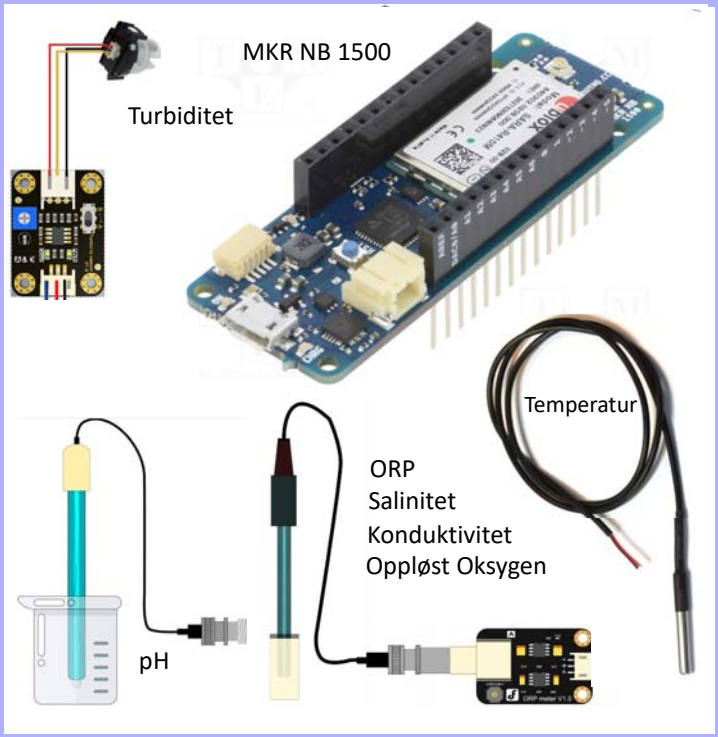
Rev 3.11

Nils Kr. Rossing

Miljøovervåking med havbøye, Arduino MKR NB 1500

Data til server ved Inst. for marin teknikk



NTNU
Trondheim
Institutt for fysikk
Skolelaboratoriet
for matematikk, naturfag
og teknologi

Desember 2022

Miljøovervåking med havbøye, Arduino MKR NB 1500

Data til server ved Inst. for marin teknikk

Nils Kr. Rossing, Skolelaboratoriet
Institutt for fysikk
i samarbeid med Institutt for marin teknikk

Miljøovervåking med havbøye, Arduino MKR NB 1500 – Data til server ved Inst. for marin teknikk

Trondheim 2022

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

Kursholdere: Nils Kr. Rossing
(Elektronikk/Progr.) 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

Skolelaboratoriet ved NTNU
Realfagbygget,
Høgskoleringen 5,
7491 Trondheim

Telefon: 73 55 11 91
https://www.ntnu.no/skolelab/

Rev 3.11 – 20.12.22

Forord

Prosjektet "Miljøovervåking med havbøye" 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. 4G 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. Det ble i den forbindelse utført en mulighetsstudie ved Skolelaboratoriet som ble videreført av to studenter samme sommer.

En kan tenke seg flere løsninger ved at man i større eller mindre grad bruker etablerte tjenester som den f.eks. Telenor tilbyr, eller bygger opp tilbudet fra bunnen av og samler data til lokale servere. Heftet beskriver en løsning der vi har valgt å sende data til en lokal server ved Inst. for marin teknikk. Hovedårsaken til dette valget var at Telenors i siste øyeblikk valgte å legge ned tjenesten høsten 2022 og heller komme tilbake med en bedre utgave høsten 2023.

Hefte beskriver gangen fra ide og til man har etablert en forbindelse mellom målesensorer til dataene kan lastes ned fra en server ved Inst. for marin teknikk og presenteres i Excel eller Google Earth.

Anvendelsen av denne type teknologiske løsninger er svært anvendelige. Tidligere er det kommet forespørsler fra Institutt for kjemisk prosessteknologi som ønsker å gjøre målinger på surhetsgrad 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.

Heftet inneholder også en gjennomgang av en rekke sensorer for registrering av parametere i havet som temperatur, pH, reduksjon-oksidasjons-potensial, ledningsevne/salinitet, turbiditet og oksygeninnhold.

En takk til student Johannes Ravn Munkvold som har lest korrektur og kommentert tidligere utgaver av heftet. Takk til Thor Inge Hansen ved Skien videregående skole som har lest gjennom heftet og kommentert. 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.

Skolelaboratoriet ved NTNU
Desember 2022
Nils Kr. Rossing

Innhold

1 Innledning

Heftet er en oppsummering av hvordan man etablerer en sensornode ved bruk av smalbånd mobildata 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 tilfredsstille våre behov i en uttestingsfase med en sensornode ombord på en havgående bøye. Dessuten legger vi opp til å legge datene ned på en lokal server ved Institutt for marin teknikk og visualisering av dataene ved hjelp av Excel og Google Earth. Prismessig er dette tilbudet også gunstig med en gratis prøveperiode på 5 mnd, kr. 15,- i etableringsgebyr inkludert 1 MB og deretter kr. 8,- for 1 MB/mnd. pluss kr. 1,49 pr. MB ut over de første 1 MB[1].

Vi har så langt ønsket å gjøre det så enkelt som mulig slik at det skal være mulig å komme fort igang. Et alternativ i så måte ville være å bruke Telenors tilbud, men da dette ennå ikke er ferdig utviklet har vi valgt, inntil videre, å laste dataene ned på den nevnte serveren.

Vi ønsker innledningsvis å tilby et kortkurs med lokale deltakere. Kursdeltakerne får en ferdig utstyrspakke med SIM-kort, dernest får de hjelp til å bygge opp hårdvare, 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. På sikt kan man se for seg at elevene selv bygger opp bøyer med sensornoder og utstyrer dem i henhold til egenutviklede prosjekter for å dekke lokale behov og interesser.

1.1 Oppbyggingen av undervisningsopplegget

1.1.1 Introduksjonskurs

Vi ser for oss et introduksjonskurs med følgende innhold:

- Introduksjon om Miljøovervåkning med havbøyer
- Litt bakgrunnskunnskap om “Internet of Things” (IoT) (Tingenes Internett)
- Kort introduksjon til mikrokontrolleren MKR NB 1500
- Lage et program som sender over måledata
- Sette opp kommunikasjonen via 4G til lokal server fra sensornoden (MKR NB 1500)
- Ta ned data fra serveren for visualisering, analyse og behandling
- Kort om aktuelle sensorer

Det er vanskelig å se for seg noe kortere enn et dagskurs dersom deltagerne skal få et inntrykk av hvordan systemet fungerer fra mottatte sensordata til man kan hente ned disse dataene fra skyen. Når datakanalen er etablert kan man konsentrere seg om selve nyttelasten og anvendelsen.

I tillegg er det viktig at de får de nødvendige marinbiologisk kunnskapene i tillegg oppbygging av en enkel bøye som kan holde elektronikken tørr og tåle litt røff sjø. Vi ser for oss et etterutdanningskurs på 3 dager.

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

På sikt vil det være naturlig å utvikle et mer omfattende kursopplegg, f.eks. i et videreutdanningskurs, med større vekt på bruk av ulike sensorer og uttesting.

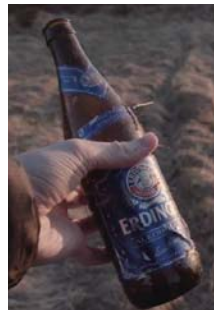
1.1.2 Videreutdanningskurs eller videregående kurs

Videreutdanningskurs vil gå over to samlinger og omfatter det samme som introduksjonskurset, men vil i tillegg legge vekt på følgende:

- Studie av miljø- og biologiske faktorer for observasjon
- Valg og montering av sensorer, ev. egenutvikling av sensorer
- Utvikling og bygging av vanntett beholder for elektronikk og lekkasjefrie gjennomføringer av kabler for eksterne sensorer og antenner.
- Metoder for energifangst og bruk av dvaletilstander for å spare energi
- Bygging av bøye ev. med fortøyning
- Analyse og presentasjon av mottatte data
- Uttesting i kontrollerbart miljø
- Uttesting i autentisk miljø
- Regulær drift over en kortere eller lengre tidsperiode

Kurset kan enten realiseres som et introduksjonskurs med påfølgende videregående kurs, eller som et videregående kurs.

1.2 En spennende historie?



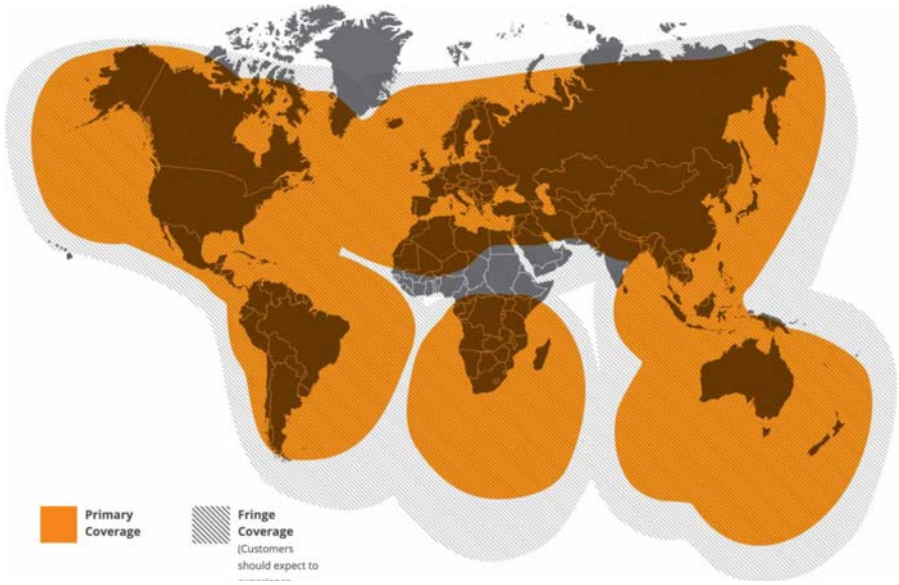
For noen år siden gjestet den tyske filmskaperen ***Steffen Krones*** fra Dresten, Norge. I løpet av turen besøkte han bl.a. en liten øy i Lofoten. Han ble slått av hvor mye plastsøppel som hadde samlet seg på strendene, og overraskelsen ble stor da han fant en tysk ølflaske blant alt søppelet. Kom den 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 bøyenes 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[2].

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 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[3]. 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.



2. https://static.fleetmon.com/static/downloads/FleetMon_S1-C_Satellite_Transponder.pdf
3. https://www.fleetmon.com/services/satellite-tracker/

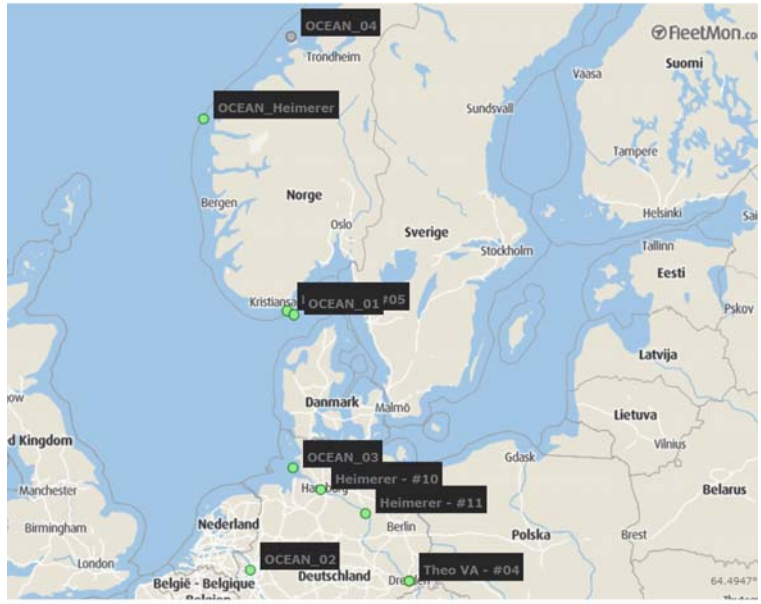
Bøyene var enkle, men bygget slik at de skulle kunne følge Elben ut til Nordsjøen og ev. videre ut på havet. Etter slipp av et titalls bøyer, fant de at flere endte opp langs norskekysten. Ett par landet blant annet utenfor Trøndelag og til og med så langt som til Lofoten[4]. Bøyene overførte posisjonen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

Simulering av bøya laget av resirkulert plast og kork

Bøya stikker opp av havet

En bøye havnet i Lofoten

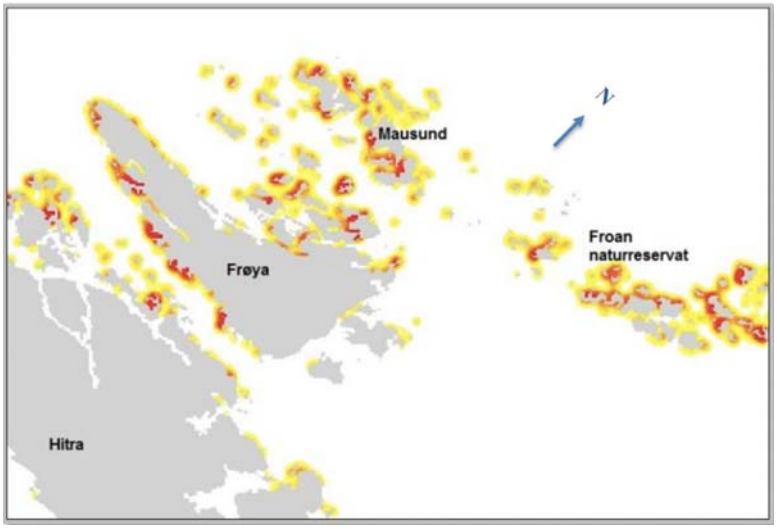
Under arbeidet med filmen søkte Steffen Krones kontakt med anerkjente tyske forskere som ***Dr. Melanie Bergmann*** og ***Dr. Lars Gutow*** ved **Alfred Wegener Institute** og fikk hjelp til å forstå hva som skjer med søppel som havner i de tyske elvene og etter hvert ender i havet. Han fikk også kontakt med den norske inuiten ***Kris*** som kjenner norskekysten godt etter utallige turer i kajakk. Tre år jobbet han med filmen som vil være klar til å vises våren 2022.



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.

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

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 det mange ting man kan tenke seg å måle. I det neste avsnittet skal vi se på noen aktuelle målinger.

1.3 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 Trolla har mye kunnskap om sensorer mm)
- Trykk

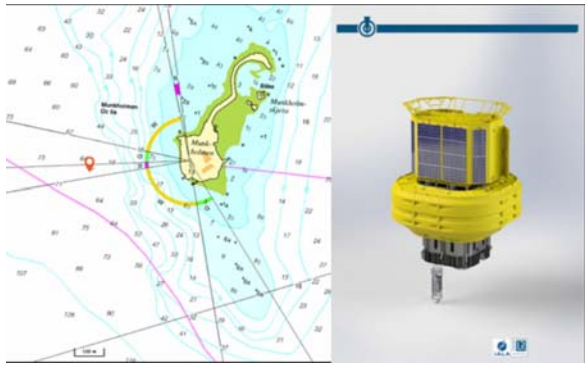
Trykk kan være lufttrykk, men om en bøye ankres opp kan måling av vanntrykk danne grunnlag for beregning av dybde, og slik gjøre det mulig å bestemme f.eks. temperatur og saltholdighet som funksjon av dybden.

Men også

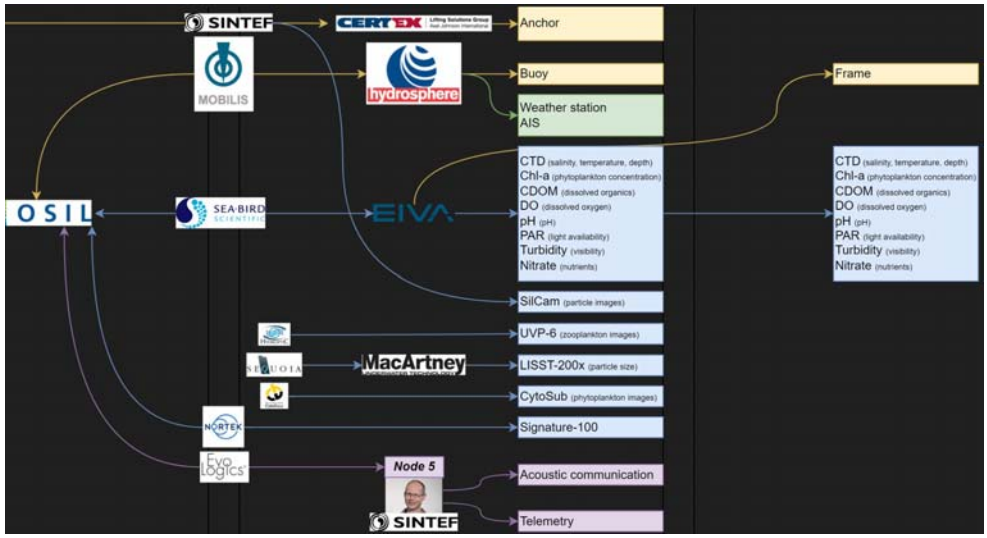
- pH
- Oksygeninnhold

... 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.

Vi har også vært i kontakt med forsker ***Emlyn John Davis***[5] som er kontaktperson ved 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 utstyret med en rekke sensorer som er levert og betjenes av ulike institusjoner som vist på figuren under[6]. Sensorene som er levert av EIVA er senkbare slik at den posisjoneres i ulike dybder.



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)

5. emlyn.davis@sintef.no
6. https://oceanlabobservatory.no/#Equipment

- Chl-a (Phytoplangton 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 Zooplangton)
- LISST-200X (Partikkel størrelse)
- CytoSub (Bilder av Phytoplangton)
- 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.

Institutt for marin teknikk ved Førsteamanuensis ***Håvard Holm*** har også vært i kontakt med Meteorologisk institutt som kan være interessert i værdata, 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.

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 ev. 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.

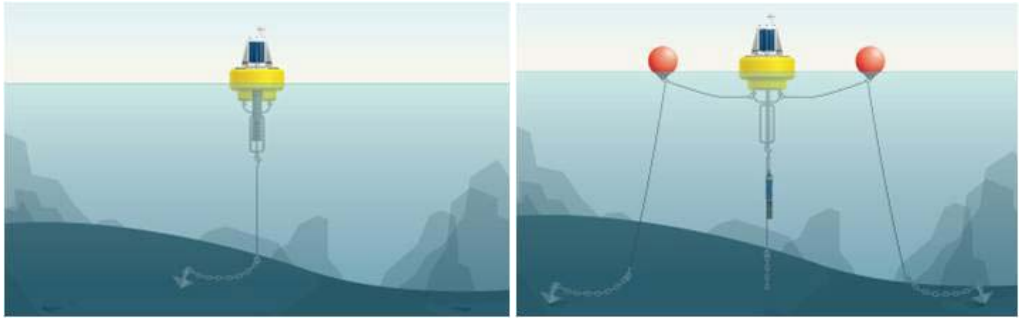
1.4 Råd mht til bygging av bøye og innpakking av elektronikk[7]

En eventuell oppankring må meldes til Sjøfartsdirektoratet med posisjon. 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.

7. Råd gitt av Odd Arne Arnesen – Mausund Feltstasjon

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 bøyene i egen boks. Bøyene kommer i alle størrelser. Jo mindre bøyen er, jo større utfordringer vil man kunne få i grov sjø.

Elektronikken kan ev. 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 batteri. 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 bøyen på land. Dette er spesielt viktig dersom målinger skal skje over lang tid slik at det kan være nødvendig med batteriskifte eller reparasjoner.



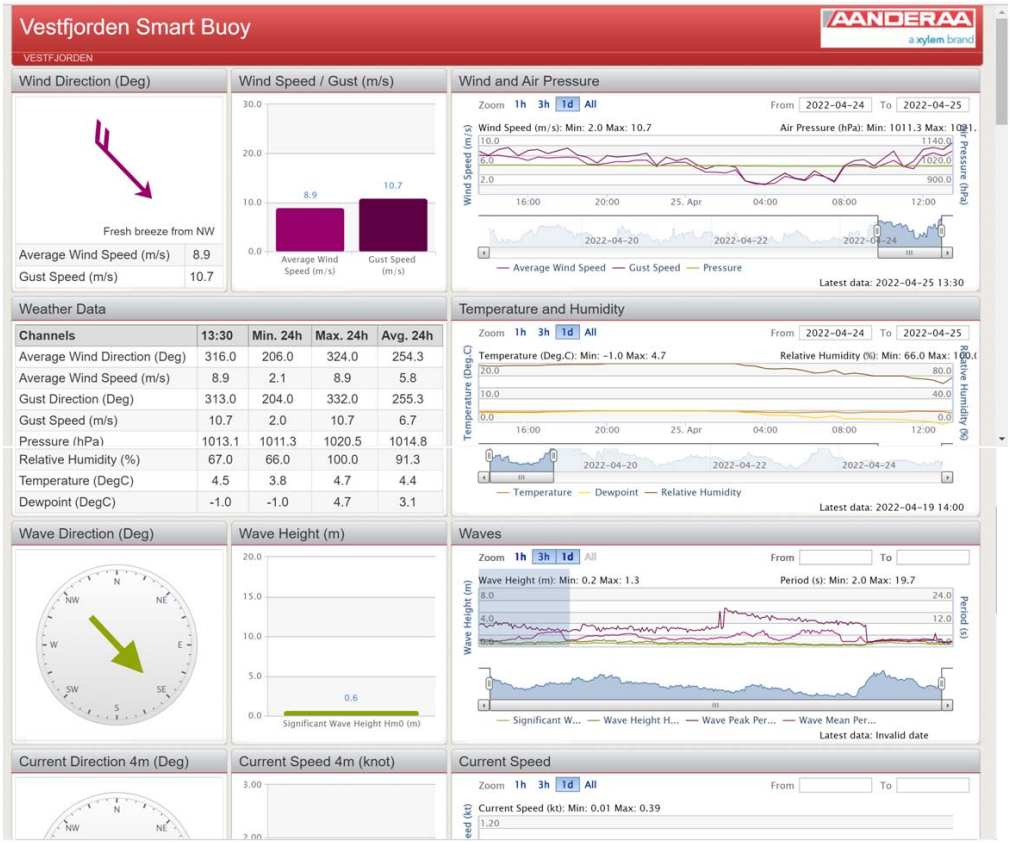
På figuren over er vist en-punkt og to-punkts oppankring. Det er viktig at bøya ikke blir dratt under ved flo og høy sjø, samtidig som den ikke har for slakk line slik at den driver av.

Våren 2022 er det lagt ut en bøye i Vestfjorden nær Lofoten som en hjelp til skipsfart og fiskefartøyer. Måle data er gjort tilgjengelig via Internett.

Dersom man går til følgende nettadresse får man opp en side med oversikt over data som oppdateres fortløpende:

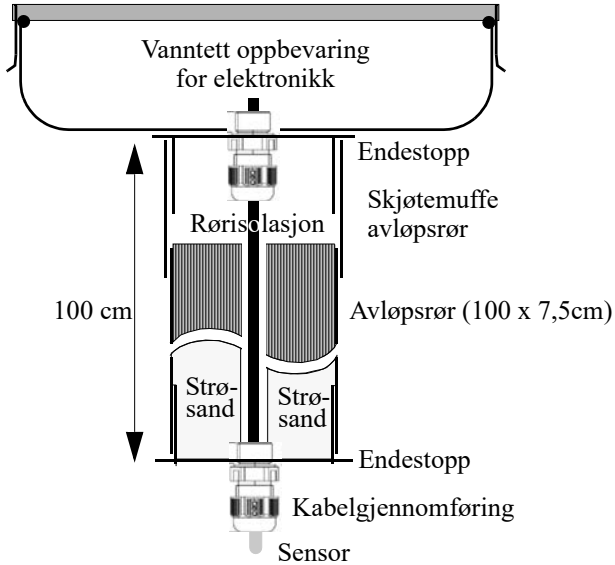
http://s2.data.aanderaa.com/AADI_DisplayProgram/setups/HC_vestfjorden/default.aspx

Figuren under viser et utsnitt av målinger.



1.4.1 Utforming av bøye for sjøvannstest 1 – et eksempel

I dette eksempel skal vi bruk avløpsrør med tilhørende koblingstykker[8]. Elektronikken plasserer 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 elektrikerrør. Vi bruker vanntette gjennomføringer for å føre ledninger og sensorer gjennom bunnen til bøya og ut i vannet.

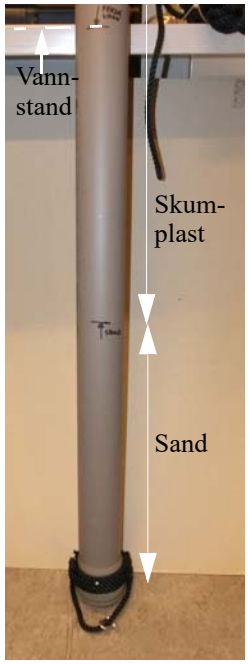


Avløpsrøret fylles med sand i bunnen slik at røret flyter stående i vannet. Sand har normalt en egenvekt på 1,4 –1,5 avhengig av type sand. Volleyball-sand er den med minst partikler og har en tetthet på 1,5. Øverste del av røret fylles med isolasjonsmateriale for rør. Dette er for å unngå at røret synker 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.

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. Gjengestykket hos kabelgjennomføringen er litt marginalt. Det er derfor lite plass til pakninger.

8. Ideen stammer fra Håvard Holm IMT, NTNU

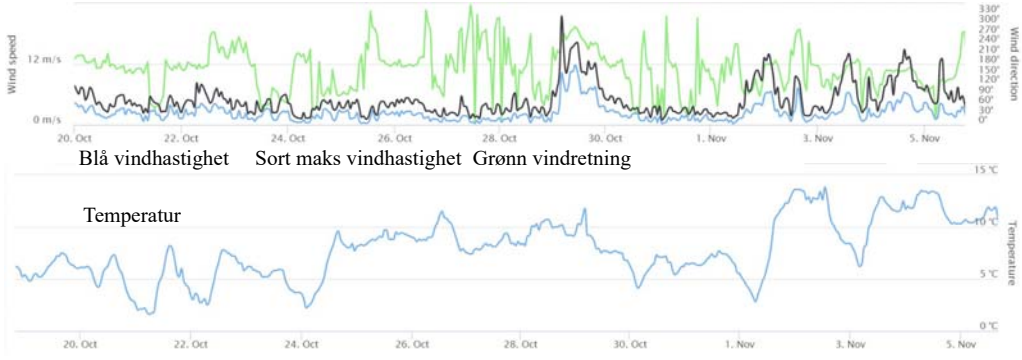
Bildene under viser detaljer fra første forsøk med bøye.



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.

1.4.2 Sjøvannstest 1

Bøya uten elektronikk ble lagt ut ved Trondheim Biologiske Stasjon (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 kloakkrøret som befant seg under vann, eller i boksen som var montert på toppen av kloakkrøret. 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.



