    float longitude = GPS.longitude();  
    float altitude = GPS.altitude();  
    float speed = GPS.speed();  
    int satellites = GPS.satellites();
```

---

138.<https://docs.arduino.cc/tutorials/mkr-gps-shield/mkr-gps-basic>



```
// Skriv ut GPS verdier
Serial.print("Location: ");
Serial.print(latitude, 7);
Serial.print(", ");
Serial.println(longitude, 7);

Serial.print("Altitude: ");
Serial.print(altitude);
Serial.println("m");

Serial.print("Ground speed: ");
Serial.print(speed);
Serial.println(" km/h");

Serial.print("Number of satellites: ");
Serial.println(satellites);

Serial.println();
}
```

#### I.4 Testprogram for MKR ENV og GPS sammen

Programmet: Testprogram-ENV-GPS-1 kombinerer de to testprogrammene for MKR ENV- og GPS-kortene

```
#include <Arduino_MKRENV.h>
#include <Arduino_MKRGPS.h>

void setup() {
  Serial.begin(9600);
  while (!Serial);

  if (!ENV.begin()) {
    Serial.println("Failed to initialize MKR ENV shield!");
    while (1);
  }

  if (!GPS.begin()) {
    Serial.println("Failed to initialize GPS!");
    while (1);
  }
}
```



```
Serial.println("Testprogram-MKR-ENV-1");

}

void loop() {
  // read all the sensor values
  float temperature = ENV.readTemperature();
  float humidity    = ENV.readHumidity();
  float pressure    = ENV.readPressure();
  float illuminance = ENV.readIlluminance();

  // print each of the sensor values
  Serial.print("Temperature = ");
  Serial.print(temperature);
  Serial.println(" °C");

  Serial.print("Humidity    = ");
  Serial.print(humidity);
  Serial.println(" %");

  Serial.print("Pressure    = ");
  Serial.print(pressure);
  Serial.println(" kPa");

  Serial.print("Illuminance = ");
  Serial.print(illuminance);
  Serial.println(" lx");

  // print an empty line
  Serial.println();

  delay(500);

  // check if there is new GPS data available
  while(!GPS.available()) {}

  // read GPS values
  float latitude = GPS.latitude();
  float longitude = GPS.longitude();
  float altitude = GPS.altitude();
  float speed    = GPS.speed();
```



```
int satellites = GPS.satellites();

// print GPS values
Serial.print("Location: ");
Serial.print(latitude, 7);
Serial.print(", ");
Serial.println(longitude, 7);

Serial.print("Altitude: ");
Serial.print(altitude);
Serial.println("m");

Serial.print("Ground speed: ");
Serial.print(speed);
Serial.println(" km/h");

Serial.print("Number of satellites: ");
Serial.println(satellites);

Serial.println();

// wait 0,5 second to print again
delay(500);
}
```

## I.5 Testprogram for kalibrering av temperatursensoren – NTC

```
/*
 * Programmet er et hjelpemiddel for å kalibrere temperatursensoren med
 * NTC-motstanden NTCLE400E3103H. Kalibreringen skjer i to runder:
 * 1. Først leser av den digitale verdien for to temperaturer i ytterkanten av det
aktuelle temperaturområdet.
 * 2. Dernest legges målinger og temperaturer inn i programmet
 * 3. så beregnes den kalibrerte temperaturen
 * Nils Kr. Rossing 02.07.22
 */

#include <stdio.h>
#include <MKRNB.h>

// ----- Legg inn verdier for kalibrering av temperatur -----//

int D_T_Lav = 338; // Digital verdi for spenningen på A0, registrert lav temperatur)
```



```
int D_T_Hoy = 710; // Digital verdi for spenningen på A0, registrert ved høy temperatur)
int D_T_Var = 0; // Digital variabel temperatur
float T_Lav = 6; // Lav kalibreringstemperatur i grader C avlest på kalibrert instrument
float T_Hoy = 16; // Høy kalibreringstemperatur i grader C avlest på kalibrert instrument

// ----- Beregnete kalibrete verdier -----//

float T_Beregnet = 0; // T - Beregnet temperatur

// Bergenete spenningsverdier:

float U_T_Var = 0; // Avlest og beregnet spenningsverdi for temperatur

int i = 0;

void setup()
{
  Serial.begin(115200);
  while (!Serial);

  analogReadResolution(12); // Sett AD-konverteren til 12 bits oppløsning
}

void loop()
{
  i = i+1;