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


Foto: Jussi Evertsen


Foto: Nils Kr. Rossing

Foto: Nils Kr. Rossing

1.4.3 Utforming av bøye for sjøvannstest 2 – et eksempel

I denne testen skal følgende ha fokus:

Bøya skal ha følgende utstyr:

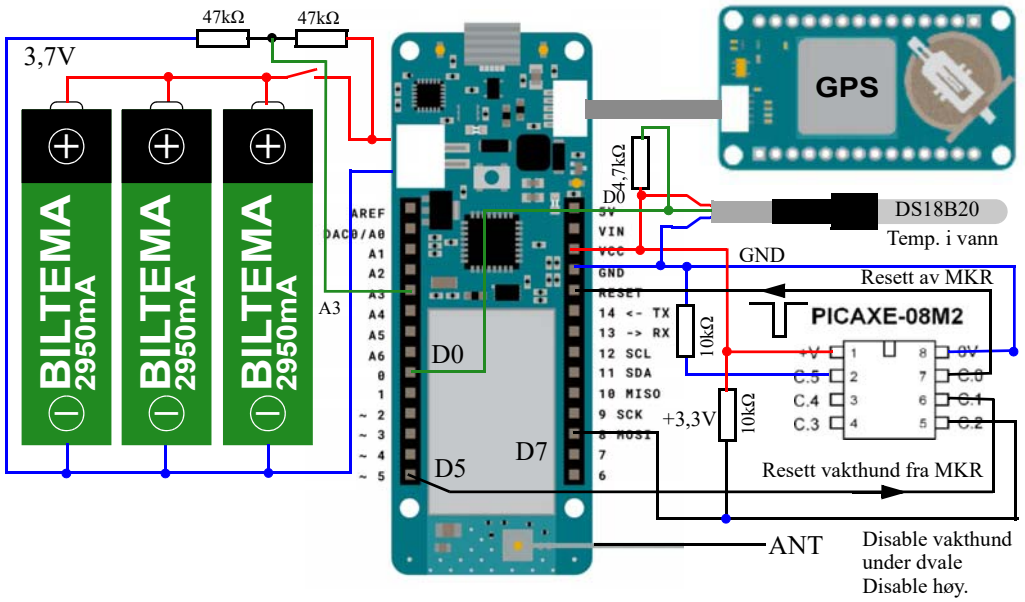
- Mikrokontroller for innsamling av måleresultater og kommunikasjon med server
- GPS-mottaker for å bestemme lokasjonen til bøya
- En enkel temperatursensor i bunnen av bøya
- Et utvendig sensorkammer med ledning og fiktiv sensor, for gi erfaring med montering og innhenting av måledata på denne måten.

Oppkobling

For dette formålet velger vi en relativt enkel oppkobling som baserer seg på å legge mikrokontrolleren i dvale. Vi velger følgende parametere:

- Måling hvert 10. min.
- Måling av sjøtemperatur og GPS-koordinater
- Måling av batterispenning
- Total batterikapasitet ca. 10000 mAh
- Velger å la "vakthunden" stå tilkoblet batteriet hele tiden, årsaken er at når "vakthunden" er uten spenningen vil reset være lav og mikrokontrolleren vil ha problemer med å våkne. Strømtrekket fra "vakthunden" er ca. 0,6 mA.

- Vi lar MKR disable "vakthunden" under dvale for så å slå den på ved oppvåkning. Dette gjøres ved å legge C2 høy under dvale.



Utvendig sensorkammer

Et av flere forslag er å montere sensorer av typen pH, konduktivitet, oksygeninnhold og turbiditet i utvendige sensorkammer, også bygget opp av deler til kloakkrør 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 sjøvann 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 tusj 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 nipler for gjennomføring av ledninger. Disse smøres med litium fett.
- Det brukes silikonpakninger mellom niplene og endestykkene. Disse smøres også inn med litiumfett
- Pakningene på innsiden av rørmuffene smøres med litiumfett, både på innsiden og på utsiden

Instrumentboks

Mikroprosessoren, 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.

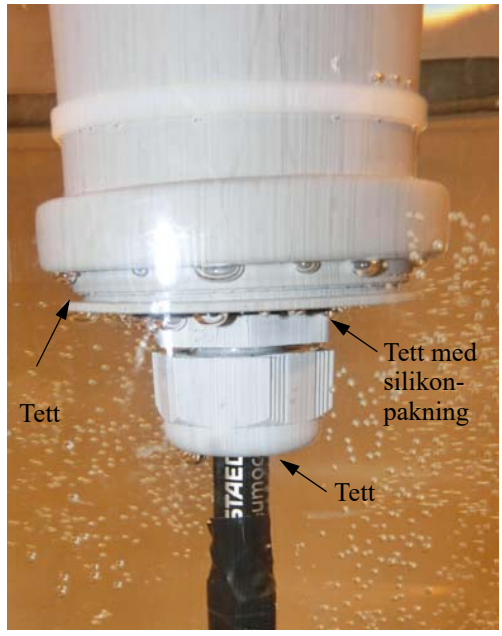
Det er flere kritiske momenter her:

- Innslag av fuktighet, noe som meget vel kan skje

- 4G-antenne får kummerlige betingelser nær batteriene. Likeså GPS-antennen som er plassert ned mot bakken og ikke opp mot himmelen.
- Ledningene løsner fra koblingsbrettet.

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 den lekket.



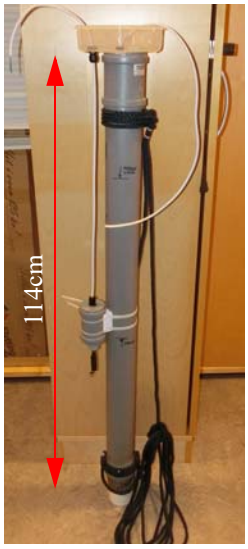
Et lodd henges under for å holde det under vann. En relativt effektiv måte å teste tetthet på er å blåse luft inn i kammeret og se hvor lufta tyter ut. Det viste seg at ledningen ikke var gjennomgående tett slik at det var mulig å blåse luft inn i kammeret. Vi oppdaget da at pakningen rundt nippelen var utett. 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 gjor med selve bøya med tempratursensor 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 nippelen under instrumentboksen som vi monterte på toppen av bøya med to nipler gjennom et endestykke. I tillegg er det montert en nippel i bunnen av instrumentboksen hvor ledningen kommer opp fra sensorkammeret.



1.4.4 Sjøvannstest 2

Planlagt uke 47, 2022.

2 Internet of Things

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

2.1 Definisjon og nytteverdi[9]

> *Internet of Things er et system av sammenkoblete 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, ev. regulering av lys og temperatur i drivhus.*

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

I bynære områder kan en se for seg andre nyttige anvendelsesområder:

> *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.*

På et mer personlig plan kan en se for 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*

9. https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT

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 finnes alt i dag.

2.2 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 en kald drikke.


Kevin Ashton (1968 –) *Neil Gershfeld (1959 –)*

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 klare 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.

2.3 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.

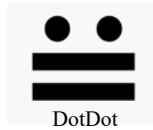
Siden det foreløpig ikke finnes noen enhetlig internasjonal standard på dette feltet så oppstår det lett problemer når ulike systemer skal utveksle data.

2.4 Internasjonal standard

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 Internet. 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[10] for IoT som gjør det mulig for "smarte objekter" å kommunisere trygt og forståelig med hverandre på alle nettverk.


DotDot

- **LiteOS** er et Linux-basert operativsystem for trådløse sensornettverk. LiteOS støtter Smarte telefoner og klokker, wearables[11], 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.

10. https://zigbeealliance.org/solution/dotdot/
11. 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.

3 Arduino MKR NB 1500

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 hårdware 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 er anbefalt av Telenor. Det finnes forøvrig flere andre varianter.

3.1 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 en krypteringskrets, ECC508 som også leveres av Microchip. Kommunikasjonsenheten er en SARA-R410M-02B levert av firmaet u-blox[12]. 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).

Power supply[13]**:** 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.

12. 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)
13. https://core-electronics.com.au/arduino-mkr-nb-1500-44916.html

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.

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-konverter for å løfte spenningen fra 3,7 V til 5V.

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 å unngå å gå særlig høyere enn 5V.

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 fra Power Bank eller fra annen batterikilde via V_{in}.

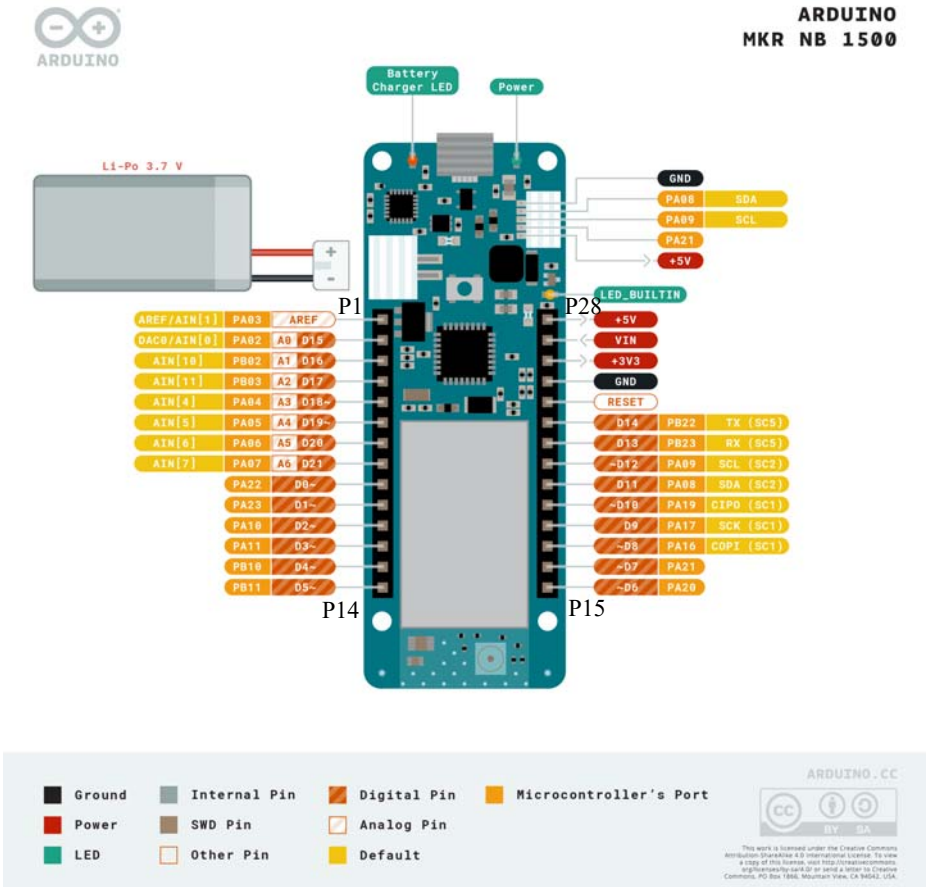
Inn- og utganger: Kortet har 8 unike digitale I/O-porter, hvor det interne LED-lyset er koblet til port 6. I tillegg har kortet 7 analoge innganger, som også kan benyttes som digitale I/O-porter etter behov. AD-konverteren knyttet til de analoge inngangene kan settes til 8, 10 eller 12 bit. I alt kan 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);[14]

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

14.https://www.arduino.cc/reference/en/language/functions/analog-io/analogreference/

Pinning: La oss se på noen sentrale parametere knyttet til mikrokontrollerkortet. Figuren under viser pinningen til MKR NB 1500.



Det er dessuten en holder for mikro SIM-kort.

Frekvensbånd og datarate[15]**:** 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 sender så mottar, eller omvendt, mens *Full dupleks* betyr at det sender og mottar 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.

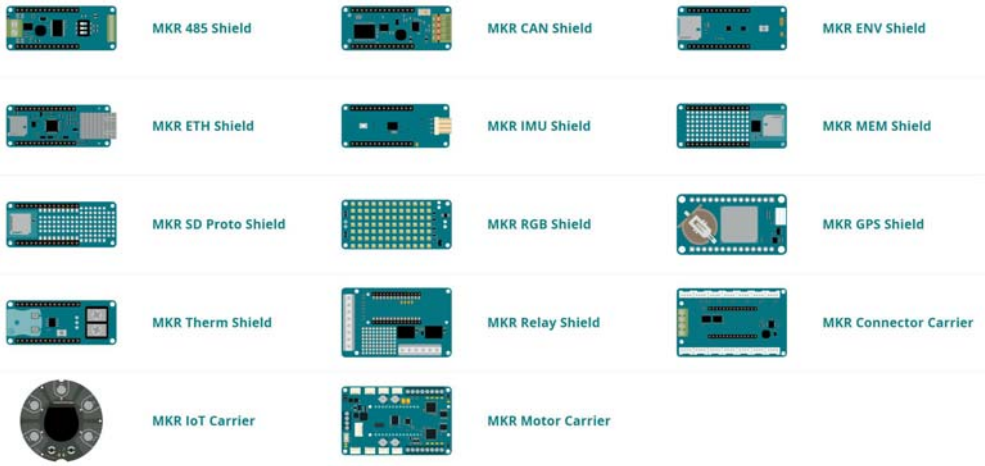
Kommunikasjon: Kortet har også UART (Tx/Rx), en I^2C, og en SPI.

15.https://www.sqimway.com/lte_band.php

Shield-kort: Det finnes en rekke Shield-kort som kan plugges ned i hovedkortet.

3.2 Shield-kort

Dette er kort som kan monteres på "ryggen" av MKR NB 1500 eller kobles til med en 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 benyttet felles I^2C-buss for å kommunisere med mikrokontrolleren. Her er en oversikt over brukte I^2C-adresser.

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)

3.2.1 Oversikt over brukte pinner på MKR NB 1500

Tabellen under viser hvordan pinnene på MKR NB 1500 er organisert når shild-kortene MKR ENV og MKR GPS plugges inn som "piggyback".

Tabell 1: Shield-kortenes bruk av pinner

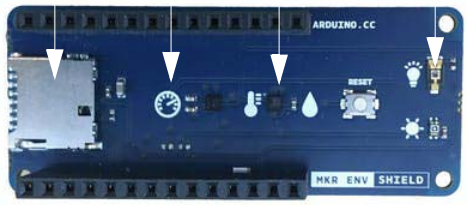
Port #	Port funksjon	Shield	Port #	Port funksjon	Shield
1	AREF		15	D6~	ENV DRDY Temp, Hum
2	A0		16	D7~	ENV DRDY, Air pressure
3	A1		17	D8~	Power "vakthund"
4	A2	ENV LYS	18	D9~	
5	A3		19	D10~	

Port #	Port funksjon	Shield	Port #	Port funksjon	Shield
6	A4		20	D11~	ENV SDA I^2C GPS SDA I^2C
7	A5		21	D12~	ENV SCL I^2C GPS SDA I^2C
8	A6		22	D13~	
9	D0~	DS18B20 VannTemp	23	D14~	
10	D1~		24	RESET	
11	D2~		25	GND	
12	D3~	Interrupt fra ext. RTC	26	+3,3V	
13	D4~	ENV SD CS	27	Vin	
14	D5~	pinWDReset	28	+5V	

3.2.2 MKR ENV Shield

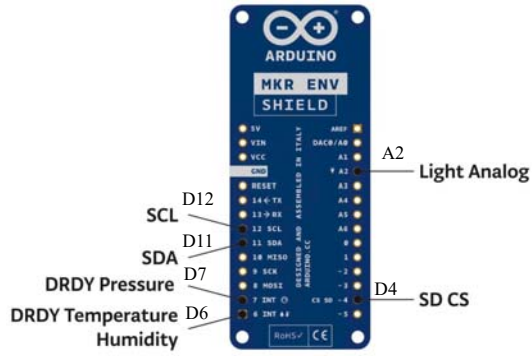
Kortet inneholder temperatur-, luftfuktighet-, lufttrykk, lyssensor og sensor for måling av UV-stråling (kun eldre kort). I tillegg har kortet en SD-kortleser for å lagre data.

SD-kortleser Trykk Temp/Fukt Lys



Tilkoblinger[16]

Kortet bruker både I^2C, SPI og en av de analoge inngangene. I^2C brukes for å kommunisere med lufttrykk, temperatur, luftfuktighet og UV-stråling (som fantes på de opprinnelige kretsene). I^2C kobles til port D12 (SCL) og D11 (SDA). Lysmåleren er koblet til analoginngangen A2 og måler lysintensiteten i Lux. SD-kortleseren er koblet til SPI grensesnittet hvor D8 (MOSI),



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

D9 (SCK) og D10 (MISO) står for kommunikasjonen. I tillegg benyttes D4 SD CS (SD-kort Chip Select) for å velge kortet. Videre er port D6 koblet til HTS221 (temperatur og luftfuktighet) DataReaDY, og D7 er koblet til LPS22HB (lufttrykk) DataReaDY som vist på figuren til høyre.

Lufttrykk (LPS22HP – I^{2}C adresse 0x5C)[17]

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

Kretsen kan kommunisere med omverdenen via I^{2}C og SPI og kan arbeide med spenninger fra 1,7 – 3,6 V og strømforbruket er så lavt som 3 µA.

Luftfuktighet og temperatur (HTS221 – I^{2}C adresse 0x5F)[18]

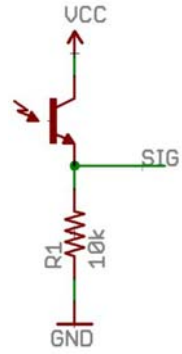
Sensoren har et måleområde fra 0 – 100 % relativ fuktighet (RH) med en oppløsning på 16 bit, og en nøyaktighet på ± 3,5% fra 20 – 80 % RH. I tillegg måler den temperaturen med 16 bits oppløsning med et måleområde fra –40°C til 120°C, og en nøyaktighet på ± 0.5°C, fra 15 til +40 °C, temperaturmålingen bruks bl.a. til å beregne relativ fuktighet. Dataraten (ODR) kan settes til mellom 1 til 12,5 måling pr. sekund.

Kretsen kan kommunisere med omverdenen via I^{2}C og SPI.

Kretsen kan arbeide med spenninger fra 1,7 – 3,6 V og strømforbruket er så lavt som 1 µA ved målerate på 1Hz.

Lyssensor (TEMPT6000 – A2)[19]

Sensoren er en fototransistor med følsomhetsvinkel på ± 60° 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[20]. 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.

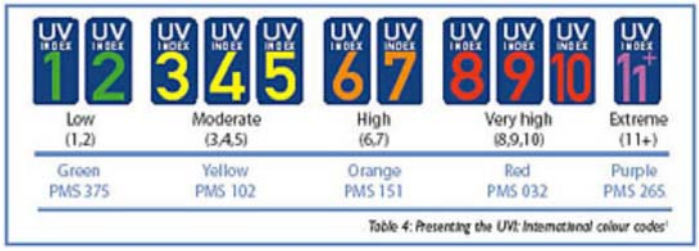


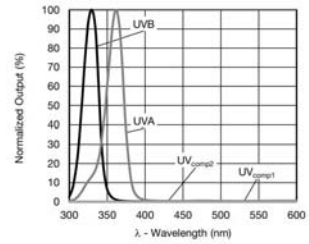
Strømforbruket varierer med lysstyrken der typisk strøm ved 1000 Lux er 0,5 mA. Ved sterkt lys kan strømmen bli vesentlig høyere.

17. https://www.st.com/resource/en/datasheet/dm00140895.pdf
18. https://www.st.com/resource/en/datasheet/hts221.pdf
19. https://www.st.com/resource/en/datasheet/hts221.pdf
20. 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^2C-buss for å overføre sensordataene. Tabellen under gir en indikasjon på faregraden for de ulike verdiene[21], mens grafen til høyre viser normalisert spektralrespons for UVA (ca. 330 nm) og UVB (ca. 360 nm)[22].





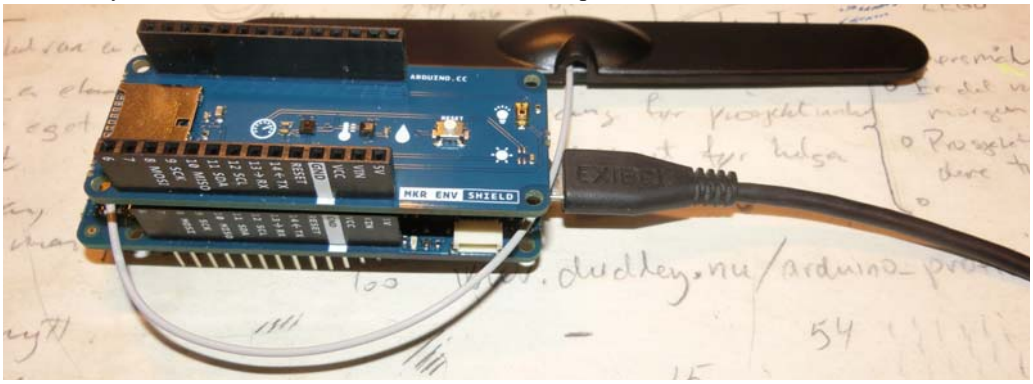
Kretsen kan operere med spenninger fra 1,7 – 3,6 V.

Energiforbruk

Det har foreløpig ikke vært mulig å bestemme 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 µA. 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 monteres på toppen av kontrollerkortet. PASS PÅ å sette kortet rette vei. Teksten på siden av hylsekontaktene viser hvilken vei kortet skal plasseres.

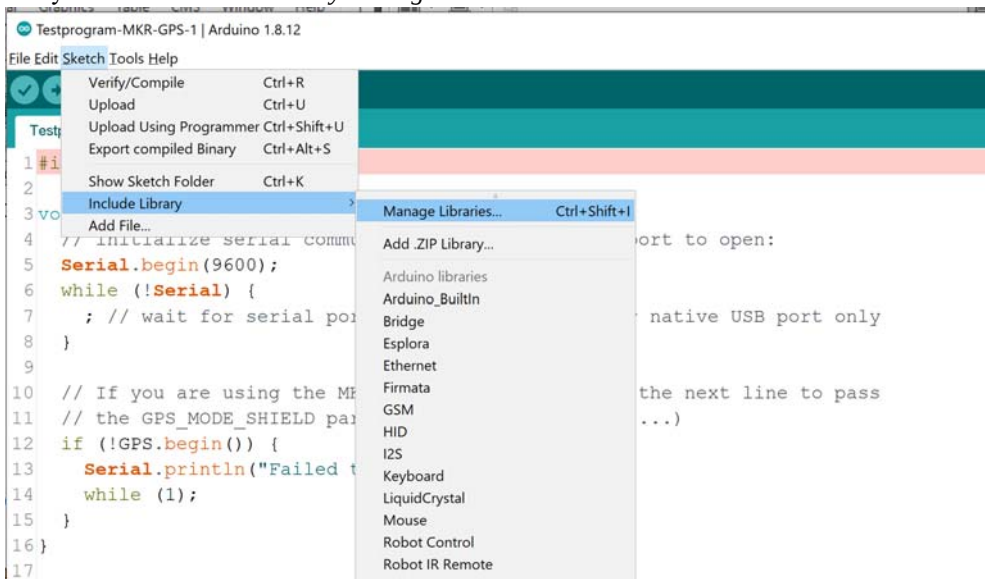


21.https://www.who.int/news-room/questions-and-answers/item/radiation-the-ultraviolet-(uv)-index
22.http://www.farnell.com/datasheets/2245219.pdf

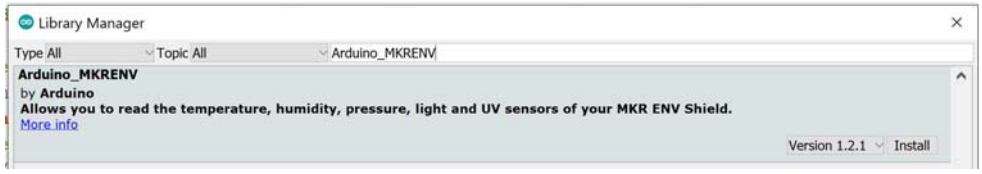
Programmering – Testprogram for MKR ENV (`Testprogram-MKR-ENV-1.ino`)

1. Installasjon av bibliotek MKR ENV

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

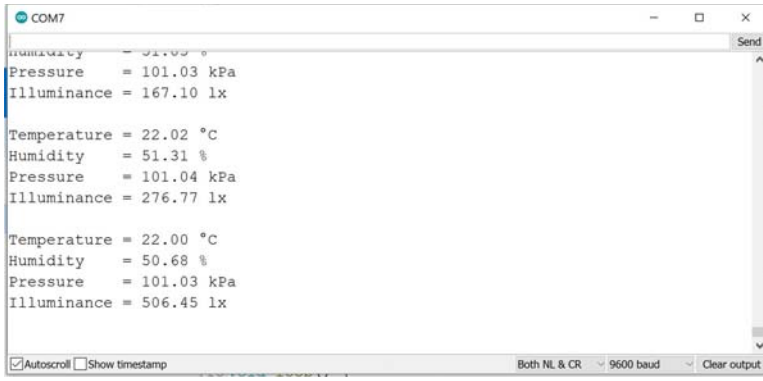


Vi får da opp følgende vindu hvor vi skriver inn *Arduino_MKRENV* i søkefeltet.



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

2. Kjør testprogrammet



Last opp testprogrammet, se avsnitt E.2.1 på side 204, og kjør programmet. Pass på å velg riktig port og åpne monitor-vinduet. Legg merke til at vi har kommentert bort koden som omhandler UVA og UVB siden vårt kort mangler denne sensoren. Om alt har gått som planlagt skal du nå se din egen posisjon 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, når vi senere skal inkludere deler av programmet i hovedprogrammet vårt.

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 kan måleverdier 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 deklareres og tilordnes avleste verdier på samme linje.

Skriving til SD-kort[23]

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

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 deklareres før void setup()-funksjonen.

4. **Deklarer et objekt for filbehandling**
   Vi deklarerer et objekt `dataFile` av typen `File`

   ```
   File dataFile;
   ```

   Denne deklareres 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

23.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 har oppstått 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 bli stoppet (`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. `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.

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(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.

9. **Lukk filen**
   Deretter lukker vi fila med følgende kommando:

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

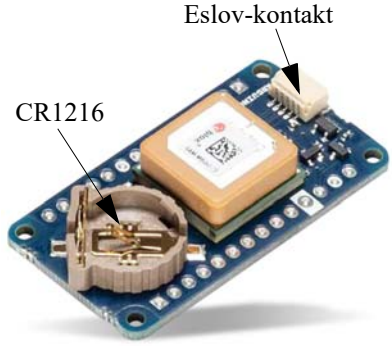
Testprogrammet ligger i vedlegg E.2.2 side 205.

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 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

3.2.3 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^2C. Brukes kabelen kommuniserer GPS-kortet med hovedkortet via I^2C-buss, men plugges det inn i "ryggen" på MKR NB 1500 benyttes UART (Rx/Tx)[24].

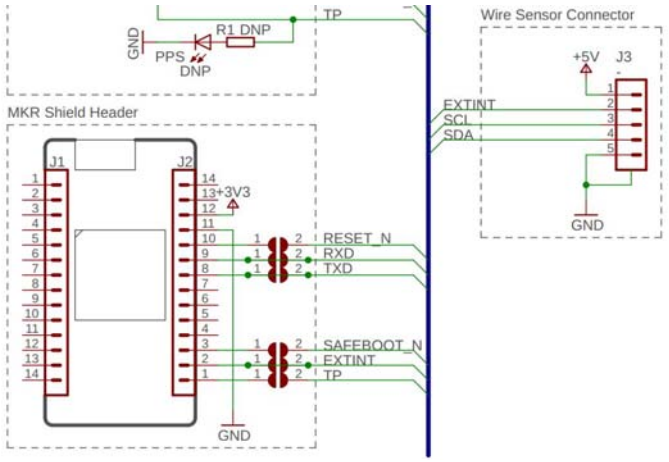


Vi anbefaler å bruke kabelen og Eslov-kontakten da det er I^2C-bussen som brukes i programvaren i denne anvendelsen.

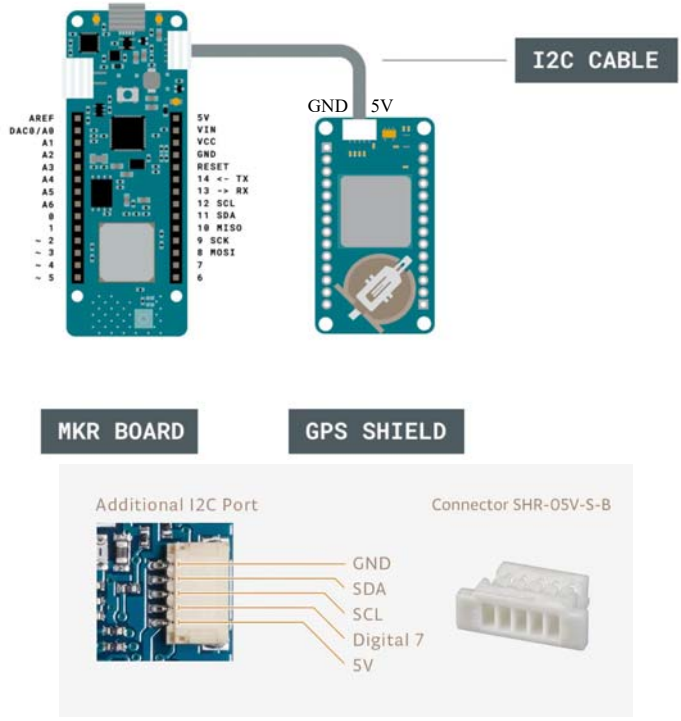
Kortet har også et backup batteri (CR1216) for å "huske" spesielle konfigurasjoner som program-vare-messig lastes opp til GPS-kretsen. Kortet arbeider på 3,3 V, og kan under spesielle tilfeller trekke opp til 40 mA.

24.https://www.farnell.com/datasheets/3317027.pdf

Figuren under viser den delen av koblingskjemaet[25] som omhandler headere og Eslov-kontakten. Her ser vi at de to kommuniserer på henholdsvis UART (Rx/Tx) og I^2C.



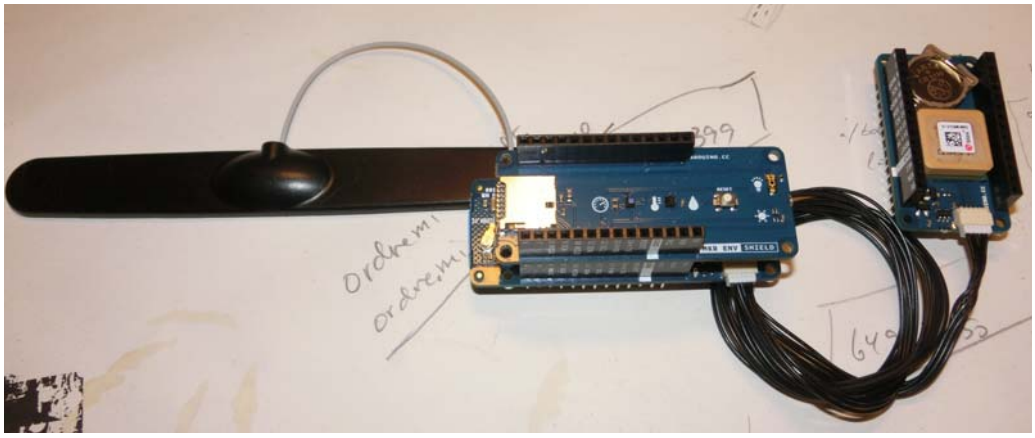
Figuren under viser hvordan MKR GPS kan kobles til MKR NB 1500 med kabel.



25.https://content.arduino.cc/assets/MKRGPSShield_V4.0_sch.pdf

Montering av MKR GPS

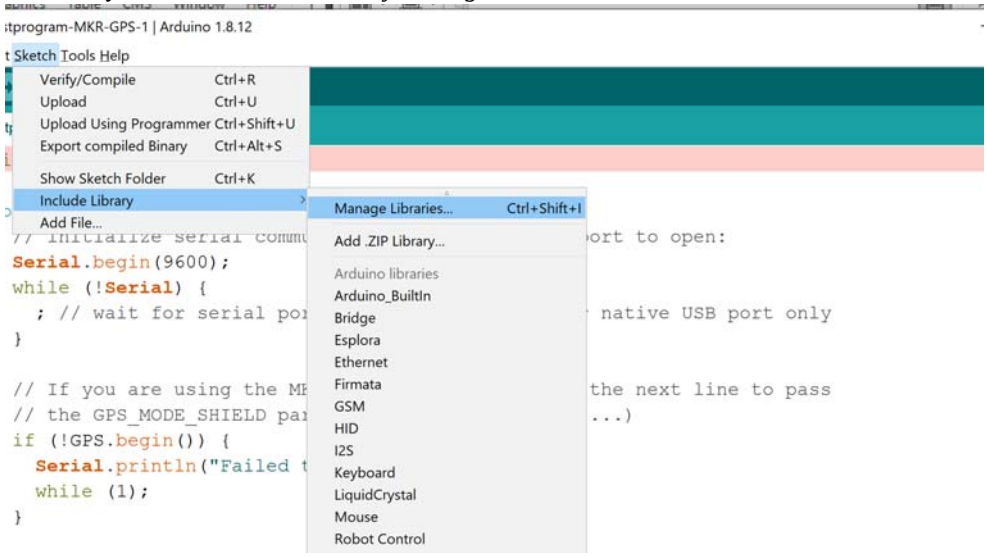
MKR GPS monteres med kabel til kontrollerkortet. Pass på å press støpselet helt inn i kontakten. På bildet er MKR GPS-kortet alt montert.



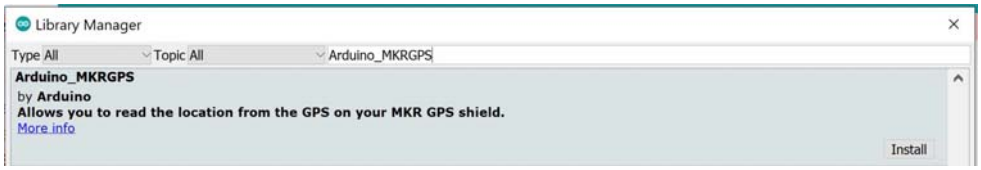
Programmering – Testprogram for MKR GPS (`Testprogram-MKR-GPS-2B.ino`)

1. Installasjon av bibliotek MKR GPS

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



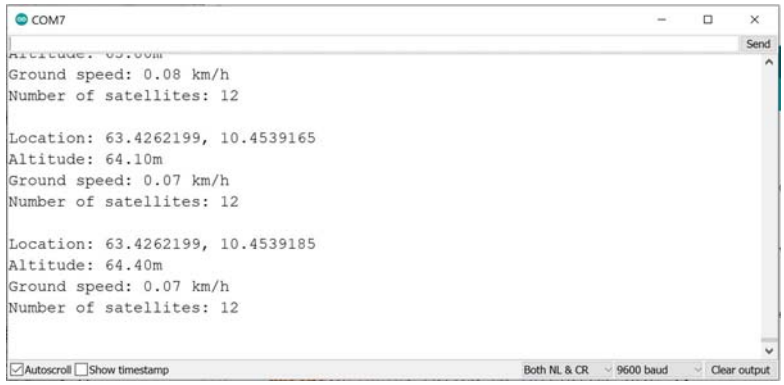
Vi får da opp følgende vindu hvor vi skriver inn *Arduino_MKRGPS* i søkefeltet.



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

2. Kjør testprogrammet

Last opp testprogrammet, se avsnitt E.2.1 på side 204, og kjør programmet. Pass på å velg riktig port og åpne monitorvinduet. Om alt har gått som planlagt skal du nå se din egen posisjon som vist på figuren over.



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 inkludere deler av programmet 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^2C-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.

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();
```

Vi legger merke til at variablene som holder de avleste verdiene deklareres 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/

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 uker og 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

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[26]. 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.

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

Sett MKR GPS i "stand by"[27]

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".

3. **Vent for nye GPS data**
   Dernest kan det ta noe tid for GPS-kretsen har låst seg til et sett med satellitter og er istand til å levere posisjonsdata.

   ```
   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 "vakthund", 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 µA.

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 blokkere 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 den 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.

27.https://www.arduino.cc/en/Reference/ArduinoMKRGPStandby

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-motakeren 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 = 2000;
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 2000 runder så vil den i starten ikke oppnå kontakt. Ved plassering i vindu så trenger den 200 – 500 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.

3.2.4 MKR MEM Shield

Kortet inneholder en lagringskrets, Flash memory 2MB (W25Q16), og en SD-kortholder 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 3.2.2 side 35.

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-kort holder (D4) og flash memory kretsen (D5).

3.2.5 MKR Termo Shield

MKR Termo Shield er et kort beregnet på måling av temperaturer over et stort temperaturområde, typisk fra –200 til +700°C eller mer. Kretsen Maxim 31855[28] gjør det mulig å bruke termokoblere (termoelement). Kretsen leverer temperaturen med 14 bits oppløsning som tilsvarer 0,25°C i et område fra –270 til +1800°C med en nøyaktighet på ±2°C. Temperaturen overføres via et SPI-grensesnitt.

Tilkoblingen kan skje via en K-type hunnkjønnet kontakt eller ved to skruterminaler.

Power supply: Kortet kan forsynes med 5V via Vin, men bruker en arbeidsspenning på 3,3V.

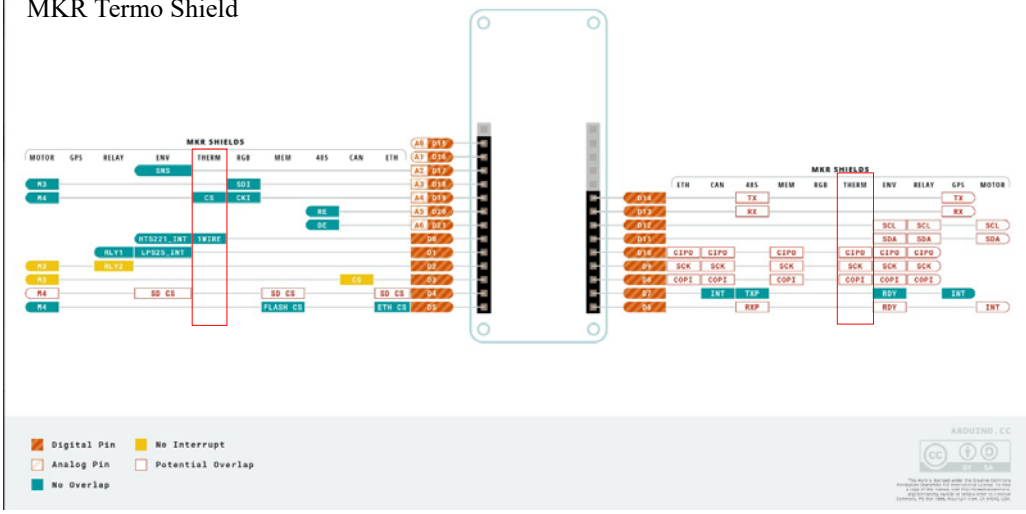
28.https://datasheets.maximintegrated.com/en/ds/MAX31855.pdf

Tilkobling: Kortet kommuniserer med mikrokontrolleren ved hjelp av følgende linjer:

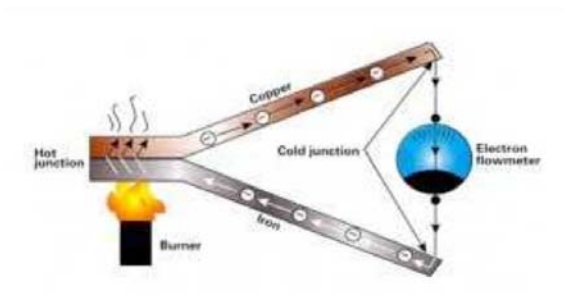
CS – D19	Ship select
1Wire – D0	1 wire kommunikasjonsbuss
CIPO – D10	Controller In Peripheral Out (SPI)
SCK – D9	Serial Clock (SPI)
COPI – D8	Controller Out Peripheral In (SPI)
GND	Jord
VCC	3,3V

Figuren under gir en fullstendig oversikt over pinningen til kortet

MKR Termo Shield



Termoelementet



Selve sensoren er et termoelement (termocoupler) som består av to ulike metaller som er forbundet i et punkt[29]. Det er koblingen mellom de to metallene som genererer en spenning som er avhengig av temperaturen.

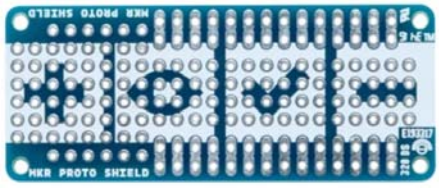
Siden det er sammenkoblingen av de to metallene som generer spenningen så vil også tilkoblingspunktene i motsatt ende der måleinstrumentet står, også være kritisk. Dette er det tatt hensyn til på dette spesielle shield-kortet.

Vi ser ingen grunn til å bruke et slikt kort for vårt formål.

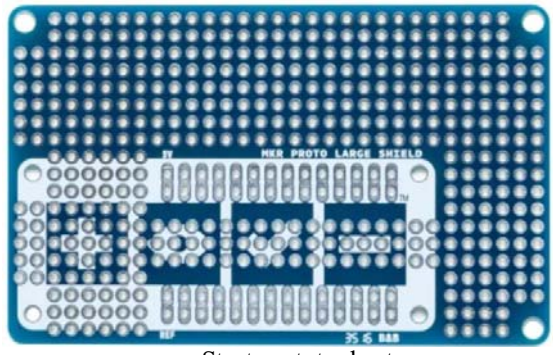
29.https://snl.no/termoelement

3.2.6 Prototyp-kort

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



Lite prototypkort



Stort prototypkort

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.

Vi har valgt å bruke det stor kortet her, da det gjør det lettere å montere koblingsplinter. Dersom de skal plasseres inne på det litte prototypkortet vil de lette komme i konflikt med korte som er tengt plassert lenger opp i stabelen. Se avsnitt 7.1.3 på side 122 for oppkobling for temperatur sensor for måling av vanntemperatur.

3.3 Dvale

Ved måling over lengre tid er det viktig at vi kan legge kretsen i dvale for så å vekke den opp når det skal utføres en måling som skal overføres til serveren.

For å kunne legge kretsen MKR NB 1500 (og andre i samme serie) i “sleep” og “deep sleep” benytter vi biblioteket[30]:

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

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

- **LowPower.idle(milliseconds);**[31]
  Dette legger kretsen i lett søvn slik at den raskt kan vekkes. “milliseconds” angir hvor lenge den skal ligge i lett søvn før den vekkes. Dersom “milliseconds” sløyfes vil den sove til den vekkes. Funksjonen kalles i programmet der man ønsker at dvale skal inntreffe.

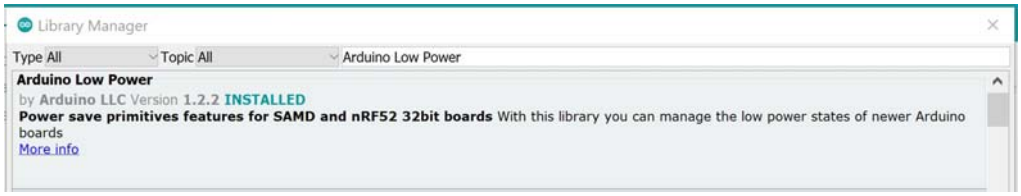
30.https://www.arduino.cc/en/Reference/ArduinoLowPower
31.https://www.arduino.cc/en/Reference/LowPowerIdle

- **LowPower.sleep(milliseconds);**[32]
  Denne kommandoen legger kretsen i dypere søvn med noe lenger oppvåkningstid, og med lavere strømforbruk. ”Milliseconds” angir hvor lenge den skal ligge i lett søvn før den vekkes. Dersom ”milliseconds” sløyfes, vil den sove til den vekkes. Funksjonen kalles i programmet der man ønsker at dvale skal inntreffe.
- **LowPower.deepSleep(milliseconds)**[33]**;**
  Alt unntatt RTC er lagt i dvale. Denne modusen har da den lengste oppvåkningstiden og det laveste effektforbruket. ”Milliseconds” i argumentet angir hvor lenge den skal ligge i dyp søvn før den vekkes. Dersom ”milliseconds” 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[34].

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 dette biblioteket.

3.3.1 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

32. https://www.arduino.cc/en/Reference/LowPowerSleep
33. https://www.arduino.cc/reference/en/libraries/arduino-low-power/lowpower.deepsleep/
34. 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 deklareres 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 oppvåkningsfunksjon, 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 egen skrevet funksjon.

   ```
   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 der 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. **Eksempelprogram**
   Det ligger et eksempelprogram i vedlegg E.2.11 side 219.

3.3.2 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 deklareres 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 når det vekkes 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. **Eksempelprogram**
   Det ligger et eksempelprogram i vedlegg E.2.10 side 218.

3.4 Reduksjon av strømforbruket

Eksempelprogrammene i vedlegg E.2.11 side 219 legger kretsen i dvale og tar den ut av dvale etter 10 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[35].

Batteriplugg: Batteripluggen på kortet er av typen hankjønn 2pin 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) er til høyre[36].

Valg av batteri: Det anbefales sterkt 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 varm. 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.

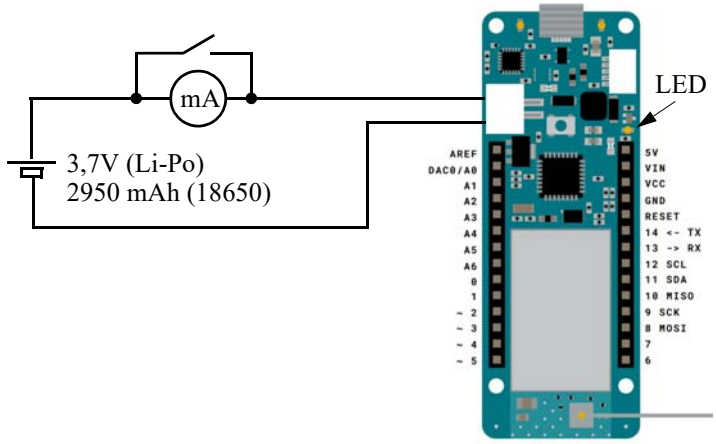
V_{in}: Dersom man ønsker å tilføre strøm gjennom V_{in} så vil den optimale spenningen være 5V og må ikke under noen omstendigheter overskride 6V.

I følgende avsnitt har vi foreslått ulike måter å håndtere effektforbruket på.

La oss se litt på strømforbruket.

3.4.1 Strømforbruk under ulike forhold

Vi tar utgangspunkt i følgende testkrets:



For å kunne kjøre testprogrammet må bibliteket ***Arduino Low Power*** installeres.

35.https://core-electronics.com.au/arduino-mkr-nb-1500-44916.html
36.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 deepSleep – 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 deepSleep – 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 deepSleep – 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

3.4.2 Tips ved bruk av deepSleep

Følgende er viktig:

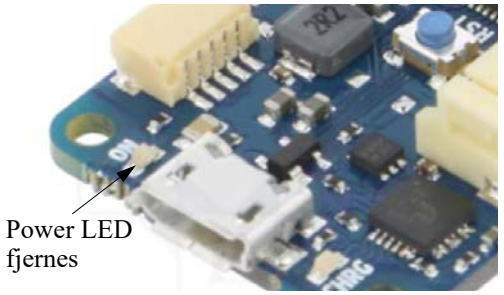
- Når man skal laste opp et nytt program til en mikrokontroller som legges i deepSleep så er sjansen for at man ikke får tilgang til porten stor. For å få lastet opp programmet, må man gi Resett på kortet to raske trykk. Dermed vil booteren være konstant tilgjengelig og man kan laste opp programmet. Pass på at mikrokontrolleren kan ha skiftet port.
- Når en MKR NB 1500 lagges i deepSleep slås nesten alle kretser av unntall 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.

3.4.3 Bruk av deepSleep (dvale) og 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[37] (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 via inntil tre porter.

Mikrokontrolleren tilføres strøm via USB-inngangen, hvilket er særdeles enkelt, men har noen ulemper. Power LED'en på kortet vil lyse og trekke strøm. Denne bør derfor fjernes (loddes ut) ved bruk av denne løsningen. Andre tilkoblede enheter, som f.eks. MKR ENV sensorkortet og MKR GPS kan legges i stand by av programmet når de ikke brukes. Backup batteriet til MKR GPS vil g jøre at oppstart av GPS går raskere.

For å få ned strømforbruket på mikrokontrollerenheten, legges denne i dvale (Deep sleep). En antar videre at det kun gjøres sjeldne målinger i korte perioder slik at det gjennomsnittlige strømforbruket holdes lavt.



NB! – Slår seg av

Et problem med flere Powerbanker 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 dersom sørge for å velge en Powerbank som ikke har denne funksjonen.

3.4.4 Bruk av dvale og Li-Po batteri

Denne løsningen ligner mye på den foran (avsnitt 3.4.3), 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:

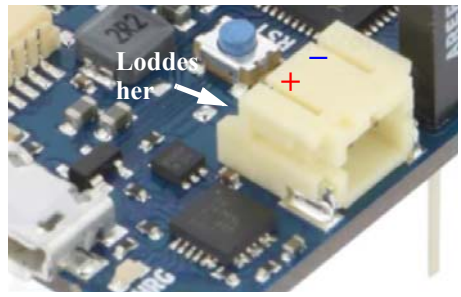
1. Man slipper å fjerne Power LED'en som ikke lyser når batteripluggen brukes.
2. Li-Po batteriet lades når man kobler USB-kabelen til mikrokontroller-kortet.

37.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[38]), 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 ev. 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 kretsen i dvale ev. 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.

3.4.5 Bruk av Li-ione batterier av typen 18650

Batterier av typen 18650 er ladbare batterier på 3,6 – 3,7V, men er noe større enn vanlige AA-batterier 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,-

38.https://no.rs-online.com/web/p/speciality-size-rechargeable-batteries/1449405

NB! En skulle tro at 18650 Li-ione 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). 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 utladning, som kan ha uheldige konsekvenser. Batteriet fra RS-online er av denne typen og får problemer med å passe inn i en standard batteriholder.

RS-online Stock No.: 880-1558

ELFA Distrelec leverer batteriholdere av ulik typer, de fleste er laget for seriekobling av batterier. Vi ønsker imidlertid å bruke holdere som parallellkobles eller 3D-printer egne holdere. Vi ønsker dessuten å bruke batterikontakter av typen 2 pin JST PHR2, og slike batteriholdere er ikke uten videre i salg, med mindre man kjøper slike kontakter i "løs vekt". Som nevnt er kontaktene av samme type som brukes til Micro:bit. Man kan evt. kjøpe hele batterikassetten til Micro:bit og klippe av pluggen. Kassetter av den typen kan kjøpes hos n00b.no for kr. 29,00 eller hos Kitronik for 0,75£ pr. stk. Eller man kan kjøpe en pakke plugger med kabel fra Banggood[39] for en billig penge. Her må man imidlertid være klar over at det er ingen standard for hva som skal være + og –. Det er derfor viktig å forsikre seg om at rød + og sort – er plassert på rett side i kontakten. 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.

Batteriholder for 18650 (ELFA)

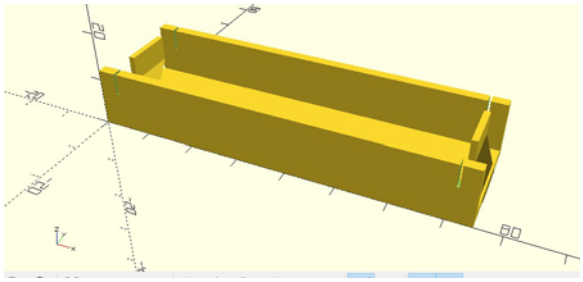
kr. 34,-

n00b.no/Kitronik
kr. 29,-/£0,75

kr. 60,-
for 100 stk
Banggood

39. 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&rmmds=search

En enkel holder kan også 3D-printes. Denne kan mangedobles slik at flere batterier av typen 18650 kan parallellkobles. Dermed beholder man spenningen, men øker kapasiteten.



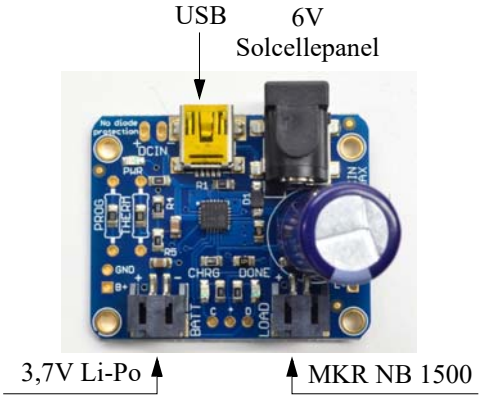
3.4.6 Bruk av dvale og solcelle ladet Power bank

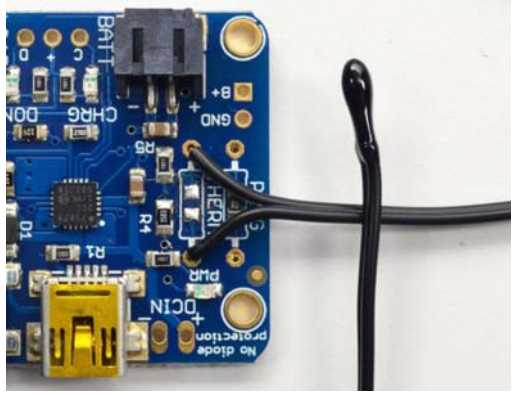
Som et alternativ til vanlig Power bank kan man benytte Power banker som lades med solceller som er montert på selve batteriet. Det finnes en variant med en kapasitet på 10 000 mAh, som påstås å være vanntett (IP66) som kan vurderes brukt. Den har dessuten 2 micro USB-plugger[40].

Det er imidlertid et åpent spørsmål hvor godt den vil egne seg for å ligge åpent på toppen av en havbøye. En kan ev. bygge den inn i en boks med et lokk som slipper gjennom sollys, dog med tap av energi gjennom plastlokket.

3.4.7 Bruk av dvale og solcelle ladet Li-Po batteri

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[41].





40.https://www.mytrendyphone.eu/shop/waterproof-solar-power-bank-dual-usb-10000mah-230881p.html
41.https://www.elfadistrelec.no/Web/Downloads/_t/ds/Adafruit_390_eng_tds.pdf

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 10kΩ som er montert på kortet der det står TERM, fjernes og erstattes med en NTC-motstand på 10kΩ 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[42], men det burde også gå an å lage et 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.

3.5 Bruk av RTC – DS3231 og Powerboost[43]

3.5.1 Introduksjon til Adafruit DS3231

DS3231 er en Real Time Clock (RTC) som kan brukes til å gi et nøyaktig tidsstempel på målingene. Dessuten kan den brukes til å vekke mikrokontrolleren med jevne mellomrom. Et batteri (CR1220) holder liv i kretsen slik at den tar vare på tiden ved strømbrudd.

Krystallen er integrert i kretsen sammen med en temperatursensor som gjør klokkefunksjonen spesielt stabil over tid.

Power-pinner:

Vin – Her tilføres spenningen fra 2,3 – 5,5V
GND – Felles jord for all logikk

42.https://www.adafruit.com/product/200
43.https://www.elfadistrelec.no/Web/Downloads/_t/ds/Adafruit_3013_eng_tds.pdf

I²C Buss-pinner:

SCL – I²C seriell klokkepinne, bruk gjerne en pull-up motstand på 10kΩ til V_{in}
SDA – I²C seriell datapinne, bruk gjerne en pull-up motstand på 10kΩ til V_{in}

Andre pinner:

BAT – På denne pinnen kan man hente ut batterispenningen fra CR1220 (3V).

32K–32KHz – Her kan man hente ut en nøyaktig klokkefrekvens på 32kHz.

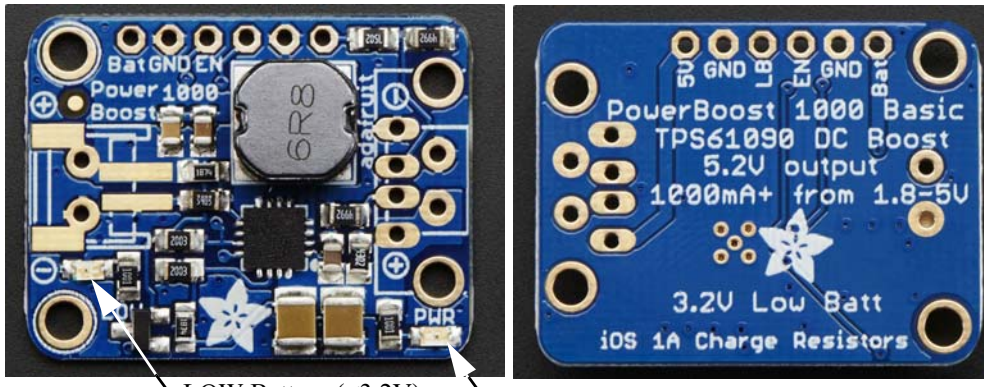
SQW – Alternativt firkantkurve eller utgang for interrupt.
Utgangen trenger en pull-up-motstand og vil gå lav når det innstilte tidspunktet for interrupt inntreffer.

RST – Utgangen er beregnet til å resette en ekstern krets i tillegg til at den resetter DS3231. Den kan også varsle om brudd på forsyningsspenningen.
En pull-up på 50kΩ holder utgangen høy så lenge kretsen har V_{in}.
Dersom V_{in} faller eller blir borte, kobles batteriet automatisk inn.

3.5.2 Introduksjon til Adafruit Powerboost 1000 Basic[44]

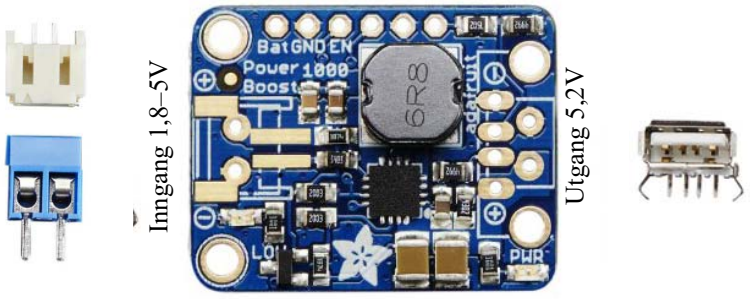
Adafruit Powerboost 1000 Basic er en DC/DC-konverter som gjerne henter en DC-spenning fra et batteri, gjerne et Li-Po batteri med spenning på 3,7V og løfter denne opp til en spenning på ca. 5V. Dette kan være hensiktsmessig for å kunne bruke standard Li-Po batterier til lading av f.eks. mobiltelefoner eller iPad. Li-Po batterer har høy kapasitet og kan levere store strømmer.

Figuren under viser over og undersiden av Powerboostern.



44.https://learn.adafruit.com/adafruit-powerboost-1000-basic/pinouts

Spenningen på inngangen kan være fra 1,8V til 5V og utgangsspenningen er ca. 5,2V. Kretsen leveres bl.a. av ELFA[45] 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

Det er power kontakter – 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 boosteren som dermed vil bli mer effektiv. Hold ledningene så korte som 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

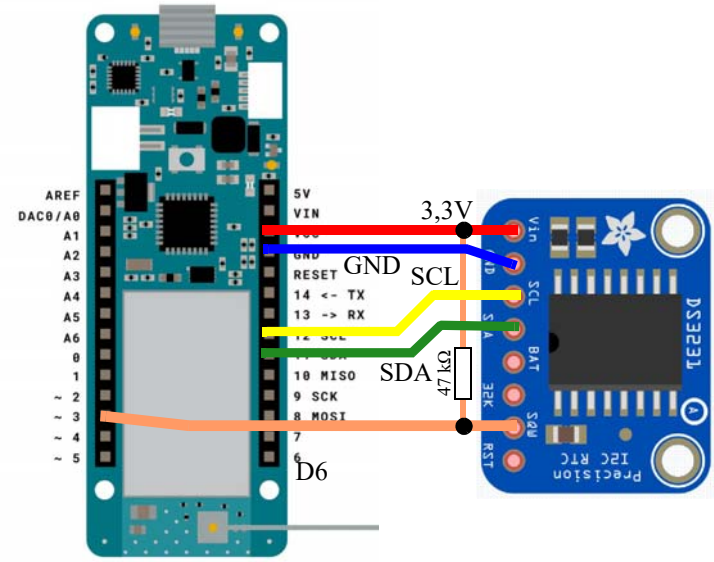
Kretsen har to kontroll-pinner:

- **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 "disablet".
- **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 mer enn 1A. En grønn LED er tilknyttet utgangen og lyser grønt når utgangen leverer strøm.

45.https://www.elfadistrelec.no/en/powerboost-1000-basic-adafruit-2030/p/30129192

3.5.3 Enkel oppkobling av DS3231 og MKR NB 1500

Figuren under viser en foreløpig oppkobling for uttesting av RTC'en DS3231.



3.5.4 Program for å sette klokken (*SetTimeDS3231Man-1.ino*)

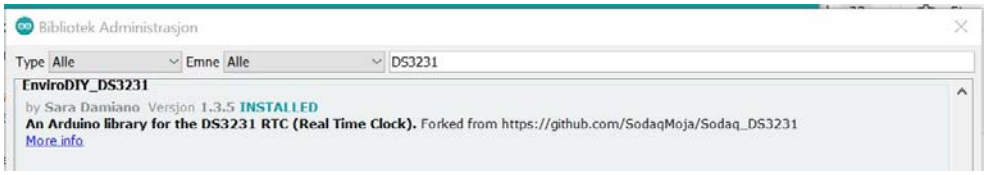
RTC leveres ikke med en klokke som går riktig fra leverandøren, vi må kjøre et program og aktivt sett den interne klokken. Dessuten bør vi sette inn et batteri (CR1220) som gjør at klokka fortsetter å gå selv om vi bryter strømmen.

Det er minst to måter å sette opp klokka i DS3231 på:

1. Enten kan vi angi tidspunktet for klokka manuelt ved å skrive inn riktig tidspunkt og så laste opp og kjøre programmet rett før dette tidspunktet inntreffer slik at den går mest mulig riktig. Med denne metoden må man regne med en usikkerhet på noen sekunder.
2. Alternativt kan man bruke et program som setter klokka i RTC'en lik PC-klokka. Denne metoden vil gi en klokke som er stilt omtrent lik klokka i PC'en som kompilerte programmet.

Her har vi valgt manuell setting av klokka (variant 1).

1. **Installasjon av biblioteket**
   Hent biblioteket til DS3231 fra https://github.com/EnviroDIY/Sodaq_DS3231 og installer på vanlig måte. Evt. bruk Library Manager:



2. **Hent opp eksempelprogrammet**
   Under File/Examples finner man katalogen *EnviroDIY_DS3231* og henter opp filen *adjust.ino*. En fornorsket utgave av den samme filen finner man i vedlegg E.4.1 side 228 under navnet *SetTimeDS3231Man-1.ino*.

3. **Skriv inn tiden**
   Så skriver man inn år dato og tid, i tillegg til ukedag i argumentet på funksjonen DateTime():