  //--- Leser inn digital spenningen for temperaturen og beregner kalibrert temper-
  ratur ----//

  D_T_Var = analogRead(A0);
  U_T_Var = 3.3 * D_T_Var/4096;

  T_Beregnet = ((T_Hoy - T_Lav)/(D_T_Hoy - D_T_Lav))*
              (D_T_Var - D_T_Lav) + T_Lav;

  //-----Skriv ut for test og kalibrering -----//

  Serial.print("Nr.: ");
  Serial.println(i);
  Serial.print("Digital temp.: ");
```



```
    Serial.print(D_T_Var);
    Serial.print(", Beregnet temp.: ");
    Serial.print(T_Beregnet, 2);
    Serial.println(" C");

    delay(1000);
}
```

## I.6 Testprogram for testing av temperatursensoren DS18B20

```
/*
 * Programmet setter øvre og nedre alarmtemperatur og henter temperaturen
 * og skriver ut denne. Nils Kr. Rossing 04.07.22
 */

#include <MKRNB.h>
#include <DS18B20.h>
#include <OneWire.h>

DS18B20 ds(0); //Sensoren er koblet til pinne D0
uint8_t address[] = {40, 250, 31, 218, 4, 0, 0, 52};
uint8_t selected;

void setup()
{
    Serial.begin(115200);
    selected = ds.select(address);

    Serial.println("Testprogram-DS18B20-Temp-1");
}

void loop()
{
    Serial.print("Temperature is: ");
    Serial.print(ds.getTempC());
    Serial.println(" C");

    delay(2000);
}
```





---

}



## Vedlegg J Eksempelprogram for EVU-kurset

Vedlegget inneholder det gjennomgående eksempelprogrammet som vi skal bruke gjennom hele kurset.

### J.1 Gjennomgående eksempelprogram (EVU-kurs23-Eksempelprogram.ino)

```
// Lynkurs-3-mWD-SD
// Kode for uttesting av program for måling og overføring av måledata til server
// Testoppstillingen inneholder følgende:
//
// MKR NB 1500 m/GSM-antenne
// MKR GPS GPS-mottaker
// DS18B20 Temperatursensor for måling i vann
// SD-kort terminal for lagring av data
//
// Nils Kr. Rossing og Thor Inge Hansen 03.02.23

// Inkludering av biblioteker:
#include <DS18B20.h> // Bibliotek for avlesning av temperatursensoren
#include <OneWire.h> // Bibliotek som etablerer entrådsbuss til temp.
sensoren
#include <Arduino_MKRGPS.h> // Bibliotek som leser av GPS
#include <MKRNB.h> // Bibliotek som overførerdata til server via GPRS
#include <Arduino_MKRENV.h> // Bibliotek for lesing av miljøkortet ENV om det
brukes
#include "ArduinoLowPower.h" // Bibliotek som legger MKR i dvale
#include <WDTZero.h> // Bibliotek som håndterer vakthunden
#include <SD.h> // Bibliotek for håndtering av SD-kort
#include <SPI.h> // Bibilotek for håndtering av serie-bus

// Deklarasjon av instanser:
WDTZero MyWatchDoggy; // Deklarerer instansen til vakthunden MyWatchDoggy av
typen WDTZero
NBClient client; // Deklarerer instansen som opererer mot serveren
GPRS gprs; // Deklarerer instansen som opprettet kontakt med GPRS-nettet
NB nbAccess; // Deklarerer instansen som som gir tilgang til GPRS-nettet
DS18B20 ds(0); // Deklarerer instansen som kommuniserer med temp. sensoren
File myFile; // Deklarerer et objekt for filbehandling, kun ved bruk
ved lagring på SD-kort

// Deklarasjon av konstanter:
```



```
#define DEBUG          // Kommenter bort denne dersom Serial.print ikke skal
inkluderes
//#define LED          // Kommenter bort denne dersom LED indikasjon ikke ønskes

// Deklarasjon av variabler:
// Temperatursensor DS18B20
uint8_t address[] = {40, 250, 31, 218, 4, 0, 0, 52};
uint8_t selected;

// Oppkobling til server
char server[] = "sensor.marin.ntnu.no"; // Navnet på serveren der data legges
char path[] = "/cgi-bin/tof.cgi?";     // Angir cgi-kode for plassering i file
char filename[]="EVU-kurs-Grx.txt";    // Navnet på fil for lagring av data
int port = 80;                         // Port 80 er default for HTTP
boolean connected = false;             // Status oppkobling til GPRS
char buffer[128];                      // Deklarsjon av buffer med tilhørende lengde

volatile unsigned long num = 0;        // Måling nummer
float w_temperature = 0;              // Vanntemperatur
float BatVolt = 0;                   // Variabel for å holde batterispenningen
long Pause = 5000;                   // En ev. pause i programloopen
int SleepTime = 5000;                // Tid for dvale, for lengre tider enn 32 sek,
sett verdien rett inn i argumentet

float GPSlat = 0;                    // GPS breddegrad
float GPSlong = 0;                   // GPS lengdegrad
unsigned long epochTime = 0;         // GPS tidsstempel fra 1.1.1980
int numSat = 0;                      // GPS antall satellitter
int maxNoChecks = 10000;             // Antall runder for lesing av GPS-data
int noChecks = 0;                    // Antall nødvendige sjekk av GPS

// Deklarasjon av porter
int BatVoltPin = A3;                 // Analog port for tilkobling av batterispenning
const int SD_CS_PIN = 4;            // Port for styring av SD-kort terminal,
fabrikkbestemt

void setup() {
  pinMode(LED_BUILTIN, OUTPUT); //Port for innebygget LED, definert somutgang

  analogReadResolution(12);         // Angir oppløsning for AD-konverter

  #ifdef DEBUG
```



```
    Serial.begin(115200);          // Setter opp data hastighet til monitor
#endif

if (!GPS.begin())                // Initialisering av GPS
{
    #ifdef DEBUG
        Serial.println("GPS initialisering mislyktes");
    #endif
}

GPS.begin();                    // Initialisering av GPS
SPI.begin();

if(!SD.begin(SD_CS_PIN))
{
    #ifdef DEBUG
        Serial.println("Initialisering SD terminal mislyktes");
        Serial.println("Har du satt i SD-kort?");
    #endif
}

// Lag en file med navnet DataFile.txt
myFile = SD.open("DataFile.txt", FILE_WRITE);
delay(1000);
myFile.close();
delay(100);

// Initialiserer dvale-funksjon
LowPower.attachInterruptWakeup(RTC_ALARM_WAKEUP, dummy, CHANGE);

// Initialisering av "vakthund"
MyWatchDoggy.attachShutdown(myshutdown); // Starter funksjonen myshutdown() ved
reset
//MyWatchDoggy.setup(WDT_SOFTCYCLE1M);    // Setter intervalltiden for
"vakthunden"

// Opprett tilkobling til 4G - GPRS-tjeneste
connectToGPRS();

delay(1000);
}
```



```
void loop() {
  //readGPSdata1();      // Inkluder GPS
  readWaterTemp();      // Inkluder måling av temp. i vann
  //readBatVolt();      // Inkluder måling av baterispennning om den er implementert

  makeString();         // Bygge opp bufferet for overføring av data
  //sdPrint();          // Skriv data til SD-terminal
  //connectToGPRS();    // Koble til GPRS-nettverket
  connectToServer();    // Koble opp mot server og overfør data om den ikke alt
  er oppkoblet

  printData();
  printDataString();    // Skriv ut bufferet
  //MyWatchDoggy.clear(); // Resetter vakthunden

  num++;                // Målingens nummer fra start
  //GoToSleep();        // Legg microkontrolleren og GPS'en i dvale
  delay(Pause);         // Ev. legg inn en pause i loopen
}

// Funksjonen leser av vanntemperaturen
void readWaterTemp()
{
  w_temperature = ds.getTempC();
}

// Funksjonen leser av batterispenningen
void readBatVolt(){
  BatVolt = analogRead(ADC_BATTERY)*(4.25/4096); //volts
}

// Funksjonen leser av GPS-data med bruk av en for-loop
void readGPSdata1()
{
  noChecks = 0;

  for (int i = 0; i < maxNoChecks; i++)
  {
    // Check if there is new GPS data available
    if (GPS.available())
    {
```



```
// If there is, read GPS values
GPSlat  = GPS.latitude();
GPSlong = GPS.longitude();
epochTime = GPS.getTime();
numSat = GPS.satellites();

break;
}

noChecks++;