```
//År, Måned, Dato, Time, Minutt, Sekund og ukedag (går fra 0 til 6)
 DateTime dt(2022, 07, 31, 20, 33, 45, 0);
```

   Ukedagene går fra Søndag (0) – Lørdag (6).

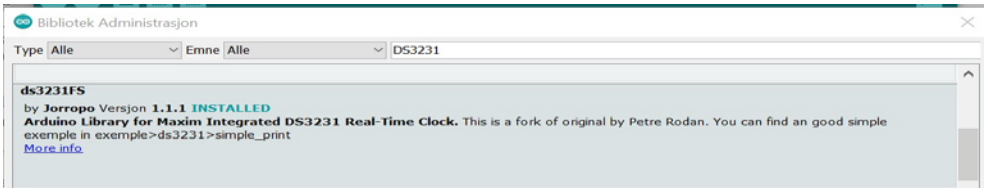
4. **Kjør programmet**
   Kompiler, last opp og kjør programmet. Programmet leser også av DS3231 og skriver dato og tid i monitoren.

Har man installert et batteri av typen CR1220 i DS3231, så vil denne beholde klokkeinformasjon selv om strømmen slås av og man slipper å stille klokka på nytt.

Henting av dato og tid (*simple_print.ino*)

For det videre arbeidet har vi valgt å bruk en annen variant av programmet. Her står man fritt til å velge. Det finnes en rekke ulike programmer for bruk av DS3231, men ikke alle lar seg bruke sammen med MKR NB 1500. Her får man bare prøve seg fram for å finne ut hva som fungerer. Vi har valgt et bibliotek som tilbyr en så enkel eksempel-fil som mulig og som lar seg bruke sammen med MKR NB 1500.

1. **Installasjon av biblioteket**
   Hent og installer biblioteket DS3231FS ved hjelp av Library Manager:



2. **Hent opp eksempelprogrammet**
   Under File/Examples finner man katalogen DS3231FS og henter opp filen *simple_print.ino.*

3. **Kjør programmet**
   Kompiler og last opp programmet. Programmet leser av tiden fra DS3231 og skriver resultatet til monitoren. Juster utskriften slik at den blir i henhold til norsk standard

Dersom man ønsker å ha et tidsstempel på målingene sine så kan man bruk denne for å lese av DS3231 og legge tidspunktet inn i resultatfila.

Oppsett av enkle alarmer (*setIntAlarm.ino*)

La oss se hvordan vi kan sette opp en enkel alarm. Vi tar utgangspunkt i biblioteket DS3231FS. Programmet *setIntAlarm.ino* setter opp *en* alarm eller gjentatte alarmer ved gitte tidspunkter, skriver ut tidspunkter hvert sekund og varsler i monitoren når alarmen inntreffer. I tillegg legges pinne SQW hos DS3231 lav idet alarmen går.

1. **Hent opp programmet**
   Programmet er vedlagt i vedlegg E.4.2 side 229 og bygger på eksempelprogrammer fra DS3231FS.

2. **Sett tidspunkt for vekking**
   Man kan sette opp en vekketid med time, minutt og sekund. Vekking vil da kunne skje daglig på dette tidspunktet.

   ```
   // Tid for oppvekking
   uint8_t wake_HOUR = 15;
   uint8_t wake_MINUTE = 49;
   uint8_t wake_SECOND = 59;
   ```

   uint8_t angir bruk av en enkelt byte (8 bit) som sparer en byte i forhold til “int”.

3. **Initier alarm**
   Dette kan bl.a. gjøres i setup()-funksjonen

   ```
   DS3231_init(DS3231_INTCN); // Initialiser DS3231
   DS3231_clear_a1f(); // Resett alarm
   set_alarm(); // Sett opp alarmtidspunkt
   ```

   Først initialiseres DS3231 hvor DS3231_INTCN angi I^2C adressen til DS3231. Dernest resettes evt. alarmer, hvilket bl.a. betyr at SQW legges høy. Så settes alarmen opp med ønsket tidspunkt. set_alarm() er en funksjon som vi ganske snart skal komme tilbake til.

4. **Vent på alarm**
   Vi lar programmet gå i loop()-funksjonen mens vi venter på at alarmen skal utløses.

   ```
   void loop()
   {
     printTime(); // Skriv ut tidspunktet
     delay(1000);
     if (digitalRead(pinRTCINT) == 0) // Sjekker at SQW er gått lav
       {
        Serial.println("Alarmen er utløst");
        delay(5000);
        DS3231_clear_a1f(); // Resett interrupt-linjen (high-z)
       }
   }
   ```

   Loop()-funksjonen skriver ut tidspunktet hvert sekund. For å detektere alarm har vi koblet SQW utgangen hos DS3231 til D3 på MKR NB 1500 og kalt denne `pinRTCINT`. Når denne går lav har alarmen gått og det varsles om det i monitoren. Vi venter 5 sek. før vi resetter SQW med kommandoen `DS3231_clear_alf();`

5. **Sett alarmtidspunktet**
   Vi har laget en liten funksjon som setter alarmtidspunktet – `void set_alarm(void)`. Dette har vi gjort for at vi senere lett skal kunne sette opp vilkårlige måleintervaller. F.sks. hvert 10 sekund eller hvert 15 minutt o.l.

```
void set_alarm(void)
{
  // flags[] angir hvilke komponenter i dato og tid som skal sjekkes
     for å gi alarm
  // A1M1 (seconds) (0 to enable, 1 to disable)
  // A1M2 (minutes) (0 to enable, 1 to disable)
  // A1M3 (hour)    (0 to enable, 1 to disable)
  // A1M4 (day)     (0 to enable, 1 to disable)
  // DY/DT          (dayofweek == 1/dayofmonth == 0)
  //Sjekker {sek, min, timer, dag, dag i uke/måned}
  uint8_t flags[5] = { 0, 1, 1, 1, 1 };

  // Sett Alarm1
  DS3231_set_a1(wake_SECOND, wake_MINUTE, wake_HOUR, 0, flags);

  // Aktiver Alarm1
  DS3231_set_creg(DS3231_INTCN | DS3231_A1IE);
}
```

   Først settes det opp et array med flagg som angir hvilke av parametrene sekunder, minutter, timer, dager og uker/måneder som skal tas med for å sjekke om alarmen er gått. Settes et flagg lik 0 i posisjonen til sekunder (som i eksempelet over), så skal bare sekunder sjekkes for match og ikke de øvrige parametrene. I vårt tilfelle betyr dette at det skal gå en alarm hver gang sekunder passerer 59, dvs. en gang i minuttet. Tilsvarende kan de øvrige flaggene settes til null der man ønsker at denne tidsangivelsen skal være med å bestemme match. Flere flagg kan settes til "0" samtidig slik at match må oppfylles for flere parametere.

   Dernest sendes informasjon over til DS3231 og alarmen er satt til de angitte betingelsene. Til sist må alarmen aktiveres slik at den faktisk kommer i funksjon.

6. **Kjør programmet**
   Kjør programmet og observer hvordan det fungerer.

Oppsett av alarm med tilfeldige intervaller (*setIntAlarm-1.ino*)

Det vil være aktuelt å sette opp alarmer av typen hvert 10 sek. eller hvert 15 minutt eller hver andre time. For å få til dette må vi fortløpende endre verdien på neste alarm. Dette er gjort i eksempelkoden i vedlegg E.4.3 side 232.

1. **Definer variabler**
   Her har vi valgt å definere noen ekstra variabler som holder de ønskede intervallene for sekunder, minutter og timer.

```
// Tid for oppvekking
uint8_t wake_HOUR = 0;
uint8_t wake_MINUTE = 0;
uint8_t wake_SECOND = 0;
uint8_t wake_interval_HOUR = 0;
uint8_t wake_interval_MINUTE = 0;
uint8_t wake_interval_SECOND = 10;
```

I dette eksempelet har vi satt opp et intervall mellom hver måling på 10 sekunder. 10 sekunder er et svært kort, men greit for uttesting.

2. **Endre alarmtidspunktet fortløpende**
   I `set_alarm(void)` funksjonen legger vi inn en oppdatering av alarmtidspunktet, slik funksjonen håndterer passering av hele minutter, timer og døgn.

```
// Beregne neste tidspunkt

wake_SECOND = wake_SECOND + wake_interval_SECOND;
if(wake_SECOND >= 60) {wake_SECOND = wake_SECOND - 60;}

wake_MINUTE = wake_MINUTE + wake_interval_MINUTE;
if(wake_MINUTE >= 60) {wake_MINUTE = wake_MINUTE - 60;}