#ifdef DEBUG
    if (noChecks == maxNoChecks) Serial.println("Mislykket henting av posisjon
fra GPS");
#endif

}
}

// Funksjonen bygger opp datastrengen for overføring
void makeString()
{
    sprintf(buffer, "/cgi-bin/tof.cgi?s,%u;%u;%0.6f;%0.6f;%i;%0.1f;%0.2f", filename,
num, epochTime, GPSlat, GPSlong, numSat, w_temperature, BatVolt);
}

// Funksjonen skriver ut bufferet til monitoren
void printDataString()
{
#ifdef DEBUG
    Serial.println(buffer);
    Serial.println();
#endif
}

// Funksjonen kobler opp til GPRS-nettverket (4G)
void connectToGPRS()
{
    while (!connected)
    {
        if ((nbAccess.begin("", true, true) == NB_READY) && (gprs.attachGPRS() ==
GPRS_READY))
    }
}
}
```



```
{
  connected = true;

  #ifdef DEBUG
    Serial.println("Vellykket tilkobling til GPRS");
  #endif

  #ifdef LED
    flash(1);
  #endif
}
else
{
  #ifdef DEBUG
    Serial.println("Mislykket tilkobling til GPRS");
  #endif

  #ifdef LED
    flash(5);
  #endif

  break;
}
}

void sdPrint(){

  myFile = SD.open("DataFile.txt", FILE_WRITE);
  delay(1000);

  if (myFile)
  {
    myFile.println(buffer);
    myFile.close();
  }
}

// Funksjonen kobler opp til server og skriver databuffer til serveren
void connectToServer()
{
```



```
if (client.connect(server, port))
{
  client.print("GET "); // Gjør et HTTP request:
  client.print(buffer);
  client.println(" HTTP/1.1");
  client.print("Host: ");
  client.println(server);
  client.println("Connection: close");
  client.println();

  #ifdef DEBUG
    Serial.print("Vellykket overføring til server av datasett nr.: ");
    Serial.println(num);
  #endif

  #ifdef LED
    flash(2);
  #endif
}
else
{
  #ifdef DEBUG
    Serial.print("Mislykket overføring til server av datasett nr.: ");
    Serial.println(num);
  #endif

  #ifdef LED
    flash(10);
  #endif
}
}

// Funksjonen blinker et angitt antall ganger som en hjelp til debugging
void flash(int number)
{
  delay(1000);
  for(int i=0; i<number; i++)
  {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(100);
    digitalWrite(LED_BUILTIN, LOW);
  }
}
```





```
    delay(100);
  }
  delay(1000);
}

// Funksjonen legger kretsen i dvale
void GoToSleep() // Legg mikrokontrolleren og GPS'en i dvale
{
  #ifdef DEBUG
    Serial.println("Går i dvale");
  #endif

  #ifdef LED
    flash(3);
  #endif

  GPS.standby();
  LowPower.deepSleep(SleepTime);
}

// Returfunksjon etter endt dvale
void dummy()
{
  GPS.wakeup();
  NVIC_SystemReset(); // Resett programmet
}

void myshutdown()
{
  #ifdef DEBUG
    Serial.println("Resetter MKR");
  #endif
}

void printData(){
  #ifdef DEBUG
    Serial.print("GPS sjekk : "); Serial.println(noChecks);
    Serial.print("Breddegrad: "); Serial.println(GPSlat,6);
    Serial.print("Lengdegrad: "); Serial.println(GPSlong,6);
    Serial.print("Epoketid  : "); Serial.println(epochTime);
    Serial.print("Antall sat: "); Serial.println(numSat);
  #endif
}
```



---

```
Serial.print("Temp. vann: "); Serial.println(w_temperature);  
Serial.print("Batterisp.: "); Serial.println(BatVolt);  
Serial.print("Måling nr.: "); Serial.println(num);  
Serial.println();  
#endif  
}
```



## Vedlegg K Operativt program med ekstern “vakthund” (Ferdig\_program\_med\_PICAXE\_MILLIS\_TEST\_2)

Dette programmet er beregnet for å brukes dersom man bruker ekstern “vakthund” som holder styr på tidspunkter for måling. Hele programmet ligger setup()-funksjonen og vil derfor kjøre en gang hver gang “vakthunden” slår på spenningen. Dersom programmet skulle “henge seg” vil en intern programvare “vakthund” sende programmet til en endestopp-funksjon som terminerer programmet og gir beskjed til den eksterne “vakthunden” om at den kan slå av spenningen. I så fall mister man en måling<sup>139</sup>.