wake_HOUR   = wake_HOUR   + wake_interval_HOUR;
if(wake_HOUR   >= 24) {wake_HOUR   = wake_HOUR   - 24;}
```

   Som vi ser så legges intervallet fortløpende til den eksisterende alarmtiden (vekketiden) i tillegg til at det sjekkes om tiden passerer hele minutter, hele timer eller hele døgn.

   Studer forøvrig det komplette programmet for å se hele sammenhengen.

3. **Kjør programmet**
   Hent programmet og se at det fungerer som forventet.

Vi ønsker dernest å undersøke om vi kan spare ytterlige energi ved å bruke DS3231 til helt å slå av MKR NB 1500 inkludert tilleggskort, mellom målingene.

3.5.5 Bruk RTC og "power booster" til å slå av mikrokontrolleren

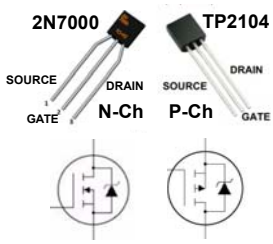
For ytterligere å spare energi kan vi slå av mikrokontrolleren og alle tilleggskortene helt og kun la DS3231 gå. DS32331 trekker mindre enn 1mA.

Oppkobling

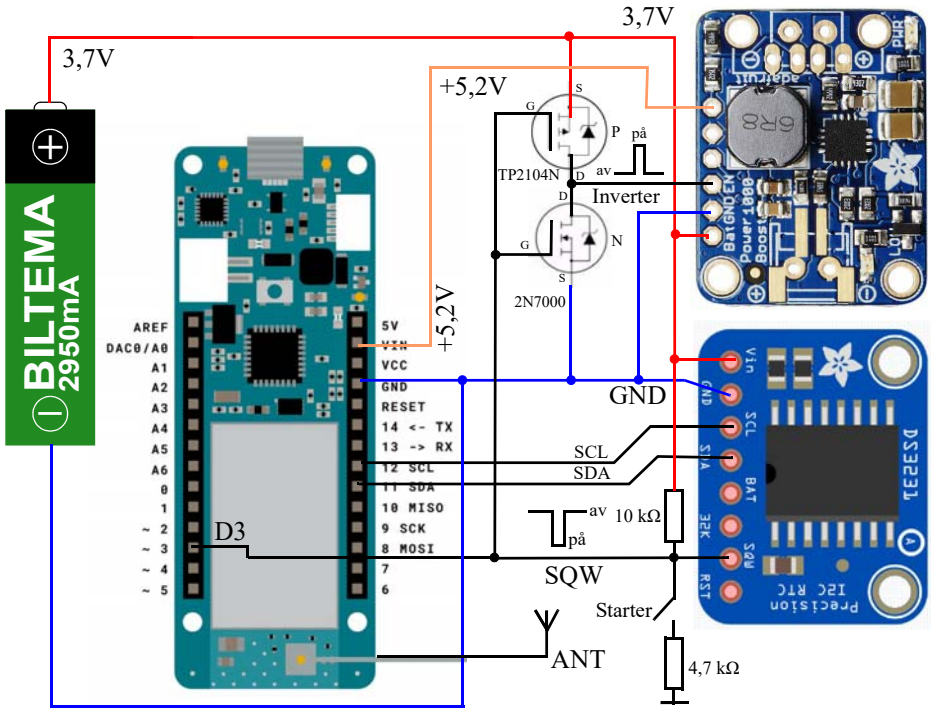
Det er rapportert at bruk av batteriinngangen direkte kan gjøre kretsen noe ustabil, spesielt når batteriet lades ut. Det er derfor anbefalt å bruke V_{in} eller USB-inngangen. For likevel å kunne bruke et standard Li-Po-batteri f.eks. av typen 18650 er det anbefalt[46] å anvende en DC-DC-konverter for å løfte spenningen opp til 5V f.eks. Adafruit 2030 – PowerBoost 1000 Basic[47]. Selv om spenningen på batteriet faller noe, så vil denne løfte spenningen opp til 5,2V. En annen fordel er at den

46. Helge Rustad, SINTEF
47. https://www.elfadistrelec.no/en/powerboost-1000-basic-adafruit-2030/p/30129192

har en enable inngang som gjør det mulig å slå av spenningen på utgangen ved hjelp av Interrupt signalet (SQW) fra DS3231. Siden interruptsignalet er aktivt lavt og enable signalet er aktivt høyt, må vi sette inn en CMOS-inverter som snur polariteten. CMOS for å spare energi når kretsen er i dvale. Dette løses enkelt med et MOSFET P/N-kanal transistorpar.



Siden vi skal forsyne MKR-kretsen med 5V, kan vi vurdere å lodde ut LED'en som indikerer at kretsen har strøm, da denne alene vil trekke typisk 10mA. Dog betyr det ikke så mye siden den ikke lyser når kretsen slås av på denne måten. DS3231 må ha spenning hele tiden så denne velger vi å forsyne direkte fra Li-Po batteriet. Denne er heller ikke så følsom for dropp i batterispenningen samtidig som den har et eget backup batteri. Dermed ender vi opp med følgende forslag til oppkobling.



Strømforbruk

La oss måle strømforbruket for kretsen vi har bygget mens vi kjører testprogrammet. Testprogrammet slår av mesteparten av kretsen når den legges i dvale, kun RTC-kretsen og inverteren har spenning. Sistnevnte skal i teorien ikke trekke strøm i hele tatt, kun når den skifter tilstand.

Det kan være fristende å øke pullup motstanden fra SQW-utgangen. Øker vi denne fra 10kΩ til 47kΩ medfører det at pulsen ikke oppnår de nivåene den skal og kretsen går istå. Det viser seg dessuten at det betyr lite for strømforbruket.

Når kretsene er operative er det flere lysdioder som trekker strøm. Det gjelder power boosteren og mikrokontrolleren som begge har en lysdiode som indikerer at de har spenning. Boosteren har i tillegg en rød diode som indikerer at batteriet leverer strøm. I tillegg har vi benyttet den innebygde lysdioden hos mikrokontrolleren til å markere at den befinner seg i setup()- og loop()-funksjonen, dessuten lyser ladeindikatoren hos mikrokontrolleren opp ved oppstart. Strømtrekket er som følger:

- Ved oppstart: 67 mA
- Ved avslutning: 45 mA (Innebygget og ladeindikator er avslått)
- Ved avslått, dvale: 1,2mA

Programvare

Når vi kobler mikrokontrolleren fra PC'en fjerner vi all skriving til monitoren via USB-kabelen (Serial.print). I stedet bruker vi den innebygde lysdioden til å gi signal slik at vi vet hva som skjer i kontrolleren. Test programmet finnes i sin helhet i vedlegg E.4.4 side 234.

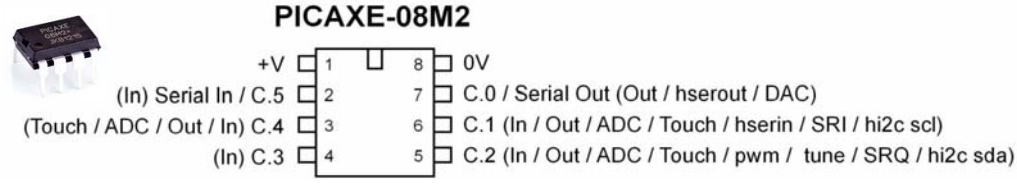
3.5.6 Watchdog

En watchdog ("vakthund") er en krets, gjerne utenfor 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 vakthunden hjelpe mikrokontrolleren ut av knipen.

Dette kan gjøres på flere måter, men en måte kan være at en timer hos "vakthunden" 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. I dette tilfellet betyr det at vakthunden legger resett-inngangen lav i et 1/2 sekund.

En slik løsning er aktuell både når kretsen er operativ under hele måleserien og når den legges i dvale.

Den enkleste løsningen synes pr. nå å benytte en liten 8 pins mikrokontroller som skreddersys for denne jobben. Av praktiske hensyn foreslås benyttet en PIC-prosessor av typen PICAXE-08M2+[48]. Den er billig (2£), kan arbeide på 3,3V og trenger ingen tilleggskomponenter. En trenger imidlertid enkelt verktøy for å programmere kretsen[49]. Microkontrolleren kjøpes fra samme nettsted[50]. Programvaren kan lastes ned fra nett[51].



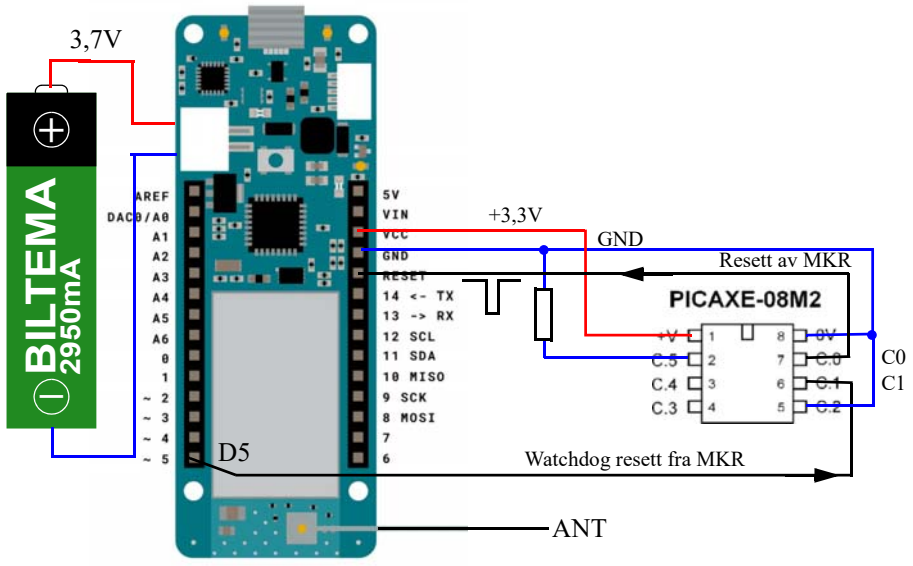
48. https://picaxe.com/hardware/picaxe-chips/picaxe-08m2-microcontroller/
49. Dette enkle verktøyet kan brukes for å programmere kretsen: https://picaxe.com/hardware/project-boards/picaxe-08-proto-board/
50. https://picaxe.com/hardware/picaxe-chips/picaxe-08m2-microcontroller/

Strømtrekk

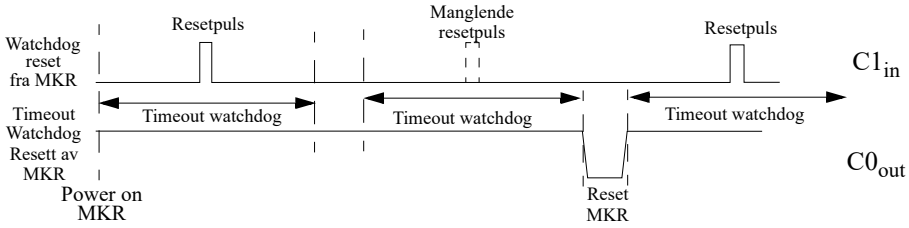
Strømtrekket til mikrokontrolleren PICAXE-08M2+ er målt til typisk 580µA.

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:



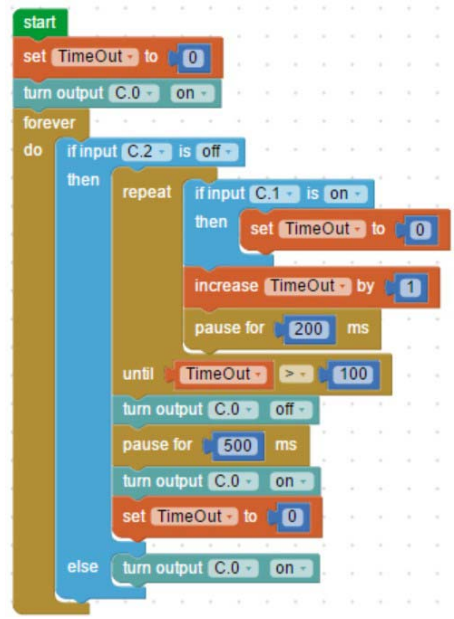
"Timeout watchdog" er tiden vakthunden venter før den går til aksjon og resetter MKR mikrokontrolleren. Normalt vil vakthunden bli resatt av resett-pulsen før timeout inntreffer. Det er viktig å huske på at første oppkobling etter resett kan ta noe lengre tid enn normalt, dette gjelder spesielt GPS-mottakeren som må låses til satellittene. Imidlertid burde dette gå raskt da GPS-kretsen har backup batteri som husker parametrene. Time out-tiden må tilpasses målesekvensen som velges.

51. Last ned og installer programvare: https://picaxe.com/software/ Velg: PICAXE Editor 6

Det viser seg at programmet ikke vil starte etter hard resett av mikrokontrolleren dersom det ligger kommandoer som kommuniserer på serielinjen. Enten må disse fjernes eller så må man lukke og åpne monitoren på nytt for at programmet skal gå videre.

Programvare og uttesting av vakthund

Figuren til høyre viser programmet som er lastet opp i "vakthunden". C0 – - C2 er porter på mikrokontrolleren:



- C0 – er en utgang som resetter MKR-kortet og starter programmet på nytt. C0 er normalt høy, men går lav i 500ms ved resett.
- C1 – er en inngang som "lytter" etter om SQW resett signalet. Når dette kommer, settes TimeOut-variabelen til null.
- C2 – er en inngang som sjekker om SQW-signalet er gått lavt. Om så er tilfelle starter TimeOut telleren å gå. Så lenge den er høy vil TimeOut telleren settes til "0".

Bruk av "vakthund" når MKR legges i dvale

Dersom vi ønsker å bruke dvale for å redusere strømforbruket, er det likevel aktuelt å ha en "vakthund" som evt. kan restarte mikrokontrolleren dersom den skulle stoppe midt i programmet og dermed hindre at mikrokontrolleren går inn i dvale.

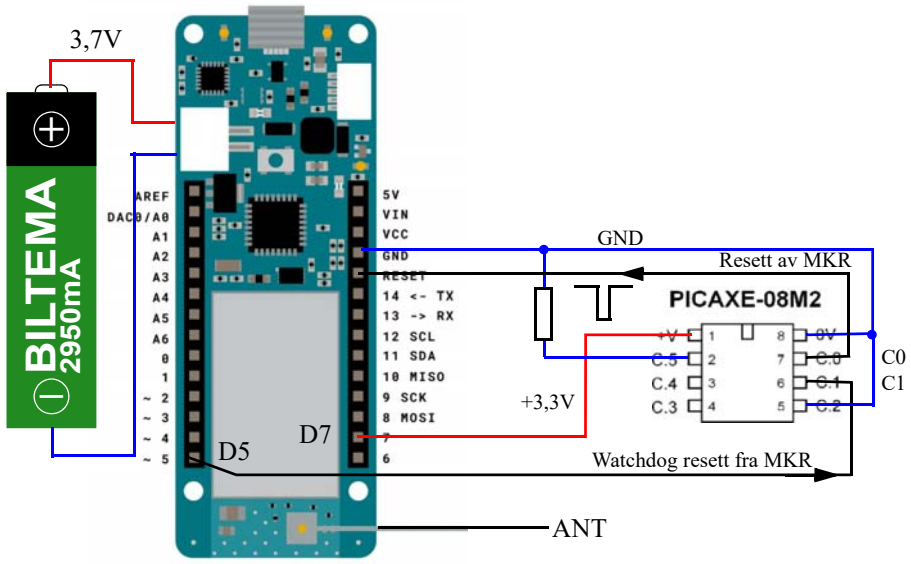
Det ideelle hadde vært om "vakthunden" kunne vært gitt spenning fra V_{CC} hos mikrokontrolleren og at denne hadde blitt slått av under dvalen, noe som ikke skjer.

1. Et alternativ er at "vakthunden forsynes med en utgang fra mikrokontrolleren Utgangene kan levere inntil 7 mA. Dette burde la seg gjøre siden "vakthunden" er svært energigjerrig (målt strømtrekk <1 mA). Dette forutsetter imidlertid at mikrokontrolleren er så oppegående at den kan forsyne "vakthunden" med spenning, selv om programmet stopper.
2. Et annet alternativ er at "vakthunden" har spenning hele tiden siden den trekker så lite strøm. I så fall må man sørge for at pulsene den leverer ikke forstyrrer kretsen i dvale og at den restarter når mikrokontrolleren går ut av dvale.

Inntil videre foreslår vi at alternativ 1 brukes.

En kan tenke seg en oppkobling som vist på figuren under:

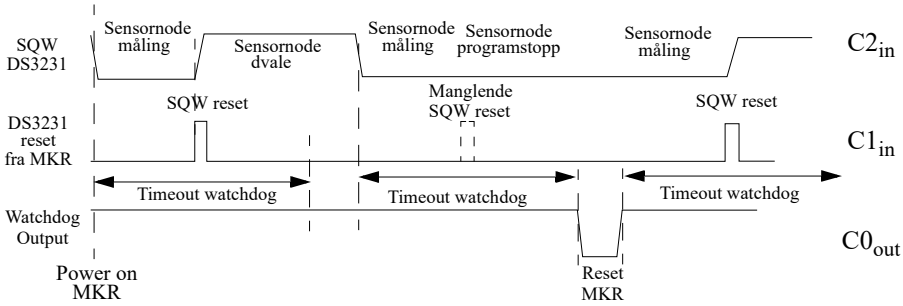
Her har vi valgt å forsyne "vakthunden" fra port 6 hos MKR NB 1500. Programmet setter port 6 høy i setup()-funksjonen og slår den av rett før MKR legges i dyp søvn før neste måling. Det er viktig at resett av MKR holdes høy selv om spenningen på "vakthunden" er avslått:



Powerbooster med "vakthund"

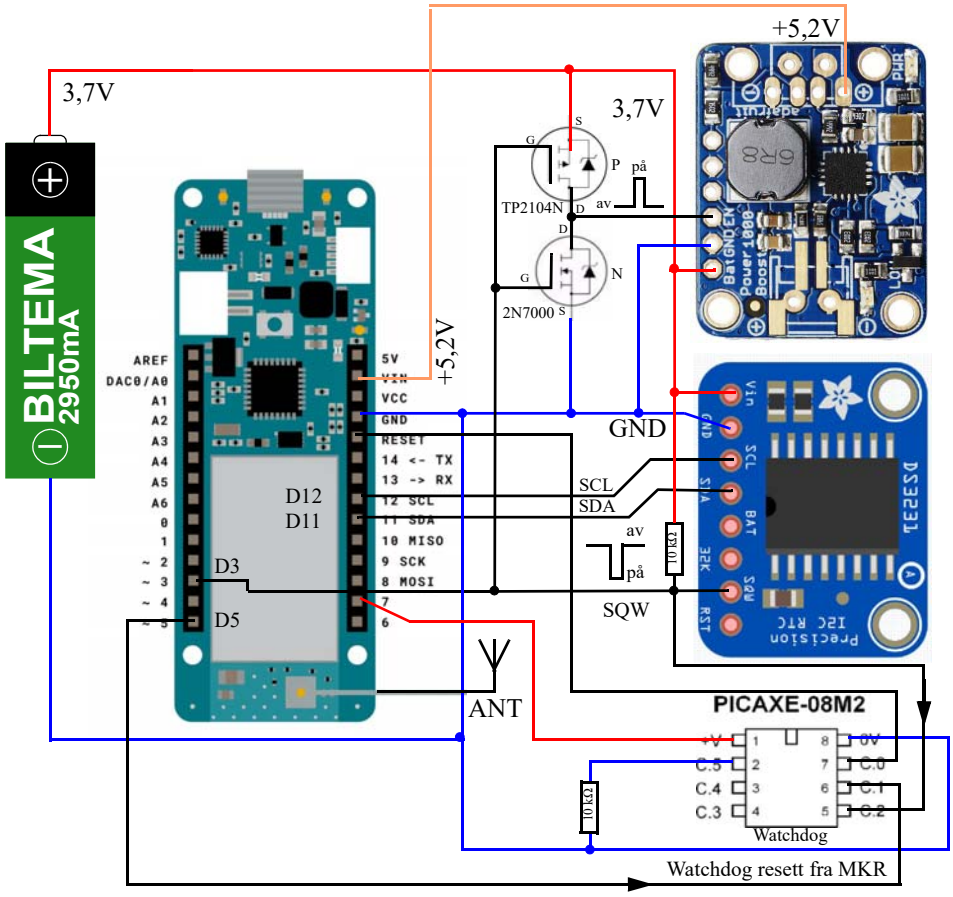
Vi skal nå se på hvordan vi kan inkludere "vakthunden" sammen med powerbooster'en.

Et ønsket forløp kan se ut som vist på figuren under:



Når strømmen slås på får både Powerbooster'en, MKR med tilleggskort og "vakthunden" spenning. Dermed starter timeren i "vakthunden" som har en relativ romslig timeout (Timeout watchdog) slik at under normale forhold vil alle målinger være avsluttet og MKR rekker å sende et resett-signal (SQW resett) som restarter timeren. Timeout kan gjerne være på ett minutt eller mer. Dersom resett-signalet uteblir vil "vakthunden" gå til timeout og sende et resett signal til MKR (Resett MKR) som starter programmet på nytt. Vi forutsetter at utgangen som gir "vakthunden" spenning ikke går ned når programmet låser seg.

Oppkoblingen kan se ut som vist på figuren under. Merk: Denne oppkoblingen er ikke testet ut, så bruk den med omtanke:



4 Oppgavesamling – grunnopplæring MKR NB 1500

I denne delen skal vi ta for oss de enkelte delprosjektene og bygge opp systemet bit for bit.

Som nevnt tar vi sikte på å bygge opp kompetansen gradvis gjennom å arbeide med små maskin- og programvaremoduler. Det endelige produktet (oppdraget) kan beskrives slik:

> **Oppdraget:** *Det skal monteres et mikrokontrollerkort for måling av miljødata (lufttrykk, luftfuktighet, vann- og lufttemperatur og lysstyrke). I tillegg skal kortet utstyres med en GPS-mottaker slik at målingene kan tid og stedfestes. Dataene skal overføres til en server hvor måleserien skal legges i en fil. De lagrede dataene skal de presenteres som tabeller og grafer, gjerne med bruk av Excel og Google Earth, eller annen passende programvare.*

Oversikt over deloppdrag

I løpet av 10–11 deloppdrag skal vi bygge opp og overføre data fra sensornoden vår til serveren og presentere dataene.

> **Deloppdrag 1:** *Installer programeditor for Arduino og tilleggsmoduler for programmering av MKR NB 1500.*
>
> **Deloppdrag 2:** *Koble opp hårdvaren, monter antennen og installere SIM-kortet*
>
> **Deloppdrag 3:** *Installer og kjør programvare for å lese av IMSI og IMEI numrene til mikrokontrollerkortet og SIM-kortet. Dette er strengt tatt ikke nødvendig da de ikke trengs for å opprette sambandet*
>
> **Deloppdrag 4:** *Installer bibliotekx og testprogrammet for å koble opp sensornoden til 4G-nettverket og overføre data til serveren. Last opp testprogrammet for overføring av dummy-data til serveren. Gå inn på serveren og kontroller at lagrede data er som forventet*
>
> **Deloppdrag 5:** *Monter MKR ENV-kortet og installer det tilhørende biblioteket. Last opp testprogrammet for testing av ENV-kortet. Inkluder avlesning av ENV-kortet i hovedprogrammet slik at de avleste dataene overføres til serveren. Kontroller at lagrede data er som forventet*
>
> **Deloppdrag 6:** *Monter MKR GPS-kortet og installer det tilhørende biblioteket. Last opp testprogrammet for testing av GPS-kortet. Inkluder avlesning av GPS-kortet i hovedprogrammet slik at de avleste dataene overføres til serveren. Kontroller at lagrede data er som forventet*
>
> **Deloppdrag 7:** *Kjør mikrokontrollerkortet alene sammen med sensorkortet for måling av vanntemperaturmåleren. Stikk sensoren ned i et glass med vann og undersøk om temperaturen er som forventet. Dernest skal avlesningen av temperaturen legges inn i strengen i hovedprogrammet og sendes over til serveren*
>
> **Deloppdrag 8:** *Inkluder måleresultatene fra vanntemperatursensoren i hovedprogrammet og send samtlige målinger til serveren. Kontroller at lagrede data er som forventet*
>
> **Deloppdrag 9:** *Presenter måleresultatene med passende programvare, f.eks. Excel og Google Earth. Kommenter resultatene*

Deloppdrag 10: *Kjør mikrokontrollerkortet alene og undersøk strømforbruket når kortet legges i dvale. Vekk kortet etter en stund og mål strømforbruket ved vanlig drift. Koble til aktuelle sensorkort og la mikrokontrolleren legge disse i stand by Mål strømforbruket når hele sensornoden er lagt i dvale og når den er i normal drift. Estimer forventet driftstid på et valgt batteri.*

4.1 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 MKR NB 1500.*

1. *Installasjon av Arduino IDE*
   Beskrivelsen av installasjonen finner du i delkapittel avsnitt 5.1 på side 83
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 5.2 på side 85.
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 bibliotekene finner du i delkapittel avsnitt 5.3 på side 87.

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

4.2 Deloppdrag 2: Bestilling og oppkobling av hårdvare og montering av SIM-kort

I dette deloppdraget skal vi bestille og montere hårdvare og SIM-kort.

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

1. *Anskaffelse av hårdvare*
   Om hardvaren ikke er bestilt så er dette tidspunktet for å gjøre det. Velger man å bruke Arduino så kan MKR NB 1500 være et aktuelt valg. Til denne kreves også en antenne som monteres til mikrokontroll pass på å få riktig antennekontakt. I tillegg kan følgende kort være greie å ha: MKR ENV (Miljøkort) og MKR GPS (GPS-kort) med batteri. Et prototypkort kan også være godt å ha dersom man ønsker å eksperimentere med egne sensorer. Vi har valgt å bruke et stort protypkort der vi har installert sensoren DS18B20 for måling av vanntemperatur, se avsnitt 7.1.3 på side 122. Se forøvrig også bestillingsliste i Vedlegg A side 170. Mikrokontrollerkortet og shield-kortene er beskrevet i avsnittene 3.1 side 31, 3.2.2 side 35, 3.2.2 side 35 og 3.2.6 side 51.
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 det i posten. Du må også bestemme deg for hvilken IoT-teknologi du trenger, LTE-M eller NB-IoT. Se avsnitt 5.4 side 87. Her er det flere muligheter avhengig av om man ønsker noen

gratis SIM-kort for utprøving, ordinært innkjøp av inntil 9 SIM-kort eller opprettelse av bedriftsavtale for mer enn 9 SIM-kort. Pass på å merke av at kortene ikke krever inntasting av PIN-kode da dette er vanskelig for vårt bruk.

3. *Monter antennen og SIM-kort*
   Antennen og SIM-kortet monteres på mikrokontrollerkortet MKR NB 1500 som omtalt i avsnitt 5.5 side 90. Vær forsiktig når du monterer antennen da antennekontakten er liten og skjør.

4.3 Deloppdrag 3: Finn ID-numrene for SIM-kort og hårdvare

I dette deloppdraget skal vi lese av numrene IMSI og IMEI som identifiserer SIM-kortet og mikrokontrollerkortet. Dette er det strengt tatt ikke nødvendig å vite, men det kan være praktisk å vite hvordan man kan skaffe seg denne informasjonen.

Deloppdrag 3: *Installer og kjør programvare for å lese av IMSI og IMEI numrene til mikrokontrollerkortet og SIM-kortet. Dette er strengt tatt ikke nødvendig da de ikke trengs for å opprette sambandet*

1. *Installer og kjør program for avlesning*
   For å kunne lese av IMSI og IMEI-numrene må vi installere et avlesningsprogram som når det kjøres, skriver ut numrene i monitoren. Disse skal vi senere bruke når vi oppretter paneler for visning av måledata. Se avsnitt 5.6 side 90.

2. *IMSI og IMEI numrene*
   Les av numrene og ta vare på dem til ev. senere bruk.

4.4 Deloppdrag 4: Installer bibliotek og last opp og kjør testprogram for overføring av dummy-data

Her skal vi forberede for overføring av dummy-data, det vil si data som genereres av programvaren og ikke av sensorer.

Deloppdrag 4: *Installer bibliotek og testprogrammet for å koble opp sensornoden til 4G-nettverket og overføre data til serveren. Last opp testprogrammet for overføring av dummy-data til serveren. Gå inn på serveren og kontroller at lagrede data er som forventet*

1. *Hent og installer bibliotek og **enkelt testprogram*** `(Eksempel-kode-1.ino)`
   Hent og installer biblioteket for å kunne bruke NB IoT teknologien. Dette er beskrevet i avsnitt 5.7.2 side 93.

2. *Studer koden hos **enkelt testprogram***
   Last opp et enkelt testprogram for testing av oppkobling til 4G-nettverket og til serveren. Programmet i vedlegg E.3.1 side 221. Gå gjennom programmet og forsøke å få oversikt over de ulike delene av programmet. Studer spesielt den delen som etablerer kontakt via radiokanalen (se avsnitt 5.7.2 side 93) og der hvor tekststrengen blir bygget opp og sendt til databasen (se avsnitt 5.7.3 side 95). Velg et unikt filnavn (<filnavn>.txt) hvor dataene skal lagres på serveren.

3. *Kompiler og kjør **enkelt testprogrammet***
   Når testprogrammet kjører vil det sende ut et par tallverdier som legges i filen hos serveren.
4. *Sjekk data på serveren*
   Gå inn på følgende nettadresse: `https://sensor.marin.ntnu.no/logs/` og sjekk at de forventede dataene ligger i filen med navnet du har valgt.
5. *Last opp, kompiler og kjør **komplett testprogram** med dummy-data* `(Eksempel-kode-3.ino)`
   Et mer komplett program for overføring av data er beskrevet i avsnitt 5.7.4 side 96 og finnes i E.3.2 side 223. Dette programmet er ment å være grunnlaget for oppbygging av det endelige programmet for overføring av sensordata.

4.5 Deloppdrag 5: Monter miljøkortet og overfør reelle måledata

Dersom vi nå har testet forbindelsen og ser at data kommer inn, så kan vi montere MKR ENV-kortet for om mulig å overføre noen reelle målinger.

Deloppdrag 5: *Monter MKR ENV-kortet og installer det tilhørende biblioteket. Last opp testprogrammet for testing av ENV-kortet. Inkluder avlesning av ENV-kortet i hovedprogrammet slik at de avleste dataene overføres til serveren. Kontroller at lagrede data er som forventet*

1. *Hent og installer MKR ENV-biblioteket*
   Hent ned og installer MKR ENV-biblioteket i Arduino editoren. Se avsnitt 3.2.2 side 35.
2. *Monter kortet MKR ENV*
   Monter kortet på "ryggen" av mikrokontrollerkortet – MKR NB 1500. Pass på at de ulike pinnene kommer i riktig hylsekontakt. Like betegnelser på hylsekontaktene skal plasseres rett over hverandre.
3. *Hent og kjør MKR ENV-testprogram* `(Testprogram-MKR-ENV-1.ino)`
   Med biblioteket følger også eksempelkode som bl.a. inneholder et testprogram. Bruk testprogrammet fra vedlegg E.2.1 side 204. Dette testprogrammet henter sensordata fra ENV-kortet og skriver dem ut i monitoren. Programmet sender ikke dataene til IoT-skyen. Det er derfor kun en test for å sjekke sensorene.
4. *Inkluder ENV-testprogrammet i hovedprogrammet*
   Vi skal nå bygge ut hovedprogrammet som samler inn måledata og sender disse til serveren. Dette gjør vi ved å inkludere nødvendige kommandoer fra eksempelprogrammet (vedlegg E.2.1 side 204) til testprogrammet for overføring av dummy-verdier (vedlegg E.3.2 side 223) som nå vil utgjøre det framtidige hovedprogrammet vårt. Pass på å lagre testprogrammet under et nytt navn.
5. *Test og kjør hovedprogrammet*
   Test det nyutviklede hovedprogrammet og sjekke at måledata kommer inn og finnes igjen på serveren. Alternativt kan man studere løsningsforslaget `(Sea-buoy-2)` som er lagt ved i vedlegg G.1 side 250 og bruke dette for å teste forbindelsen.

Vi er nå klare til også å inkludere GPS-målinger.

4.6 Deloppdrag 6: Monter GPS-kortet og inkluder GPS data

Dersom vi nå har testet forbindelsen og ser at måledata kommer inn fra MKR ENV-kortet, er vi klare til å gjøre det samme med MKR GPS-kortet.

> **Deloppdrag 6:** *Monter MKR GPS-kortet og installer det tilhørende biblioteket. Last opp testprogrammet for testing av GPS-kortet. Inkluder avlesning av GPS-kortet i hovedprogrammet slik at de avleste dataene overføres til serveren. Kontroller at lagrede data er som forventet*

1. *Hent og installer MKR GPS-biblioteket*
   Hent ned og installer MKR GPS-biblioteket i Arduino editoren. Se avsnitt 3.2.2 side 35.
2. *Monter GPS-kortet*
   Monter GPS-kortet ved hjelp av kabelen som kommuniserer med microkontrolleren via I^2C-bussen. Dersom GPS-kortet plugges ned som et shield-kort så benyttes Rx/Tx (UART) for å overføre data, hvilket betyr at testprogrammet må skrives noe om.
3. *Hent og kjøre MKR GPS-testprogram* `(Testprogram-MKR-GPS-2B.ino)`
   Med biblioteket følger også eksempelkode som bl.a. inneholder et testprogram. Hent testprogrammet fra vedlegg E.2.3 side 208 og se at GPS-data leses og skrives til monitoren.
4. *Inkluder GPS-testprogrammet i hovedprogrammet* `(Sea-buoy-3)`
   Vi skal nå fortsette å bygge ut hovedprogrammet som samler inn måledata og sender disse til serveren. Dette gjør vi ved å inkludere nødvendige kommandoer fra eksempelprogrammet (vedlegg E.2.3 side 208) til testprogrammet for overføring av dummy-verdier (vedlegg E.3.2 side 223) eller aller helst bygge videre på programmet som samler data fra miljøkortet (MKR ENV se deloppdrag 5 side 78).
5. *Test og kjør hovedprogrammet*
   Test det utvidete hovedprogrammet og sjekke at måledata kommer inn og vises på utskriften hos serveren. Alternativt kan man studere løsningsforslaget som er lagt ved i vedlegg G.2 side 253 og bruke dette for å teste forbindelsen.

4.7 Deloppdrag 7: Monter sensorkortet for måling av vanntemperaturen

Dette kortet skal *stå nederst i stabelen* og inneholder foreløpig kun den vanntette temperatursensoren DS18B20 for måling av temperatur i vannet. På dette kortet kan man montere flere sensorer så lenge det er plass.

> **Deloppdrag 7:** *Kjør mikrokontrollerkortet alene sammen med sensorkortet for måling av vanntemperaturmåleren. Stikk sensoren ned i et glass med vann og undersøk om temperaturen er som forventet. Dernest skal avlesningen av temperaturen legges inn i datastrengen i hovedprogrammet og sendes over til serveren*

1. *Monter sensorkortet*
   Plugg mikrokontrollerkortet MKR NB 1500, ned i sensorkortet med tilkobling for temperatursensoren, Koble DS18B20 til skrukoblingene på kortet omtalt i avsnitt 7.1.3 side 122.

2. *Installer biblioteket*
   Last ned og installer biblioteket for sensoren DS18B20. Dette er omtalt i avsnitt 7.1.3 under punkt 2. side 80.
3. *Last opp testprogrammet* `(Testprogram-DS18B20-Temp-1.ino)`
   Hent opp og last inn testprogrammet for test av DS18B20 sensoren. Se at du får rimelige temperaturverdier. Dette er omtalt i avsnitt 7.1.3 side 122.
4. *Mål kjent temperatur i vann*
   Stikk sensoren i vann med ulik temperatur å undersøk at den gir rimelige resultater. Kontroll målingene med et kalibrert termometer.

4.8 Deloppdrag 8: Inkludere vanntemperatur i hovedprogrammet

I dette deloppdraget skal vi inkludere måling av vanntemperaturen i hovedprogrammet.

Deloppdrag 8: *Inkluder måleresultatene fra vanntemperatursensoren i hovedprogrammet og send samtlige målinger til serveren. Kontroller at lagrede data er som forventet*

1. *Inkluder temperaturmålinger i hovedprogrammet*
   Med utgangspunkt i testprogrammet, lag en funksjon i hovedprogrammet som leser av sensoren og legger måleresultatet inn i datastrengen som sendes over til serveren.
2. *Formater datafilen*
   Sørg for at dataene inneholder *kun tallverdiene* uten tekst slik at det er lettere å hente dataene inn i Excel. Lag en overskrift i resultatfilen som forteller hvilke data hver av kolonnene inneholder.
3. *Overfør data til serveren*
   Kjør den nye utgaven av hovedprogrammet og gå inn på serveren og sjekk at du har mottatt de inkluderte målingene av vanntemperaturen. Alternativt kan man studere løsningsforslaget som er lagt ved i vedlegg G.4 side 267 (sea-buoy-5A.ino) og bruke dette for å teste overføringen til serveren.

4.9 Deloppdrag 9: Presentasjon av måleresultatene

I dette deloppdraget skal vi presentere måleresultatene.

Deloppdrag 9: *Presenter måleresultatene med passende programvare, f.eks. Excel og Google Earth. Kommenter resultatene.*

1. *Lag grafer av målinger*
   Presenter måledata som funksjon av tiden. Vis gjerne luft- og vanntemperatur i samme diagram. Bruk gjerne Excel. Se ev. beskrivelse i avsnitt 6.2 side 101.
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 6.5 side 108.

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

4.10 Deloppdrag 10: Legg mikrokontrollerkortet i dvale mellom hver måling

Reduksjon av strømforbruket er avgjørende dersom målinger skal gå over lengre tid, dvs flere dager uker eller måneder. Det er derfor viktig å kunne legge mikrokontrollerkortet i dvale, det samme gjelder ENV- og GPS-kortet og andre sensorer som kobles til sensornoden.

> **Deloppdrag 10:** *Kjør mikrokontrollerkortet alene og undersøk strømforbruket når kortet legges i dvale. Vekk kortet etter en stund å mål strømforbruket ved vanlig drift. Koble til aktuelle sensorkort og la mikrokontrolleren legge disse i stand by. Mål strømforbruket når hele sensornoden er lagt i dvale og når den er i normal drift. Estimer forventet driftstid på batteriet.*

1. *Legg mikrokontrollerkortet i dvale og mål strømforbruk*
   Tilfør hovedprogrammet tilstrekkelig kode for å legge mikrokontrolleren i dyp dvale over en viss tid. Mål strømforbruket til mikrokontrollerkortet under normal drift og dyp dvale. Se avsnitt 3.3 side 51 og avsnitt 3.4 side 55. Et testprogram for å legge MKR NB 1500 i dvale er lagt ved i E.2.11 side 219.
2. *Legg den komplette sensornoden i dvale og mål strømforbruk*
   Tilfør hovedprogrammet tilstrekkelig kode for å legge sensornoden i dyp dvale over en viss tid. Mål strømforbruket til sensornoden under normal drift og dyp dvale. Dersom flere tilleggskretser er inkludert i systemet, kan det være lurt å måle strømforbruk ved dvale og ved normal drift for alle involverte kort.
3. *Estimer driftstiden til et valgt batteri*
   Ut fra innhentede måledata estimer driftstiden til det valgte batteriet.

Bruk av RTC
Dersom strømforbruket er for høyet, vurder bruk av RTC-krets som kan brukes til å slå av strømmen til mikrokontrolleren. Se avsnitt *Bruk RTC og "power booster" til å slå av mikrokontrolleren på side 68*.

5 Installasjon av programpakker

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

5.1 Installasjon av programvare

Om du har installert Arduino IDE tidligere kan du hoppe over dette avsnittet og gå direkte til avsnitt 5.2 side 85.

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

5.1.1 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.

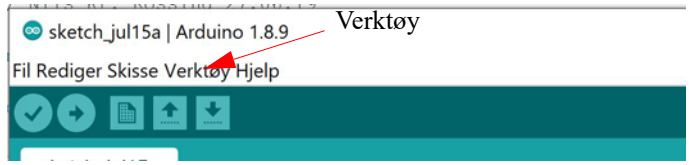
Installasjon av programvaren:

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 5.2 på side 85), 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. Dersom 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.

Dernest 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 goden til mikrokontrolleren.

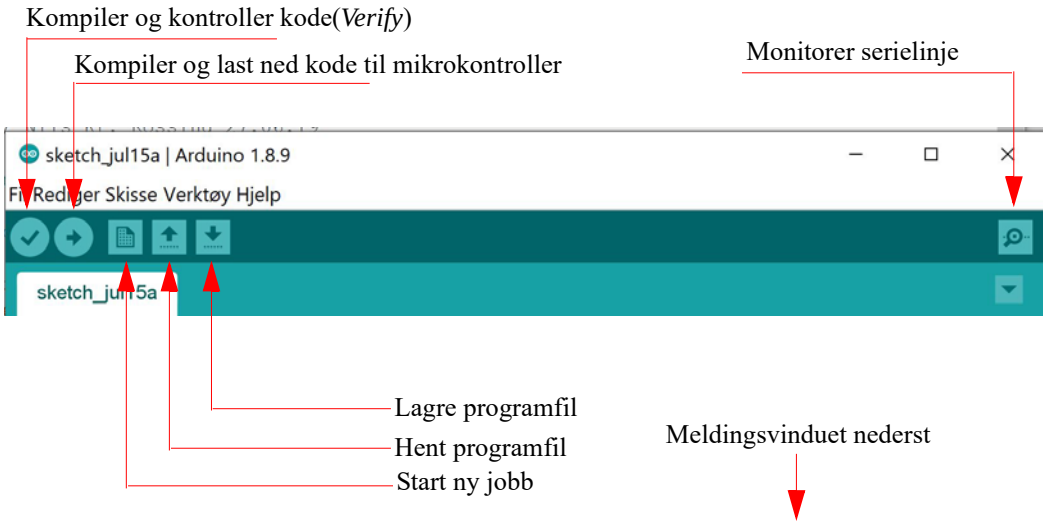
 Kompiler og last ned programmet til mikrokontrolleren

 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

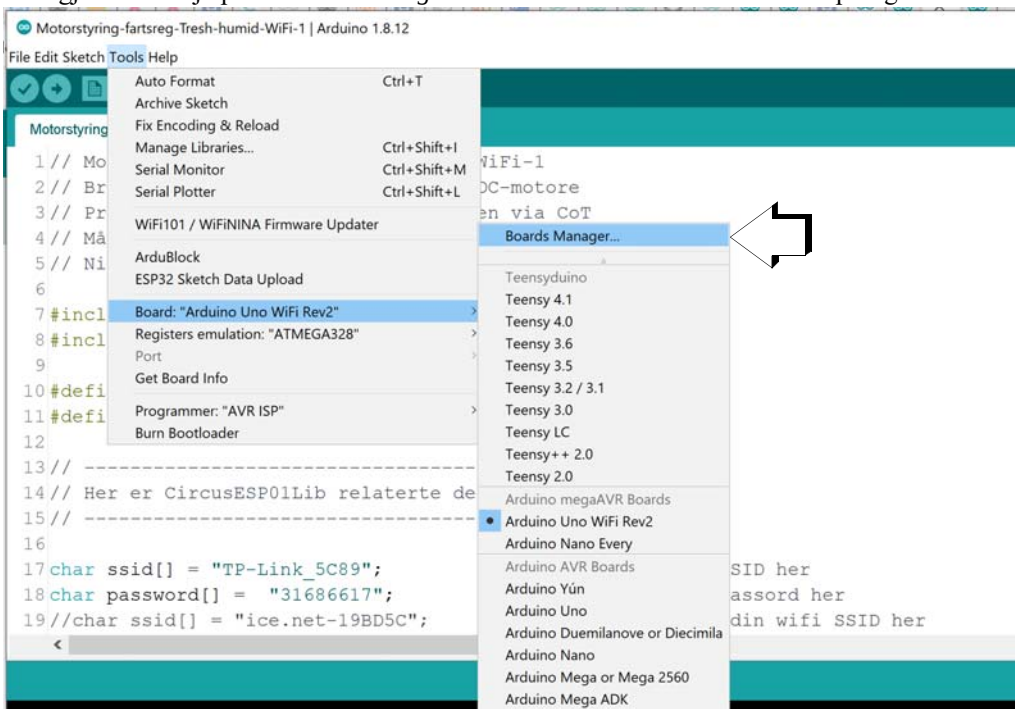
Det hender at en ikke oppnår umiddelbar kontakt med Arduino-kortet når en forsøker å laste ned et program. Feilmeldingen: *avrdude: stk500_getsync(): not in sync: resp=0x00* i meldingsvinduet betyr at det ikke oppnås kontakt med kortet. Dette kan skyldes flere ting:

- 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.*
- 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)[52]. I så fall lukkes programmet som stenger for overføringen og man prøver å overføre Arduino-programmet på nytt.

5.2 Installasjon av ekstra pakke for programmering av MKR NB 1500

Vi skal nå installere en programpakke som 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.

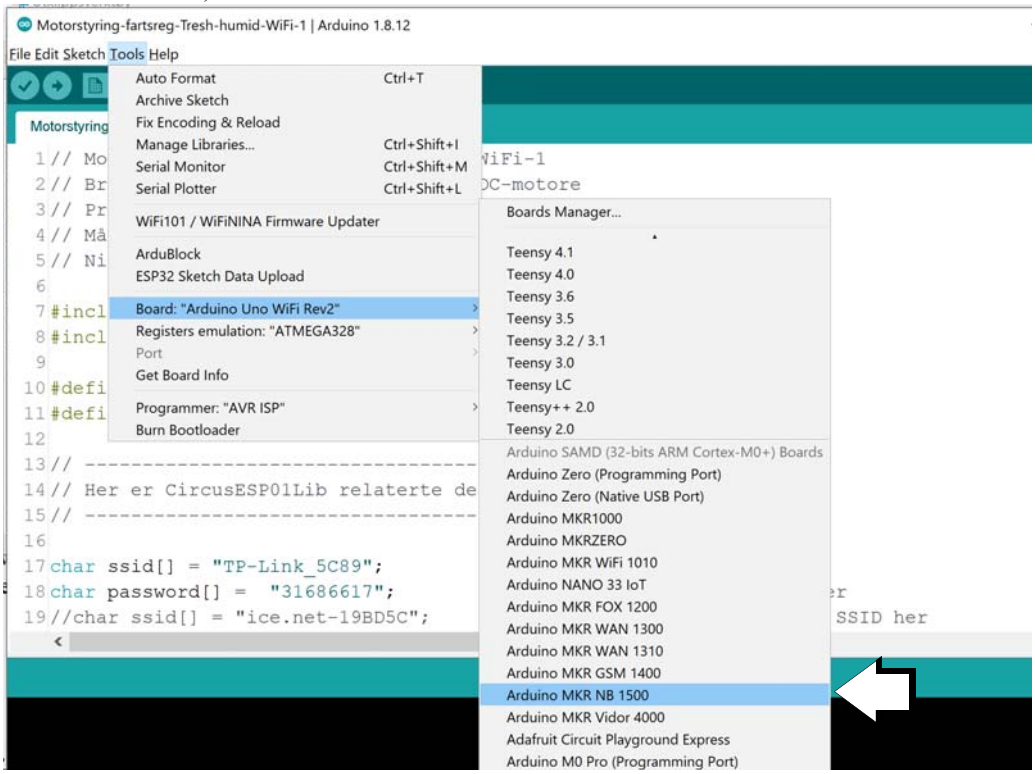


52. Kan se ut som om dette problemet er fjernet på nyere utgaver av CURA.

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:

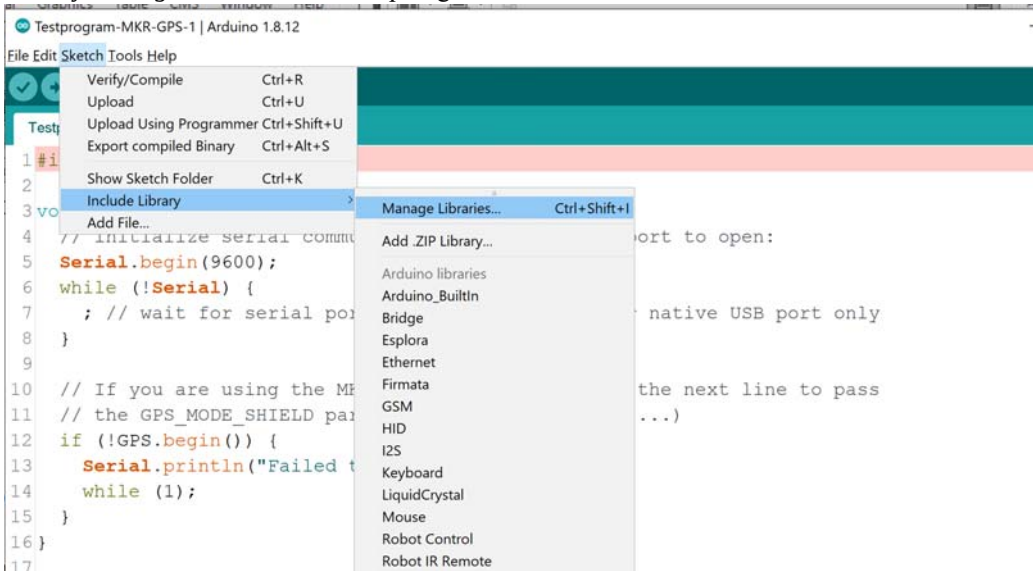
Vi velger det kortet (rammen) som kommer opp og trykker INSTALL nederst til høyre. Programvaren for denne kort-familien består av 6 pakker og det tar litt tid å laste ned og installere den.

Når vi skal programmere kortet MKR NB 1500, må vi huske å velge dette kortet fra menyen av mulige kort, se figuren til høyre. Vi går da inn på Tool –> Boards –> Arduino SAMD(32-bits ARM Cortex-M0+) Boards –> Arduino MKR NB 1500.

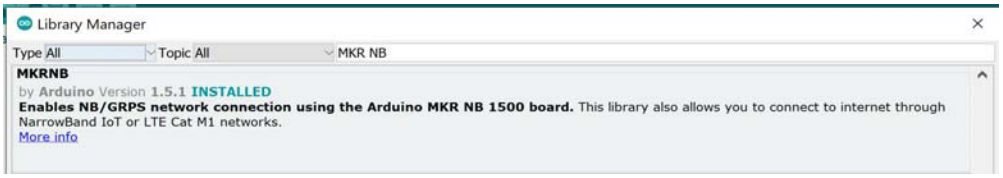


5.3 Installasjon av bibliotek for programmering av MKR NB 1500

Det finnes en rekke shield-kort som kan plugges på MKR NB 1500 med tilhørende biblioteker. Her skal vi i første omgang kun beskrive biblioteket vi trenger for å kunne kommunisere smalbånds 4G. Vi bruker da *Library manager* som vi finner ved å gå vi 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.

5.4 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[53]. Vi må også bestemme hvilken teknologi vi ønsker å bruke. Det er spesielt to som er aktuelle[54]:

53. 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 kjernenett for mobilproduksjon i Norge. https://www.com4.no/loesninger/m2miot-kommunikasjon/
54. https://www.telenor.no/bedrift/iot/abonnement/oversikt/

NB-IoT

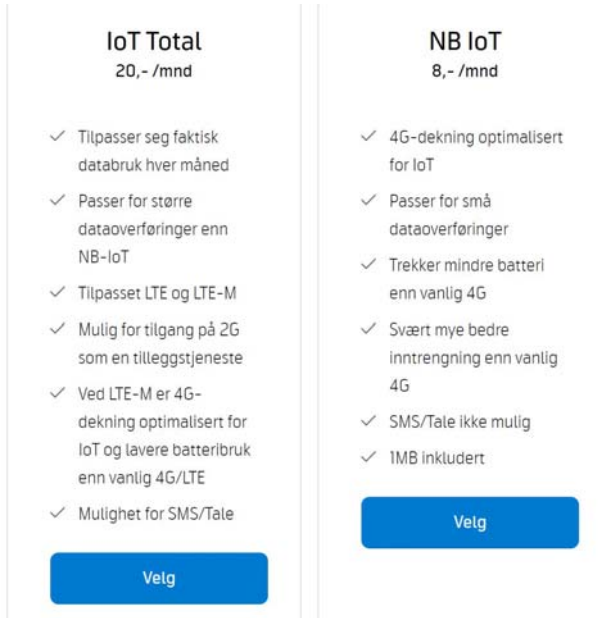
Narrowband IoT (NB-IoT) er en IoT-teknologi basert på 4G-nettet som fungerer praktisk talt hvor som helst i landet. Den egner seg godt der hvor "tilkoblede ting" er lokalisert i avsidesliggende eller vanskelig tilgjengelige områder. Med lange avstander fra den neste mobile basestasjonen, eller i skjermede 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 grisgrendte 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.

Et naturlig spørsmål er hva Telenor legger i begrepet *befolkningsdekning*, 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 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.

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.[55]



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 sammenhengen opptrer vi som en bedrift.

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

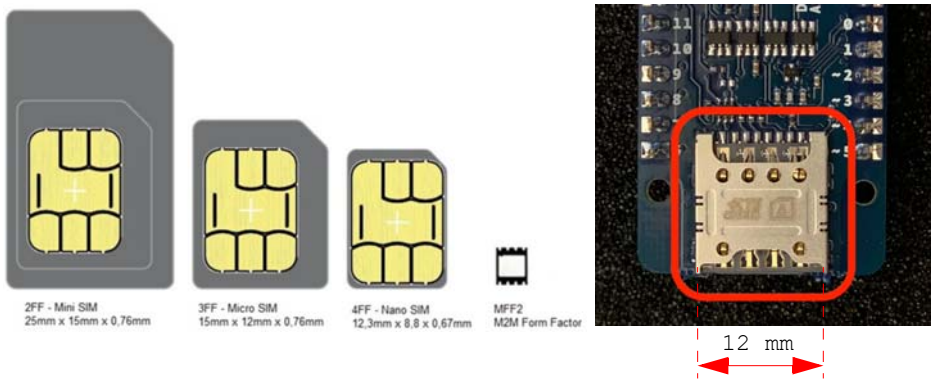
55.https://bedriftsbutikk.telenor.no/m2m/

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.



5.4.1 Bestilling av prøveabonnement for uttesting – inntil 5 SIM-kort for 5 måneder

Dersom man kun ønsker å bli kjent med teknologien før man inngår et endelig abonnement, kan man få et gratis prøveabonnement som varer i 5 måneder: https://www.telenor.no/bedrift/iot/kontakt/bestilling-sim/

5.4.2 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

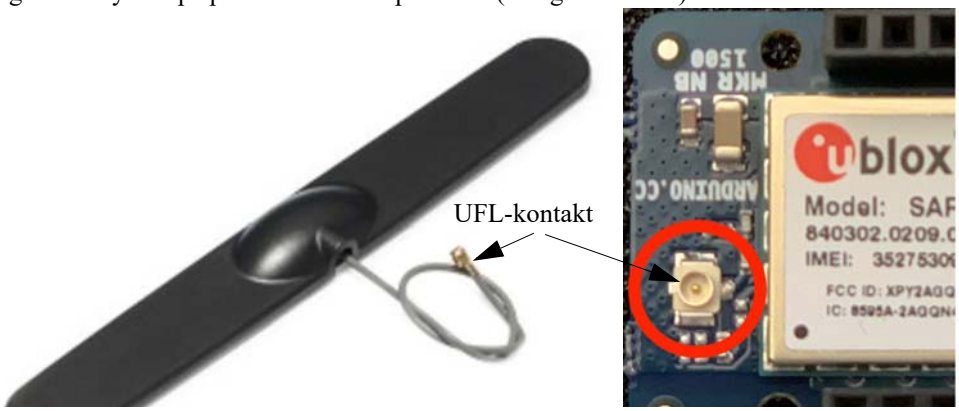
Velg og gå til kassen og betal

5.4.3 Bestilling av et antall fra 10 eller flere

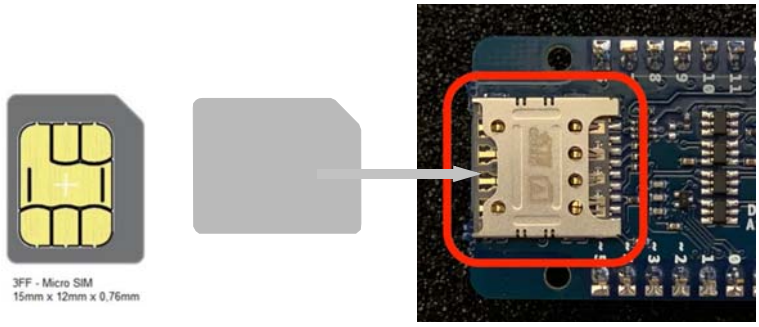
Dersom antallet kort er lik eller større enn 10 så må man opprette en bedriftskonto og være en autorisert bestiller i bedriften. Man går først til adressen: https://www.telenor.no/bedrift/min-bedrift/ hvor man blir registrert som bedriftens autoriserte innkjøper.

5.5 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 antydet hvilken vei kortes 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.

5.6 Bestem SIM-kortets IMSI og IMEI

I den løsningen vi beskriver her trenger vi strengt tatt ikke å kjenne SIM-kortets og mikrokontrollerens ID, men det kan være greit å vite hvordan vi henter ut denne informasjonen.

For å bestemme IMSI og IMEI må vi legge inn et program i MKR NB 1500 som gjør oss i stand til å hente ut data fra kortets ID-koder. Men la oss først se hva IMSI og IMEI står for?

> **IMSI – International Mobile Subscriber Identity**[56]**.** *Dette er en unik identifikator som definerer en abonnent i den trådløse verdenen, inkludert landet og mobilnettverket som abonnenten tilhører. Den har formatet MCC-MNC-MSIN. MCC = Mobile Country Code (f.eks. 047 for Norge*[57]*); MNC = Mobile Network Code (f.eks. 242 for Telenor*[58]*), MSIN = sekvensielt serienummer. All signalering og meldinger i GSM- og UMTS-nettverk bruker IMSI som den primære identifikatoren til en abonnent. IMSI er lagret på SIM-kortet. IMSI-nummeret består av 15 siffer.*
>
> **IMEI – International Mobile Equipment Identity** *er et unikt nummer som gis til hver enkelt mobiltelefon, vanligvis finnes det bak batteriet. IMEI-nummeret til mobiltelefoner koblet til et GSM-nettverk, lagres i en database (EIR – Equipment Identity Register) som inneholder numrene til alt gyldig mobiltelefonutstyr. Når en telefon meldes stjålet eller ikke er typegodkjent, merkes nummeret som ugyldig. IMEI-nummeret består også av 15 siffer.*

For å få tak i IMSI- og IMEI-numrene må vi lese SIM-kortet og Arduino-kortets serienummer. Det kan vi gjøre med et lite program som vi laster inn og kjører i MKR NB 1500 mens SIM-kortet står i holderen.

Kompiler og last opp programmet[59] (se også vedlegg E.3.2 side 223):

```
// baud rate used for both Serial ports
  unsigned long baud = 115200;

  void setup() {
    // enable the POW_ON pin
    pinMode(SARA_PWR_ON, OUTPUT);
    digitalWrite(SARA_PWR_ON, HIGH);

    // reset the ublox module
    pinMode(SARA_RESETN, OUTPUT);
    digitalWrite(SARA_RESETN, HIGH);
    delay(100);
    digitalWrite(SARA_RESETN, LOW);

    Serial.begin(baud);
    SerialSARA.begin(baud);
  }
```

56. https://m.blog.naver.com/framkang/220363349346
57. https://en.wikipedia.org/wiki/Mobile_country_code
58. https://www.3glteinfo.com/mobile-country-code-mcc-and-mobile-network-code-mnc/
59. https://forum.arduino.cc/t/getting-the-mkr-nb-1500-with-the-sara-r410m-to-connect-to-an-mqtt-server/612502

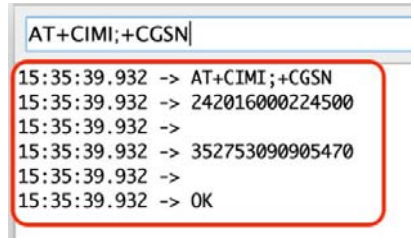
```
void loop() {
  if (Serial.available()) {
    SerialSARA.write(Serial.read());
  }

  if (SerialSARA.available()) {
   Serial.write(SerialSARA.read());
  }
}
```

Åpne og sett opp monitoren på som vist på figuren til høyre:

Skriv: AT+CIMI;+CGSN på kommandolinjen.

Legg merke til at kommandoen AT+CIMI[60] er en kommando som etterspør og returnerer IMSI-nummeret til SIM-kortet, mens AT+CGSN[61] er en kommando som etterspør og returnerer IMEI-nummeret til Arduino-kortet (dvs. serienummeret).

Sett kommunikasjonshastigheten til 115 200 baud (2) og formatet til "New line" og "Carrage Return" (Both NL & CR) (1). Skriv *AT+CIMI;+CGSN* i kommandolinjen (3) og trykk Send (4). Her etterspør CGSN produktets serienummer.



Man skal da få et svar som ligner på det som er vist på figuren til høyre. Vi legger merke til at det første nummeret (CIMI) starter med 242 som er identifikasjonen for Telenor. Det andre nummeret angir CGSN. Ellers består tallet av 15 siffer som forventet.

Ta vare på disse numrene, vi skal bruke dem når vi oppretter en konto i IoT-skyen (MIC).

5.7 Sending og mottak av meldinger via NB-IoT til serveren

5.7.1 Testing av forbindelse med serveren `(Eksempel-kode-1.ino)`

Til dette bruker vi et enkelt testprogram som legger litt tekst ned på serveren.

60. https://m2msupport.net/m2msupport/atcimi-request-international-mobile-subscriber-identity/
61. https://m2msupport.net/m2msupport/atcgsn-request-product-serial-number-identification/

Hent testprogrammet: `Eksempel-kode-1.ino` (vedlegg E.3.1 side 221).

I resten av avsnittet skal vi gå gjennom koden slik at vi forstår hva programmet gjør.

5.7.2 Gjennomgang av programkoden `(Eksempel-kode-1.ino)`

La oss se nærmere på hvordan vi sender over data fra mikrokontrolleren til serveren. For å gjøre dette trenger vi et bibliotek med et kommandosett som kan koble sensornoden opp til serveren via 4G-nettverket ved bruk av GPRS (General Packet Radio Services) som baserer seg på å sende pakker av data. På denne måten er det mulig å dele en radiokanal med mange andre, hvilket gjør nettverket effektivt og bruken billig for den som trenger å overføre små datamengder.

Programvaren i mikrokontrolleren må derfor:

1. Koble seg opp mot 4G-nettverket (GPRS/NB-IoT)
2. Koble seg opp mot den aktuelle serveren
   Her trenger man bl.a. informasjon om IP-adressen og sti til den aktuelle katalogen hos serveren

La oss se på et eksempelprogram som følger med biblioteket

Vi tar da utgangspunkt i en av eksempelfilene laget av Tom Igoe i 2012 – NBWEBClient.ino.

1. **Inkluder biblioteket**
   Vi har tidligere installert biblioteket med header-filen: MKRNB.h. Nå inkluderer vi det i programvaren med følgende kommando:

   ```
   #include <MKRNB.h>
   const char PINNUMBER[]     = “”;
   ```

   Denne legges øverst i programfila

2. **Definisjon av strenger**
   Følgende objekter defineres. Disse skal vi aktivt bruke senere i programmet

   ```
   char server[] = "sensor.marin.ntnu.no";
   char path[] = "/cgi-bin/tof.cgi?";
   int port = 80; // port 80 er default for HTTP
   ```

   Her ser vi de valg som er gjort ved Inst. for marin teknikk når det gjelder serveren og stien til serveren. Denne kan endres etter som man får på plass en server enten for testing eller endelig bruk.

3. **Koble til 4G-nettet (GPRS)**
   Så skal vi koble kretsen til 4G-nettet (GPRS). Dette gjøres med disse kommandoene:

   ```
   boolean connected = false;
   char buffer[256]; // Buffer som holder filnavn og måledata

   while (!connected)
   {
      if ((nbAccess.begin(PINNUMBER) == NB_READY) &&
         (gprs.attachGPRS() == GPRS_READY))
   ```

```
    {
      connected = true;
    }
    else
    {
     Serial.println("Not connected");
     delay(1000);
    }
  }
```

Denne er valgt lagt i setup()-funksjonen. Vi legger merke til programmet går i sløyfe helt til den oppnår kontakt og `connect = true;` Inntil den får kontakt skriver den ut "Not connected".

4. **Koble opp mot serveren, gjør en forespørsel og send over data**
   Når vi er kommet ut på 4G-nettet er det på tide å henvende seg til serveren som vi alt har lagt inn adressen til:

```
delay(2000);
if (client.connect(server, port))
{
  Serial.println("connected");
  client.print("GET "); // Make a HTTP request:
  sprintf(buffer,"/cgi-bin/tof.cgi?Gruppe-1.txt,lat=%.6f,lon=%.6f",67.5, 5.6);
  client.print(buffer);
  client.println(" HTTP/1.1");
  client.print("Host: ");
  client.println(server);
  client.println("Connection: close");
  client.println();
}
else
{
  Serial.println("connection failed");
}
```

Det viser seg at det er viktig å legge inn en tidsforsinkelse mellom oppkobling til GPRS-tjenesten og oppkoblingen til serveren. Vi har valgt å legge inn en forsinkelse på 2 sek.

Vi legger merke til `client.print(path);` er byttet ut med et buffer som består av en tekststreng bygget opp av stien med filnavnet: `"/cgi-bin/tof.cgi?,` og som deretter følges av det som skal legges i fila. En har i dette eksempelet valgt å starte med navnet på den gruppen som står ansvarlig for å samle inn disse dataene, for deretter å legge inn to merkelapper (lat= og lon=), for så å fylle på med innholdet i de to variablene. Å legge inn tekst i selve datafile er kanskje ikke så lurt med tanke på etterbehandling, men gjør det lettere å lese dataene i tekstfila.

Denne er også lagt inn i setup()-funksjonen. Dvs. i dette eksempelet skrives kun en datalinje før programmet termineres i loop()-funksjonen

5. **Loop()-funksjonen**
   I dette tilfellet består loop()-funksjonen kun av lytting etter kommandoer sendt fra serveren, for deretter å stoppe.