```
// Programmet er skrevet av Thor Inge Hansen
// Programmet er modifisert av Nils Kr. Rossing
// 29.05.23 - Montert ENV-kort og testet SD-kort (nkr)
// 25.07.23 - Lagt inn mulighet til å velge bort MKR ENV dersom dette
ikke er tilkoblet (nkr)
// 25.07.23 - Oppdatert filen med kommentarer (nkr)

#include <DS18B20.h>
#include <OneWire.h>
#include <Arduino_MKRENV.h>
#include <SPI.h>
#include <SD.h>
#include <Arduino_MKRGPS.h>
#include <MKRNB.h>
#include <WDTZero.h>

WDTZero myWDT;
NBClient client;
GPRS gprs;
NB nbAccess;
DS18B20 ds(0);
File myFile;

//#define printState; // Kommenter ut for å slå av alle Serial print
//#define MKR_ENV; // Kommenter ut dersom kortet MKR ENV ikke brukes
```

---

139. Programmet er utviklet av Thor Inge Hansen ved Skien vgs. i samarbeid med Skolelaboratoriet ved NTNU



```
uint8_t address[] = { 40, 250, 31, 218, 4, 0, 0, 52 };
uint8_t selected;
char server[] = "sensor.marin.ntnu.no";
char path[] = "/cgi-bin/tof.cgi?";
char filename[] = "EVU-kurs-Grx.txt";
int port = 80;
bool connected = false; // Status oppkobling
char buffer[128];

unsigned long pause = 1000 * 60 * 6; //6 min
int shutOffPin = 1; // Port til shutOffPin
float w_temperature = 0;
float GPSlat = 0;
float GPSlong = 0;
unsigned long epochTime;
int numSat = 0;
float temperature = 0;
float humidity = 0;
float pressure = 0;
float illuminance = 0;
float battVolt = 0;

void setup() {
  pinMode(shutOffPin, OUTPUT);
  digitalWrite(shutOffPin, HIGH);
#ifdef printState
  Serial.begin(9600);
  while (!Serial) {}
  Serial.println("Ferdig_program_med_PICAXE_MILLIS_TEST_2");
#endif

  analogReadResolution(12); // 12Bits

  myWDT.attachShutdown(shutDownFunc); // Initialiser intern vakthund
```



```
myWDT.setup(WDT_SOFTCYCLE2M);           // Sett toimeout vakthund til ca.
2 minutter (128 sek)

while (!GPS.begin()) {}                  // Initialiser GPS-mottakeren

readWaterTemp();

readGPSdata();                          // Les GPS data
delay(500);
readENVData();
delay(1000);                             // Delay for å få stabile spenningsverdier
readBattVoltage();                       // Les batterispenning
delay(1000);
makeString();                            // Bygg opp bufferet
sdPrint();                               // Skriv til SD-kort
#ifdef printState
  serialPrint();
#endif
connectToGPRS();                         // Koble opp mot 4G (GPRS)
connectToServer();                      // Koble opp til server
delay(1000);
#ifdef printState
  Serial.println(millis());
#endif
digitalWrite(shutOffPin, LOW);           //Be PICAXE om å slå av spenningen
}

// Les av vanntemperatur
void readWaterTemp() {
  w_temperature = ds.getTempC();
}

// Les av batterispenningen
void readBattVoltage() {
  //Forutsetter lik deling av batterispenning mellom to like resistorer
  og at A1 brukes
```



```
battVolt = analogRead(A1) * 2 * 3.3 / 4096;
}

// Les av GPS-data
void readGPSdata() {

#ifdef printState
  Serial.println("Started readGPSdata");
#endif

  while (!GPS.available()) {} // vent til GPS er tilgjengelig

  GPSlat = GPS.latitude();
  GPSlong = GPS.longitude();
  epochTime = GPS.getTime();
  numSat = GPS.satellites();
}

// Les av miljødata
void readENVData() {
#ifdef MKR_ENV
  ENV.begin();
  temperature = ENV.readTemperature();
  humidity = ENV.readHumidity();
  pressure = ENV.readPressure();
  illuminance = ENV.readIlluminance();
#endif
}