```
if (client.available())
{
  Serial.print((char)client.read());
}
// if the server's disconnected, stop the client:
if (!client.available() && !client.connected())
{
  Serial.println();
  Serial.println("disconnecting.");
  client.stop();
  for (;;);  // Do nothing forevermore:
}
```

Dette enkle testprogrammet beskrevet foran finnes i E.3.1 side 221.

5.7.3 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. I første omgang er det lurt å begynne enkelt med å sende over dummy-data for så å bygge ut med reelle data i det omfanget som er nødvendig for å få med hele datasettet.

6. **Oppbygging av buffer for overføring av data**
   Vi har tidligere deklarert denne tekststrengen under punkt 4. side 94:

```
sprintf(buffer, "/tof/tof.cgi?Gr-1.txt,lat=%.lf,lon=%.6f", latitude, longitude);
client.print(buffer);
```

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

   `sprintf()` står for "string print format" og vil lage en streng, satt sammen av tekst og tallverdier som her.

   `buffer,` som er det første elementet i argumentet, angir hvor den endelige tekststrengen skal plasseres. Denne er tidligere deklarert som et buffer på 256 karakterer og kan etter behov gjøres lengre (se punkt 3. side 93). Selve tekststrengen er plassert i hermetegn: " " .

   `/tof` angir katalogen der fila skal legges.

   `/tof.cgi?` er et lite skript (program) som web-servere 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 "?", i dette eksempelet ...

   Gr-1.txt, og opprette en fil med dette navnet, evt. skrive inn i en eksisterende fil med navnet. Deretter legges dataene fortløpende inn i fila i henhold til angitt format. Dette er en tekst-fil. Ved å føye til .txt vil den lett la seg åpne med en text-editor f.eks. notepad.

   `lat=%.lf,lon=%.lf", -` Deretter følger teksten som skal stå i den første raden, `lat=%.lf,lon=%.lf",`. Inne i teksten ønsker vi å plassere resultatene, som i vårt eksempel er bredde- (lat=) og lengdegrader (lon=). Stedene i teksten der vi ønsker å plassere disse verdiene, angis med en *formatspesifikasjon,* et %-tegn ("format specifier"). I tillegg

angis antall siffer etter komme med ".n". Dvs. at ".6" betyr at variablene skal angis med 6 desimaler. "f" angir at det er snakk om et flyttall. "i", et heltall, "d" et desimaltall og "s" at det er en tekststreng. Denne bokstaven kalles *formatkarakteren* ("Format character").

`latitude, longitude); –` Etter hermetegnet kommer listen over variable som skal settes inn i tekst-strengen på de angitte plassene. I vårt tilfelle er dette *latitude* og *longitude*. Man kan eventuelt sette inn rene tall (`67.5, 5.6`).

Den ferdige tekststrengen legges i strengvariabelen: *buffer*. Deretter sendes innholdet av bufferet 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. I vårt tilfelle kan det være interessant å ha med en rekke variabler som vist:

```
sprintf(buffer, "/tof/
tof.cgi?Gr.1.txt,no=%d,time=%.2f,lat=%.6f,lon=%.6f,C_luft=%.1f,
C_vann=%.1f,hum=%.1f,pres=%.1f,light=%.1f", no, timeSec,
longitude, latitude, temperature, w_temperature, humidity,
pressure, illuminance);
```

Med tanke på at dataene skal kunne leses enkelt ved hjelp av Excel kan det også være lurt å droppe teksten mellom hver verdi:

```
sprintf(buffer, "/tof/  tof.cgi? Gr-1.txt,
%d,%.2f,%.6f,%.6f,%.1f,%.1f,%.1f,%.1f,%.1f",
no, timeSec, longitude, latitude, temperature, w_temperature,
humidity, pressure, illuminance);
```

Denne strengen er ikke så lett å lese, men egner seg godt for Excel.

5.7.4 Et funksjonelt program for overføring av data `(Eksempel-kode-3.ino)`

Det neste testprogrammet er en videreutvikling av `Eksempel-kode-1.ino`. Følgende endringer er gjort:

1. Alle målevariablene som vi trenger er deklarert som globale variabler
2. Oppkobling til 4G og server er lagt i egne funksjoner som kalles fra loop()-funksjonen
3. Alle målevariabler er tilordnet dummy-verdier i egne funksjoner, hvorav noen er tildelt randomiserte verdier

Hent testprogrammet: `Eksempel-kode-3.ino` (vedlegg E.3.2 side 223).

Deklarasjon av variabler

Vi har valgt å deklarere alle målevariabler som globale variabler slik at det skal være enkelt å utveksle verdier mellom funksjonene:

```
// Deklarasjon av sensorvariabler fra ENV-kort
float temperature = 0; // Målt lufttemperatur i C
float humidity    = 0; // Målt luftfuktighet i %
float pressure    = 0; // Målt lufttrykk i mBar
float illuminance = 0; // Målt lysintensitet i Lux

// Deklarer sensorvariable fra vannsensorer
float w_temperature = 0;// Målt vanntemperatur

// Deklarasjon av sensorvariabler fra GPS-kort
float latitude;     // Breddegrad
float longitude;    // Lengdegrad
float altitude;     // Høyde over havet
float speed;        // Beregnet hastighet
float timeSec;      // Antall sekunder fra 1.1.1980 - Epoketid
int   satellites;   // Antall satellitter
long  no = 0;       // Måling nummer
```

Oppkobling til 4G og server, og målinger er lagt i egne funksjoner

Eksempelkoden er bygget opp av en rekke funksjoner som skal gjøre det lett å få oversikt over gangen i programmet. Vi har også lagt inn en for loop() slik at vi generer et mindre antall målinger under uttestingen:

```
void loop()
{
  for(int i=1; i<6; i++)
  {
   no=i;
   readGPSdata();          // Les GPS-posisjon mm
   readENVdata();          // Les ENV-målinger
   readWaterData();        // Les av sensorer i vann
   printDataToMonitor();// Skriv måleverdier til monitor i samme format som datafila
   connectToGPRS();        // Koble opp til GPRS nettverket med APN, login og passord
   Serial.print("Kobler til og overfører data: ");
   Serial.println(no);
   connectToServer();      // Overfør data til server
   //checkServerMessage();// Sjekk om det er noen melding fra serveren til
sensornoden
   delay(5000);
 }
 for(;;)                   // Stop programmet
 ;
}
```

De egen definerte funksjonene er markert i bått over og er gitt navn som er betegnende for hva de gjør. Det samme gjør kommentarene.

Funksjoner som utfører målinger

Vi viser her de tre funsjonene som tilordner måleverdier. Det er i disse funksjonene vi etter hvert skal legge inn de virkelige måleverdiene fra ENV- og GPS-kortet og fra målingen av temperaturen i vann. Foreløpig er det bare lagt inn dummy-verdier:

```
void readGPSdata()
{
Serial.println("Leser GPS-data");
latitude   = 63.426269 + random(0, 9)/100.0;
longitude  = 10.454045 + random(0, 9)/10.0;
altitude   = 65        + random(0, 4);
speed      = 0         + random(0, 9);
satellites = 6;
timeSec    = millis()/1000.0;
}
void readENVdata()
{
Serial.println("Leser miljo-data");
temperature = 21.0;
humidity    = 45.5;
pressure    = 1089.3;
illuminance = 148.5;
}
void readWaterData()
{
Serial.println("Leser vanntemperatur");
w_temperature = 12.0;
}
```

Legg spesielt merke til at det lagt inn randomiserte verdier i noen av desimalene i lengde- og bredde-gradene og i høyden for at vi ser at det skjer noe med verdiene.

Funksjoner som overfører målingene til serveren

Vi kjenner igjen kommandoene for å koble til og overføre data til serveren:

```
void connectToServer()
{
  // Koble opp til server og send over data, varsler om oppkobling mislykkes
  if (client.connect(server, port))
  {
    Serial.println("Tilkoblet server");
    client.print("GET "); // Gjør et HTTP request:
     sprintf(buffer, "/tof/tof.cgi?Gr-1.txt,no=%d,time=%.2f,lat=%.6f,lon=%.6f,
     C_luft=%.1f,C_vann=%.1f,hum=%.1f,pres=%.1f,light=%.1f", no, timeSec,
     longitude, latitude, temperature, w_temperature, humidity, pressure,
     illuminance); // Legg måleverdier inn i bufferet
    client.print(buffer);
```

```
        client.println(" HTTP/1.1");
        client.print("Host: ");
        client.println(server);
        client.println("Connection: close");
        client.println();
      }
    else
      {
        Serial.println("Mislykket tilkobling til server"); // Tilkobling mislyktes
      }
    }
```

Resten av funksjonene har vi omtalt tidligere eller de taler for seg selv.

6 Behandling og visning av måledata

I dette kapittelet skal vi ganske kort vise hvordan vi henter inn data til Excel og Google Earth for visning av resultatene.

6.1 Skriving til og henting av data fra server

6.1.1 Skriving 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 måleseriene 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 denne slik:

```
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 skript (program) som web-servere bruker for å styre data på serveren. I vårt tilfelle vil skriptet ta tak i det neste argumentet etter "?" i dette eksempelet Gr-no og opprette en fil med dette navnet, ev. skrive inn i en eksisterende fil med dette navnet. Dernest legges dataene fortløpende inn i file i henhold til ønsket format.

Det er derfor viktig at hver enkelt er seg bevisst hvilket filnavn som brukes slik at de finner igjen sine data.

6.1.2 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/Gruppe-1.txt
```

Kursdeltakerne må selv holde orden på filnavnene slik at dataene ikke blandes.

Man kan åpne fila med en 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.

6.2 Skrive data til file

6.2.1 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 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 er dataene tapt for alltid.

Følgende gir noen anbefalinger for organisering av data:

6.2.2 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 (ev. 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 tidspunktet hentet fra GPS-mottakeren (Epoch time) og legge inn dette. En nøyaktig tidsangivelse vil være essensiell dersom målinger skal brukes av meteorologisk institutt. Se *GPS biblioteket på side 46*.

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 (se avsnitt 3.5 side 61.

6.2.3 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.

6.2.4 Den endelige datafilen

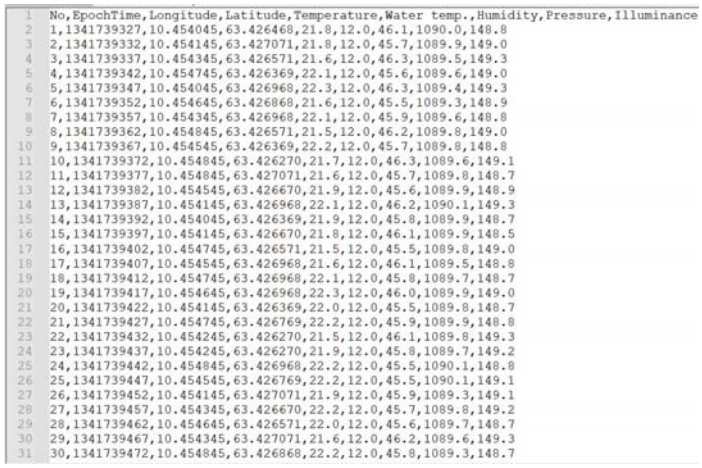
For å teste ut bruk av Excel har vi, som tidligere nevnt, laget en testfil med simulerte data. Denne viser et eksempel på hvordan dataene kan organiseres:

I filen 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
```

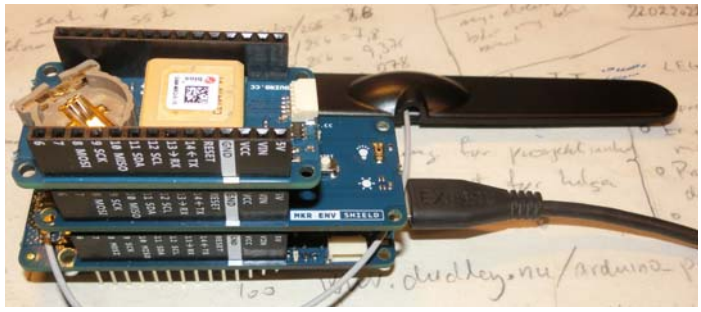
Figuren under viser den endelige filen vist ved hjelp av Notepad++ :



```
No,EpochTime,Longitude,Latitude,Temperature,Water temp.,Humidity,Pressure,Illuminance
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
7,1341739357,10.454345,63.426968,22.1,12.0,45.9,1089.6,148.8
8,1341739362,10.454845,63.426571,21.5,12.0,46.2,1089.8,149.0
9,1341739367,10.454545,63.426369,22.2,12.0,45.7,1089.8,148.8
10,1341739372,10.454845,63.426270,21.7,12.0,46.3,1089.6,149.1
11,1341739377,10.454845,63.427071,21.6,12.0,45.7,1089.8,148.7
12,1341739382,10.454545,63.426670,21.9,12.0,45.6,1089.9,148.9
13,1341739387,10.454145,63.426968,22.1,12.0,46.2,1090.1,149.3
14,1341739392,10.454045,63.426369,21.9,12.0,45.8,1089.9,148.7
15,1341739397,10.454145,63.426670,21.8,12.0,46.1,1089.9,148.5
16,1341739402,10.454745,63.426571,21.5,12.0,45.5,1089.8,149.0
17,1341739407,10.454545,63.426968,21.6,12.0,46.1,1089.5,148.8
18,1341739412,10.454745,63.426968,22.1,12.0,45.8,1089.7,148.7
19,1341739417,10.454645,63.426968,22.3,12.0,46.0,1089.9,149.0
20,1341739422,10.454145,63.426369,22.0,12.0,45.5,1089.8,148.7
21,1341739427,10.454745,63.426769,22.2,12.0,45.9,1089.9,148.8
22,1341739432,10.454245,63.426270,21.5,12.0,46.1,1089.8,149.3
23,1341739437,10.454245,63.426270,21.9,12.0,45.8,1089.7,149.2
24,1341739442,10.454845,63.426968,22.2,12.0,45.5,1090.1,148.8
25,1341739447,10.454545,63.426769,22.2,12.0,45.5,1090.1,149.1
26,1341739452,10.454145,63.427071,21.9,12.0,45.9,1089.3,149.1
27,1341739457,10.454345,63.426670,22.2,12.0,45.7,1089.8,149.2
28,1341739462,10.454645,63.426571,22.0,12.0,45.6,1089.7,148.7
29,1341739467,10.454345,63.427071,21.6,12.0,46.2,1089.6,149.3
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. Det er viktig at vi lagrer denne fila som en vanlig tekstfil eller CSV-fil (Comma Seperated Values).

6.3 Bruk av Excel for visualisering av data

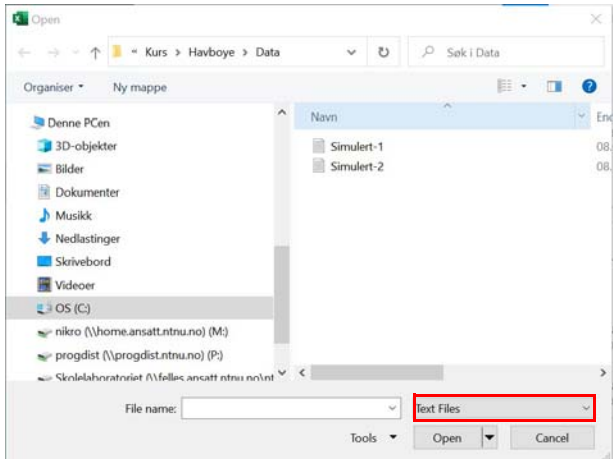


Et praktisk og vanlig verktøy for analyse av data er Excel. I dette kapittelet skal vi vise hvordan vi kan importere og tegne ut grafer av innsamlede data fra Havbøya. I dette eksempelet har vi følgende data: Måling nr., tidsangivelse (Epoketid), lengde- og breddegrad, luft- og vanntemperatur, luftfuktighet, lufttrykk og lysintensitet på overflata. Dataene for dette eksempelet er simulert.

6.3.1 Importer data fra en tekst-fil til Excel

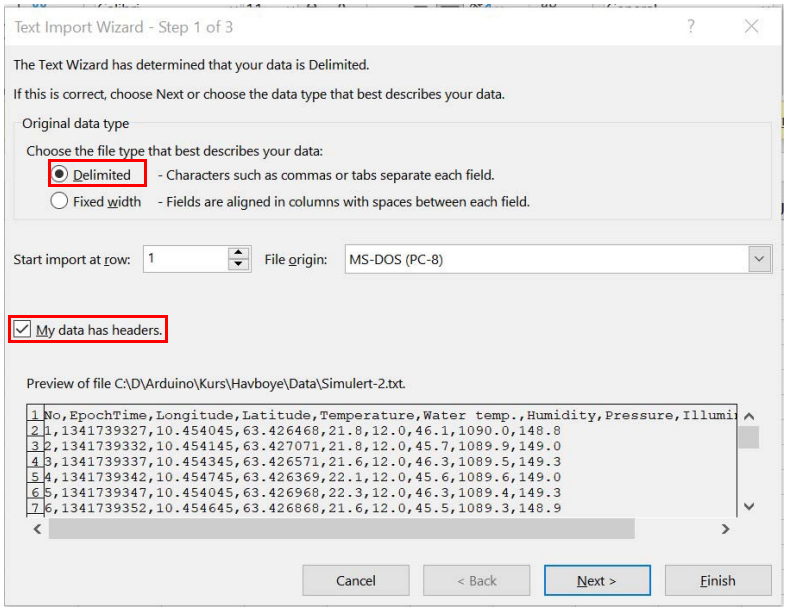
Importering av data i Excel kan gjøres på ulike måter. En måte som fungerer er å hente inn filen med "Open". Da vil man ledes gjennom følgende vinduer for at dataene skal bli formatert på ønsket måte.

Velg ønsket fil:

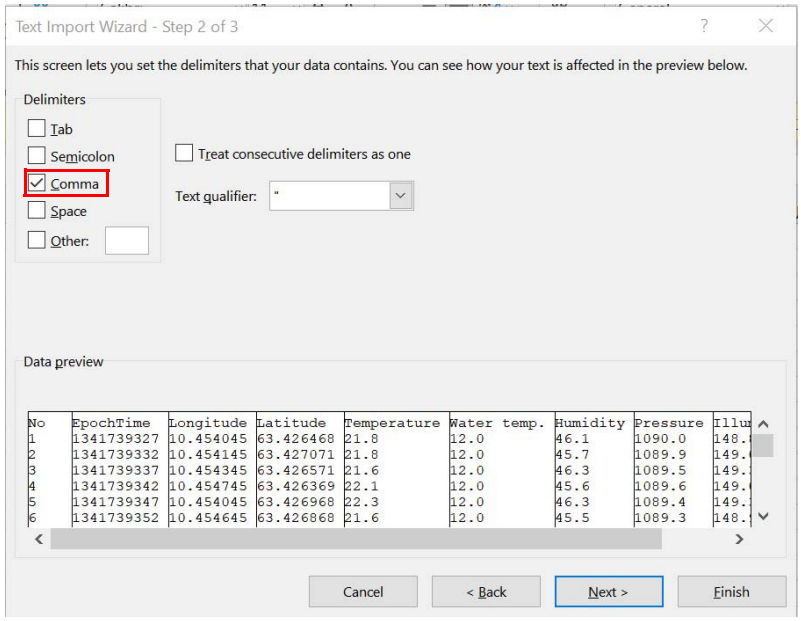


Siden det er en tekst-fil så velg denne typen format i innboksen nederst i høyre hjørne, dermed blir slike filer synlige i fil-oversikten.

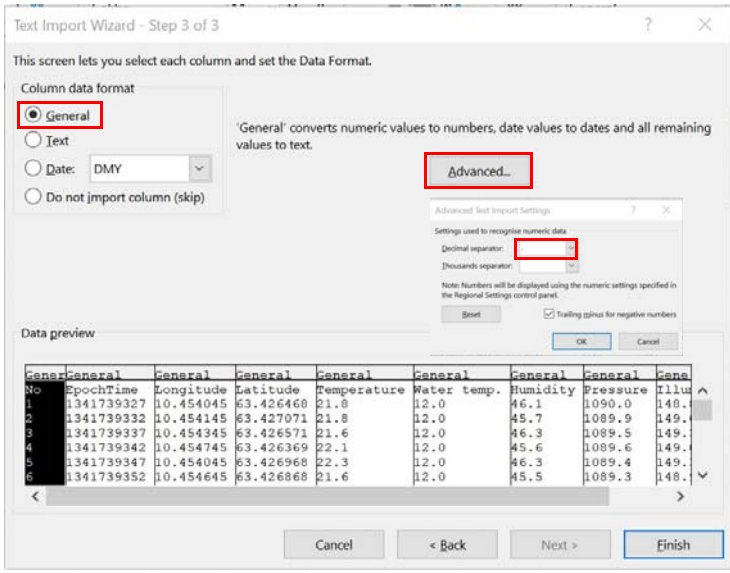
Velg “Data med skilletegn” (Delimited) og bestem fra hvilken rad du ønsker å starte å importere data. Likeså er det viktig å merke av om vi bruker en topptekst, som vi har gjort her. Trykk så “Neste”:



Velg hvilken type skilletegn som er benyttet. I vårt tilfelle har vi benyttet “komma”.



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.

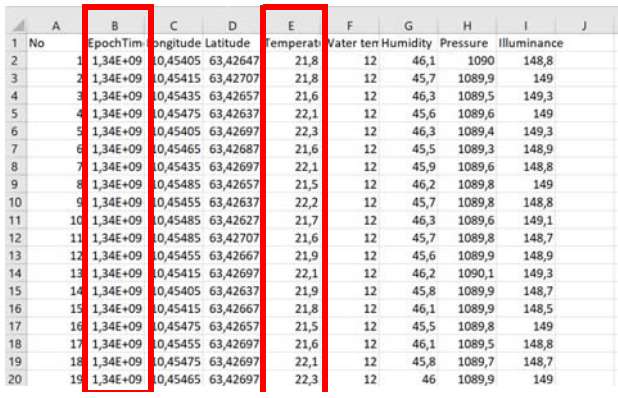
	A	B	C	D	E	F	G	H	I	J
1	No	EpochTim	Longitude	Latitude	Temperati	Water ten	Humidity	Pressure	Illuminance	
2	1	1,34E+09	10,45405	63,42647	21,8	12	46,1	1090	148,8	
3	2	1,34E+09	10,45415	63,42707	21,8	12	45,7	1089,9	149	
4	3	1,34E+09	10,45435	63,42657	21,6	12	46,3	1089,5	149,3	
5	4	1,34E+09	10,45475	63,42637	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	

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.

6.4 Lage grafer

Slik kan vi lage grafer:

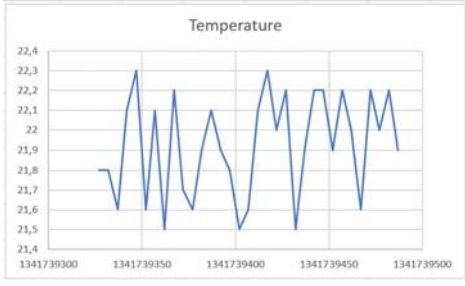
1. Merk de to kolonnene som skal representeres av en graf.
   Det kan være lurt å lage en overskrift på kolonnene om det ikke alt er gjort. Overskriften vil automatisk tilordnes seriene i grafen. Merk de kolonnene som skal utgjøre datasettet. I vårt tilfelle kan det være fornuftig å bruke Epoketiden langs x-aksen. Vi får da tegnet grafene som funksjon av tiden, f.eks. temperatur som funksjon av tiden.

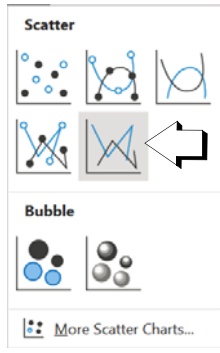


	A	B	C	D	E	F	G	H	I	J
1	No	EpochTim	ongitude	Latitude	Temperat	Water ten	Humidity	Pressure	Illuminance	
2	1	1,34E+09	10,45405	63,42647	21,8	12	46,1	1090	148,8	
3	2	1,34E+09	10,45415	63,42707	21,8	12	45,7	1089,9	149	
4	3	1,34E+09	10,45435	63,42657	21,6	12	46,3	1089,5	149,3	
5	4	1,34E+09	10,45475	63,42637	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	

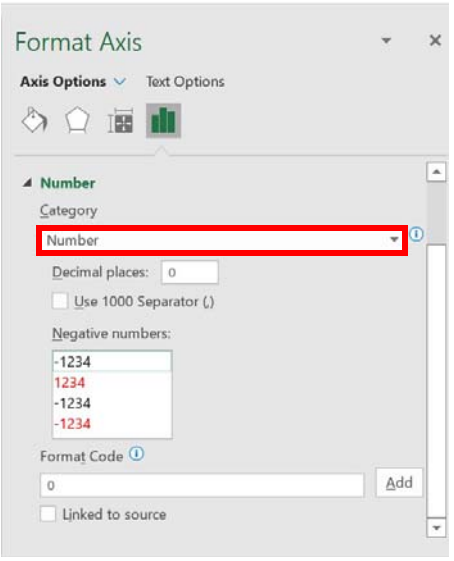
2. Velg type representasjon fra menyen "Sett inn" (Insert) og velg f.eks. punktdiagram med linjer mellom punktene i diagrammet:

3. Velg linjediagram som vist på figuren til høyre fra nedtrekksmenyen.
4. Når dette er gjort vil diagrammet tegnes ut som vist på figuren under.

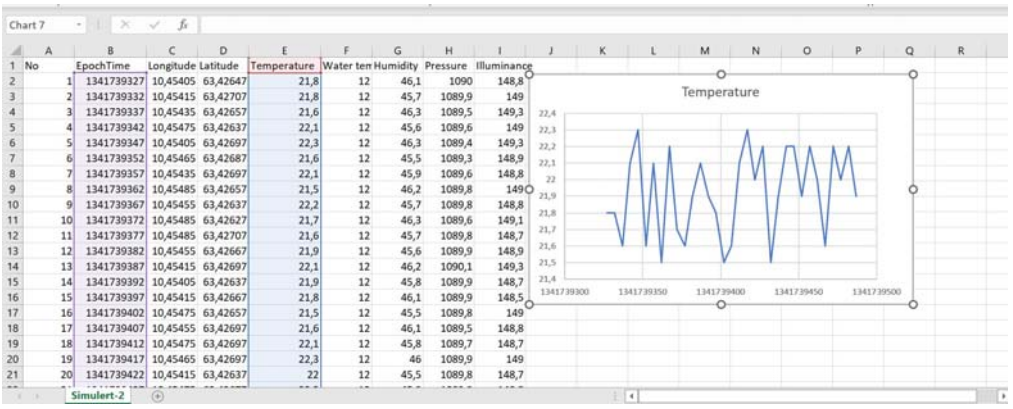




5. Ønsker vi å endre på formatet på aksene, så klikker vi på den akse-benevningen vi ønsker å endre og vi får opp vinduet vist til høyre. Her kan man f.eks. endre på tallformatet til akse-benevningen. F.eks. fra vitenskapelig notasjon til vanlig tallnotasjon for tidsangivelsen langs x-aksen.



6. Vi kan dermed ende opp med en presentasjon som vist under:

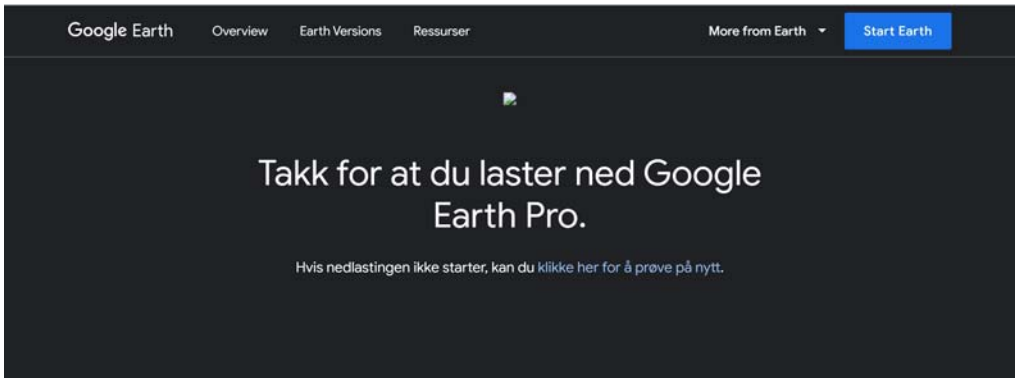


No	EpochTime	Longitude	Latitude	Temperature	Water ten	Humidity	Pressure	Illuminance
1	1341739327	10,45405	63,42647	21,8	12	46,1	1090	148,8
2	1341739332	10,45415	63,42707	21,8	12	45,7	1089,9	149
3	1341739337	10,45435	63,42657	21,6	12	46,3	1089,5	149,3
4	1341739342	10,45475	63,42637	22,1	12	45,6	1089,6	149
5	1341739347	10,45405	63,42697	22,3	12	46,3	1089,4	149,3
6	1341739352	10,45465	63,42687	21,6	12	45,5	1089,3	148,9
7	1341739357	10,45435	63,42697	22,1	12	45,9	1089,6	148,8
8	1341739362	10,45485	63,42657	21,5	12	46,2	1089,8	149
9	1341739367	10,45455	63,42637	22,2	12	45,7	1089,8	148,8
10	1341739372	10,45485	63,42627	21,7	12	46,3	1089,6	149,1
11	1341739377	10,45485	63,42707	21,6	12	45,7	1089,8	148,7
12	1341739382	10,45455	63,42667	21,9	12	45,6	1089,9	148,9
13	1341739387	10,45415	63,42697	22,1	12	46,2	1090,1	149,3
14	1341739392	10,45405	63,42637	21,9	12	45,8	1089,9	148,7
15	1341739397	10,45415	63,42667	21,8	12	46,1	1089,9	148,5
16	1341739402	10,45475	63,42657	21,5	12	45,5	1089,8	149
17	1341739407	10,45455	63,42697	21,6	12	46,1	1089,5	148,8
18	1341739412	10,45475	63,42697	22,1	12	45,8	1089,7	148,7
19	1341739417	10,45465	63,42697	22,3	12	46	1089,9	149
20	1341739422	10,45415	63,42637	22	12	45,5	1089,8	148,7

6.5 Bruk av Google Earth for visning av posisjon fra GPS

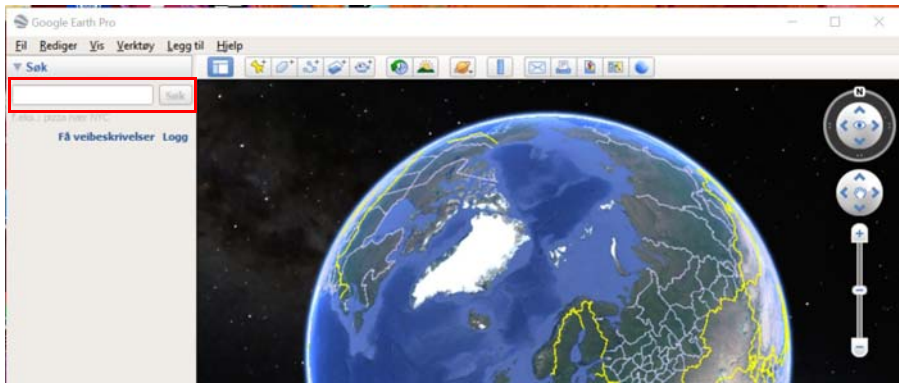
I dette avsnittet skal vi gi en oppskrift for hvordan vi kan vise en posisjon eller en trase i Google Earth.

Google Earth kan lastes ned og installeres fra følgende adresse: https://www.google.com/earth/versions/download-thank-you/



6.5.1 Plotting av en enkeltposisjon

Når man starter Google Earth så får man opp følgende vindu.



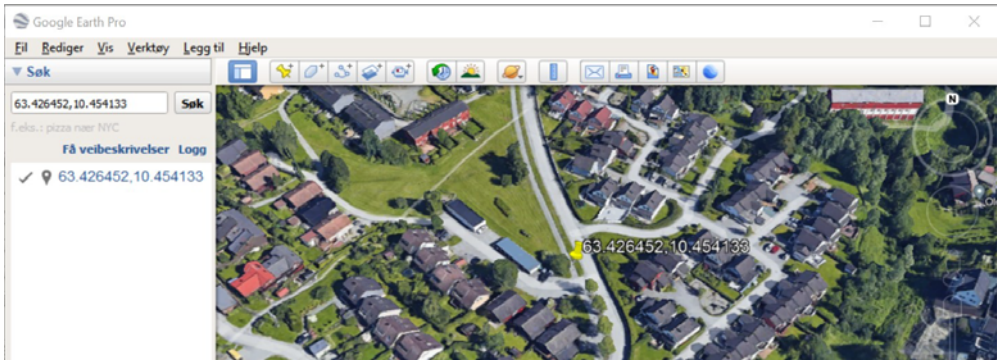
Ved å skrive inn bredde- og lengdegrader i innboksen øverst til venstre i vinduet, vil man kunne "forflytte seg" til det angitte stedet.

Lengde- og breddegrader legges inn som vist under:

63.426452,10.454133
Breddegrader [°],Lengdegrader [°]

Her er selvfølgelig rekkefølgen viktig.

Legg merke til at en minst bør ha med 4 siffer etter komma, gjerne 5 eller 6, høyde godtas ikke.



6.5.2 Plotting av en trase i Google Earth

Dersom man ønsker å plotte en trase i Google Earth kan man benyttet et programmeringsspråk kalt "Keyhole Markup Language" (KML) som er et språk 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 rekkefølgende på koordinatene er omvendt av det man 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>
```