// Bygg opp databufferet for overføring til server
void makeString() {
  sprintf(buffer, "%s%s,%u,%.6f,%.6f,%i,%.1f,%.1f,%.1f,%.1f,%.2f,%.2f",
    path, filename, epochTime, GPSlat, GPSlong, numSat, temperature, w_tem-
    perature, humidity, pressure, illuminance, battVolt);
}
```



```
// Koble til 4G GPRS
void connectToGPRS() {
#ifdef printState
  Serial.println("Started connectToGPRS()");
#endif

  while (!connected) {
    if ((nbAccess.begin("", true, true) == NB_READY) && (gprs.attachGPRS()
== GPRS_READY)) {
      connected = true;
#ifdef printState
      Serial.println("GPRS connected");
#endif
    } else {
      delay(1000);
#ifdef printState
      Serial.println("connected still False");
#endif
    }
  }
}

// Koble til server
void connectToServer() {
#ifdef printState
  Serial.println("in connectToServer()");
#endif

  if (client.connect(server, port)) {
#ifdef printState
    Serial.println("Connected to Server");
#endif

    client.print("GET "); // Gjør et HTTP request:
    client.print(buffer);
    client.print(',');
  }
}
```



```
    client.print(millis() + 10271); //Prints runtime + set delay + calculated offset to show the runtime untill shuOffPin turns low
    client.println(" HTTP/1.1");
    client.print("Host: ");
    client.println(server);
    client.println("Connection: close");
    client.println();

#ifdef printState
    Serial.println("String uploaded to Server");
#endif
} else {

#ifdef printState
    Serial.println("No connection to server");
#endif
}
}

// Skriv data-bufferet til SD-kort
void sdPrint() {

    if (!SD.begin(4))
        while (1)
            ; //STOPP OPP

    myFile = SD.open("EVU-kurs.txt", FILE_WRITE);

    if (myFile) {
        myFile.println(buffer);
        myFile.close();
    }
}

// Skriv ut databufferet til monitoren
void serialPrint() {
```





```
#ifdef printState
  Serial.println(buffer);
#endif
}

// Funksjon som ber PICAXE slå av spenningen
void shutDownFunc() {
  digitalWrite(shutOffPin, LOW);
  delay(5000);
}

void loop() {}
```







Prosjektet “Miljøovervåking med bøye” handler om å samle inn måledata fra kystnære strøk. Teknologien kan imidlertid brukes i mange ulike sammenhenger, i områder der man normalt ikke har tilgang til lokale nettverk slik at man kan overføre data via Wi-Fi. For å få til dette tar man i bruk ulike typer smalbånds mobildata som f.eks. NB IoT som Telenor med flere tilbyr.

Våren 2020 kom det en henvendelse fra Håvard Holm ved Institutt for marin teknikk som hadde en ide om å la elever bygge havgående bøyer med billige materialer og utstyre dem med elektronikk som kunne samle inn måledata fra havet og overføre disse til en server på land.

En kan tenke seg flere løsninger ved at man i større eller mindre grad bruker etablerte tjenester eller bygger opp tilbudet fra bunnen av og samler data til lokale servere. Vi har i dette tilfellet valgt å sende data til en lokal server ved Inst. for marin teknikk ved NTNU.

Hefte beskriver gangen fra ide og til man har etablert en forbindelse mellom målesensorer til data kan lastes ned fra serveren og analyseres ved hjelp av Excel, Python-script eller Google Earth. I tillegg beskrives aktuelle Arduino-komponenter og sensorer for måling i vann og en enkel bøye for uttesting.

### ***Nils Kr. Rossing***

Dosent emeritus ved Skolelaboratoriet  
Institutt for fysikk, NTNU  
E-post: [nils.rossing@ntnu.no](mailto:nils.rossing@ntnu.no)

### ***Thor Inge Hansen***

Adjunkt 1 ved Skien videregående skole  
E-post: [thor.inge.hansen@vtfk.no](mailto:thor.inge.hansen@vtfk.no)



Trondheim

Institutt for  
fysikk

Skolelaboratoriet  
for matematikk, naturfag  
og teknologi

Tlf. 73 55 11 43

<https://www.ntnu.no/skolelab>