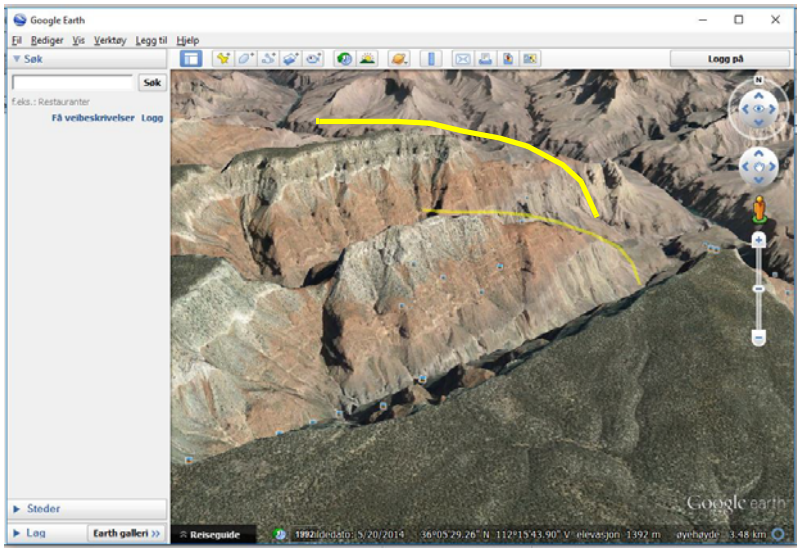
```
    <PolyStyle>
      <color>7f00ff00</color>
    </PolyStyle>
  </Style>
  <Placemark>
    <name>Absolute Extruded</name>
    <description>Transparent green wall with yellow outlines</description>
    <styleUrl>#yellowLineGreenPoly</styleUrl>
    <LineString>
      <extrude>0</extrude>
      <tessellate>0</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 og bli lastet inn i programmet.

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).



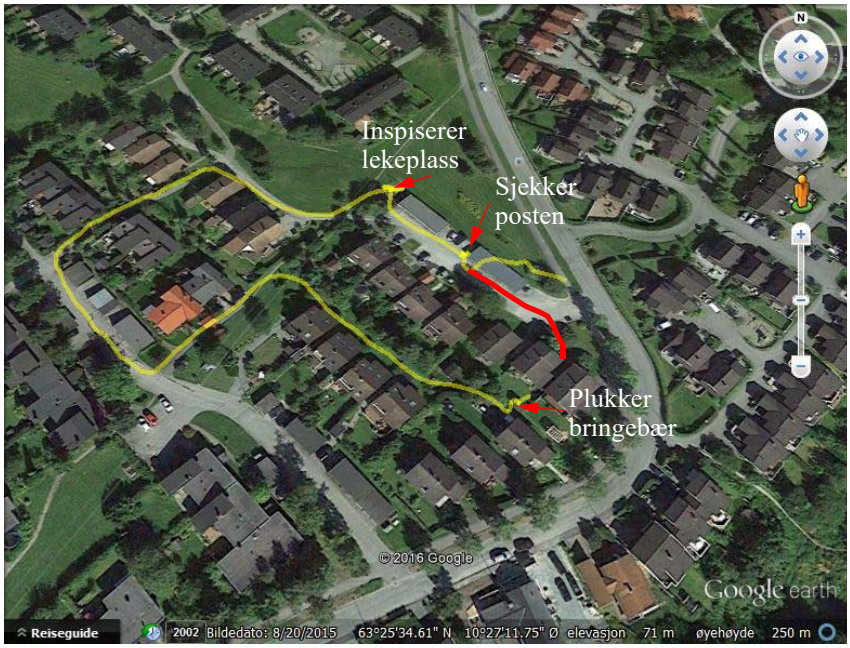
Dersom man ønsker å vise hvor man har gått en tur så kan det være upraktisk å måtte vise absolutt høyde. Høydemålinger kan ha store avvik slik at en kan oppleve at traseen går mange meter over eller forsvinner under bakken til tross for at man hadde begge beina på jorda. I slike situasjoner kan det være 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.

6.5.3 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.***

6.6 Bruk av Streamlit for presentasjon av resultater

<Her kommer det forhåpentlig mer>

Se: https://streamlit.io/

7 Sensorer

I dette kapittelet skal vi beskrive aktuelle sensorer og hvordan vi kan koble disse til mikrokontrolleren. 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

Pin #	Port # /funksjon	Shield/Sensor	Pin #	Port # /funksjon	Shield/Sensor
1	AREF		15	D6~	ENV DRDY Temp, Hum
2	A0	PROTO Temp. Sensor/NTC	16	D7~	ENV DRDY Pressure
3	A1		17	D8~	Power “vakthund”
4	A2	ENV LYS	18	D9~	
5	A3	Batterinivå	19	D10~	
6	A4		20	D11~	ENV SDA I^2C GPS SDA I^2C
7	A5		21	D12~	ENV SCL I^2C GPS SDA I^2C
8	A6		22	D13~	
9	D0~	PROTO Temp. DS18B20	23	D14~	
10	D1~		24	RESET	
11	D2~		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.	https://no.rs-online.com/web/p/thermistor-ics/1793885/	NOK 32,02
DS18B20	Temperatur u/vann	https://no.rs-online.com/web/p/sensor-development-tools/2049893/	NOK 81,35
Leverandør: RFRobot			
SEN0237-A	Oppløst oksygen	https://www.dfrobot.com/product-1628.html	$ 169,00
DFR0300-H	Konduktivitet (K=10)	https://www.dfrobot.com/product-1797.html	$ 79,90
SEN-0161	pH sensor	https://www.dfrobot.com/product-1025.html	$ 29,50
SEN-0189	Turbidimeter	https://www.dfrobot.com/product-1394.html	$ 9,90
SEN-0165	ORP-sensor	https://www.dfrobot.com/product-1071.html	$ 89,00
Totalt	Kurs =10,63 (13.10.22)		$377,30

7.1 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.

7.1.1 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 går opp, 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 Coefficien*t, dvs. at resistansen avtar med økende temperatur.
- PTC – *Positive Temperatur Coefficient*, dvs. at resistansen øker med økende temperatur.

7.1.2 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[62] sammen med et plastisk bindemiddel.

En forenklet sammenheng mellom resistansen (R_{NTC}) og temperaturen (T) kan uttrykkes som:

$$R_{NTC} = Ae^{B/T} \tag{7.1}$$

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 rundt denne referansetemperaturen antas B-verdien å være tilnærmet konstant ($B_{25/85}$ – B-verdien er tilnærmet konstant innen området 25°C til 85°C).

Vi kan da sette opp følgende to ligninger:

$$R_{NTC} = Ae^{\frac{B_{25/85}}{T}} \tag{7.2}$$

62. Sintring betyr at metallpulver knyttes sammen ved hjelp av oppvarming, men uten å smelte.

$$R_r = Ae^{\frac{B_{25/85}}{T_r}} \tag{7.3}$$

Ved å eliminere *A* fra disse uttrykkene, kommer vi fram til følgende sammenheng, løst med hensyn til resistansen R_{NTC} i NTC-resistoren:

$$R_{NTC} = R_r \cdot e^{\left(\frac{B_{25/85}}{T} - \frac{B_{25/85}}{T_r}\right)} \tag{7.4}$$

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*. Temperaturen angis i grader Kelvin.

Siden det ofte er B for området 25° – 85°C som er oppgitt kan en lett oppleve at man havner på siden av det spesifiserte området, siden vi ofte ønsker å måle i området 0° – 25°C. Siden vi linearisere og kalibrerer sensoren trenger ikke dette bety så ny for vår anvendelse.

NTCLE400E3103H – NTC Thermistor 10kΩ

Fra databladet[63] for *NTCLE400E3103H* finner vi følgende: R_{25} er referansemotstand (R_r) ved 25 °C (T_r = 298 K):

ELECTRICAL DATA AND ORDERING INFORMATION

R_{25} (Ω)	R_{25}-TOL. (± %)	$B_{25/85}$ (K)	$B_{25/85}$-TOL. (± %)	SAP MATERIAL AND ORDERING NUMBER (1)(2)		
				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 7.1 Datablad for NTC-motstand NTCLE400E3103H, 2,2 – 100 kΩ (Epoxy type)

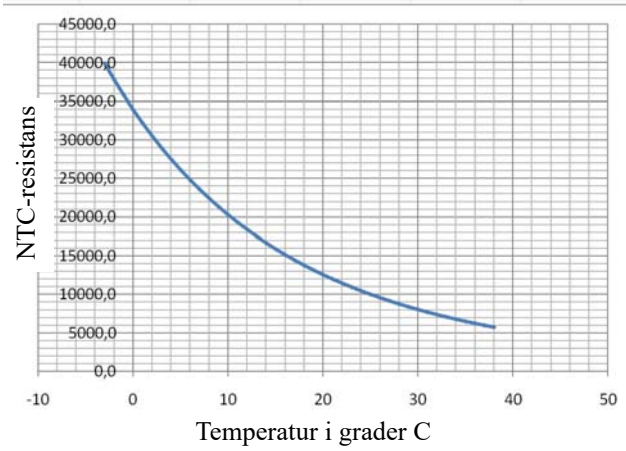
Med disse dataene kan vi skrive:

$$R_{NTC} = 10\text{k} \cdot e^{\left(\frac{3977}{T} - \frac{3977}{298}\right)} \tag{7.5}$$

hvor $B_{25/85}$ = 3977 (NTCLE400E3103H – 10 kΩ) og referansetemperaturen T_r = 298 K.

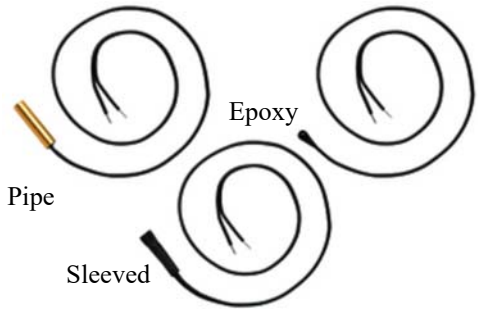
63. Databladet er hentet fra: https://docs.rs-online.com/6544/0900766b8167e65c.pdf

Dersom vi beregner verdier for R_{NTC} i temperaturområdet 25° – 85°C, får vi følgende graf:



Figur 7.2 NTC motstand som funksjon av temperaturen 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* (τ), 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.



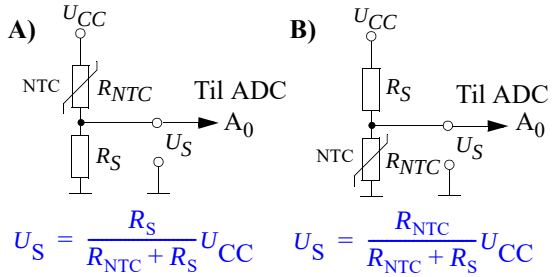
Response time (1):		
NTCLE400...Epoxy	≈ 7	s
NTCLS100...Sleeve	≈ 15	
NTCLP100...Pipe	≈ 10	

I databladet finner vi at temperaturresponsen 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, hvilket vil være katastrofalt for målingen.

Oppkobling mot ADC



Spenningsdeler for konvertere variasjon i resistans til variasjon i spenning.

Siden grensesnittet til mikrokontrolleren krever en spenning, kobles NTC-motstanden i serie med en motstand (R_S) som vist i figuren til høyre, 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 antydet på figuren over 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.

I vårt tilfelle er NTC-motstanden plassert 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 ev. kan legges inn i programmet i mikrokontrolleren.

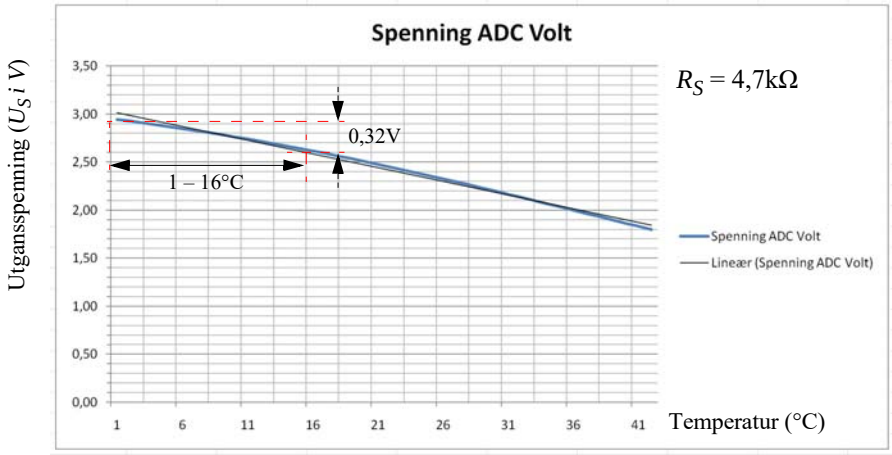
Vi setter:

$$R_{NTC} = 10\text{k} \cdot e^{\left(\frac{3977}{T} - \frac{3977}{298}\right)} \tag{7.6}$$

inn i ligningen:

$$U_S = \frac{R_{NTC}}{R_{NTC} + R_S} U_{CC} \tag{7.7}$$

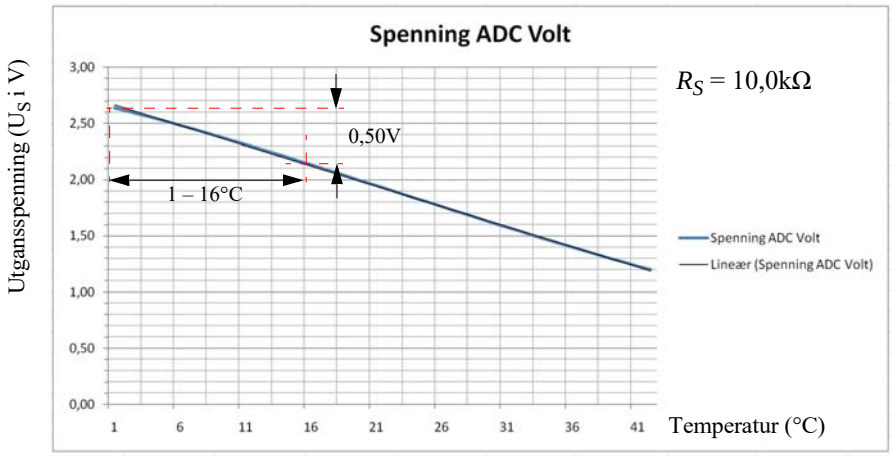
... og vi kan beregne U_S som funksjon av temperaturen for ulike verdier av seriemotstanden, R_S. I figuren under har vi modellert spenningen U_S som funksjon av temperaturen i området 0 – 42°C med en seriemotstand på R_S = 4,7 kΩ:



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 V.

Selv om kurven buer noe så er den relativt lineær i området 1 – 16°C som er det temperatur-området som sannsynligvis er mest aktuelt ved måling i havet langs kysten. I det nevnte temperaturområde gir et spenningssving på ca. 0,32V, dvs. 21,3 mV pr. °C.

Dersom vi velger en seriemotstand R_S = 10kΩ ser kurven blir mer lineær over hele området fra 1 – 42°C.



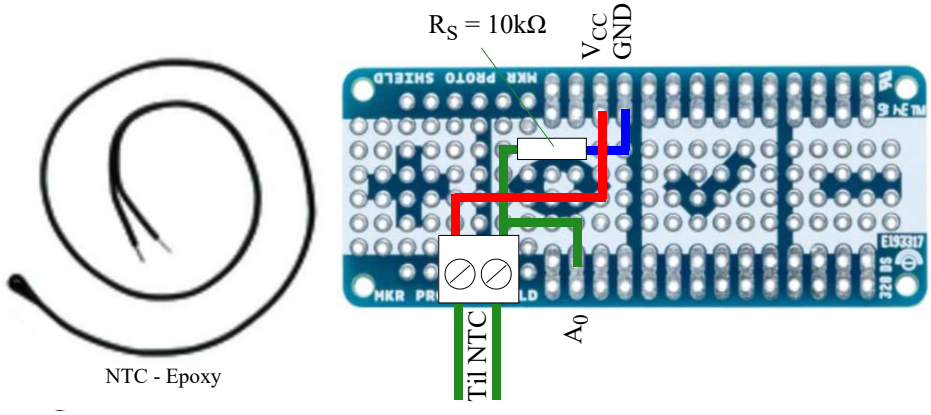
Vi registrerer at avvikene ved endene av området er mindre med 10kΩ enn ved 4,7kΩ. Dessuten er spenningssvinget i området 0 – 42°C lik U_{diff} = 1,08 V. Temperaturområdet 1 – 16°C gir et spenningssving på ca. 0,50V, dvs. 33,3 mV pr. °C.

Vi ser at 10kΩ gjør at kurven blir mer lineær og gir et større spenningssving som utnytter det dynamiske området til ADC'en bedre. Likevel ser vi at spenningssvinget er svært beskjedent.

Fra avsnitt 3.1 side 31 husker vi at MKR NB 1500 har inntil 12 bit oppløsning og at forsyningsspenningen og sannsynligvis referansespenningen til AD-konverteren er 3,3V. Det betyr at hvert bit tilsvarer et spenningssving på 3,3V/4096 = 0,8 mV dvs. ca. 42 bit pr. °C som gir en teoretisk oppløsning på 0,024 °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 gjør vi på et prototyp kort.



Sensor-kabelen er 400 mm, men kan skjøtes. Man må imidlertid være klar over at en skjøt lett kan lekke sjøvann, hvilket vil påvirke målingene sterkt. Sørg derfor at skjøten ligger tørt og godt over vannlinjen.

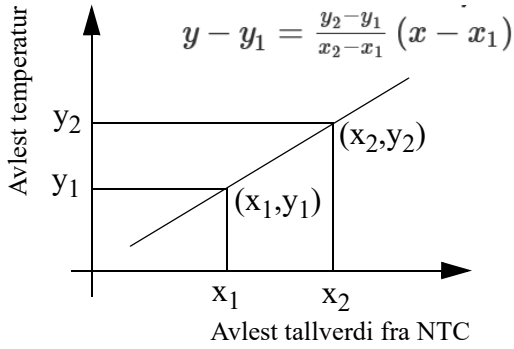
Det neste vi må gjøre er å kalibrere sensoren.

Kalibrering av temperatursensoren

Denne beregningen antar 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°C) dersom man velger motstandsverdien i seriemotstanden lik referansemotstanden. Under denne betingelsen kan følgende metode benyttes med tilstrekkelig nøyaktighet:

1. Bruk et begerglass med vann. Bruk vann med to ulike temperaturer, for eksempel 5°C og 15°C. Sett gjerne et stor glass med vann i kjøleskapet og bruk dette når vannet er avkjølt til å blande vann med ulike temperaturer.
2. Fyll et begerglas med det kalde vannet fra kjøleskapet og stikk sensoren ned i vannet. Vent 1 minutt før du foreta målingen.

3. Bruk et termometer å mål "virkelig" temperatur i det kalde vannet (y_1), les også av tallverdien som kommer fra 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°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:

$$y = ((y_2 - y_1)/(x_2 - x_1)) * (x - x_1) + y_1 \quad (7.8)$$

Hvis $k = (y_2 - y_1)/(x_2 - y_2)$ (stigningskoeffisienten), kan vi skrive ligningen slik:

$$y = k\,(x - x_1) + y_1 \quad (7.9)$$

Sett verdiene inn i formelen over og finn et uttrykk for y (temperatur i °C) som funksjon av tallverdien hentet fra AD-konverteren som måler spenningen fra NTC-motstanden (x).

I vedlegg E.2.5 finner du et program som du kan bruke når du skal finne måleverdiene fra AD-konverteren.

Bruk av kalibreringsprogrammet

1. Monter PROTO-kortet på "ryggen" av MKR NB 1500. Pass på at det plasseres riktig vei.
2. Hent programmet for kalibrering av temperatursensoren: `Testprogram-MKR-Temp-Kalibrering-1` fra vedlegg E.2.5 side 211.
3. Installer og kjør programmet.
4. Dypp sensoren i kaldt vann (f.eks. 6°C) sammen med et ordinært termometer. Åpne monitoren og les av den digitale verdien. Les av temperaturen på termometeret.

5. Dypp sensoren i lunkent vann (f.eks. 16°C) sammen med et ordinært termometer. Åpne monitoren og les av den digitale verdien. Les av temperaturen på termometeret.
6. Sett inn de digitale verdiene og temperaturene i programmet:

```
// ------------- Legg inn verdier for kalibrering av temperatur ---------------//

int D_T_Lav = 338;  // Digital verdi for spenningen på A0, registert lav temperatur)
int D_T_Hoy = 710;  // Digital verdi for spenningen på A0, registert 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
```

7. 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.
8. Testprogrammet er nå klart til å bli inkludert i hovedprogrammet.

7.1.3 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.

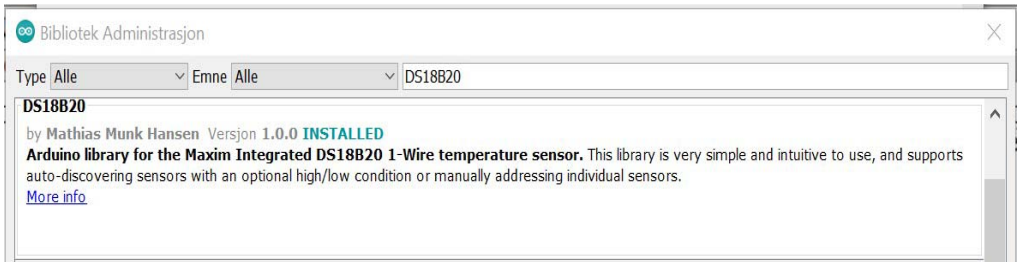
Sensorer har et måleområde på –55 til 125°C med en oppløsning på fra 9 – 12 bit.

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).

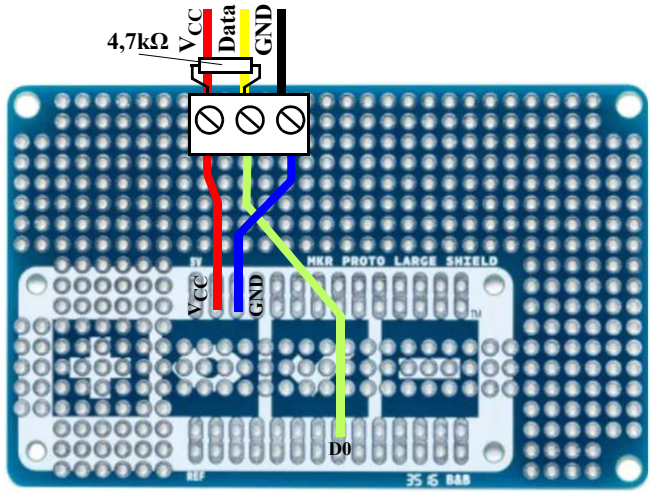
DS18B20: Gå til: *Skisse* > *Inkluder bibliotek* > *Administrer bibliotek* og skriv DS18B20 i søkefeltet og velg biblioteket DS18B20 skrevet av ***Mathias Munk Hansen***:



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

Oppkobling

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 E.2.6.

```
/*
 * 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;
```

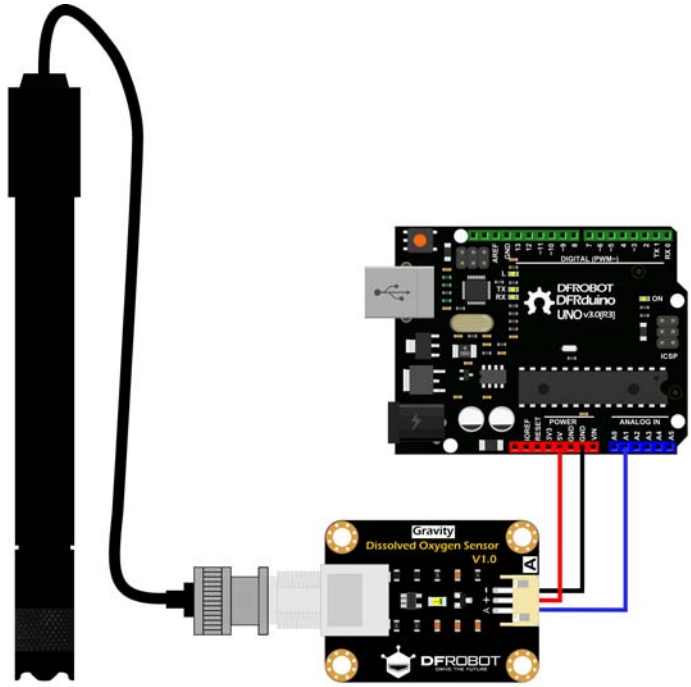
```
void setup()
{
  Serial.begin(115200);
  selected = ds.select(address);

  Serial.println("Testprogram-DS18B20-Temp-1");
}

void loop()
{
  Serial.print("Temperature is: ");
  Serial.print(ds.getTempC());
  Serial.println(" C");

  delay(2000);
}
```

7.2 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[64].

64.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, skyll 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 fingernegler 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[65].

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

7.2.1 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.

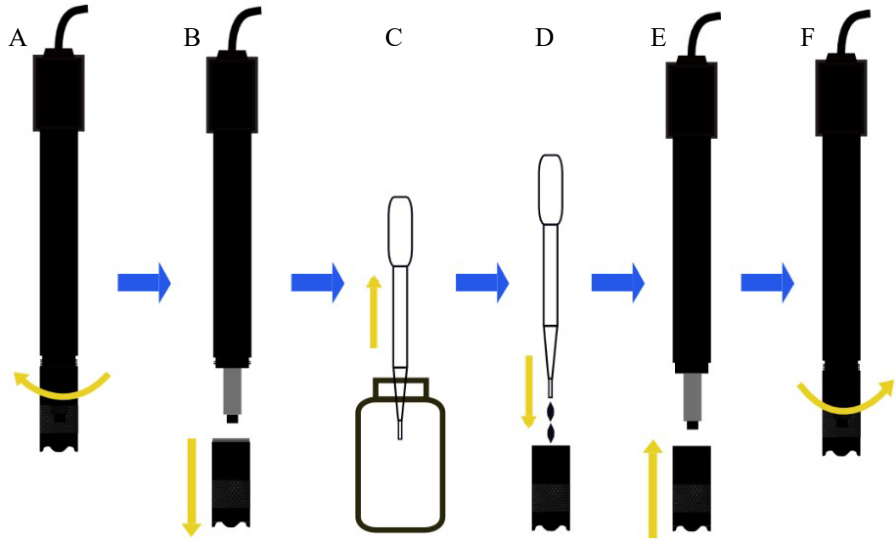
65.https://atlas-scientific.com/probes/industrial-dissolved-oxygen-probe/

Framstilling av NaOH oppløsning:

For å lage en 1 molar oppløsning av et stoff, tar man en mengde av stoffet tilsvarende atomvekten i gram og tilsetter 1 liter vann. Atomvekten for NaOH er 23 (Na) + 16 (O) + 1 (H) = 40. Dvs. for å lage 1dL 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:



A. Skru av beholderen med membranen foran på proben ...

B. ... og ta den helt av

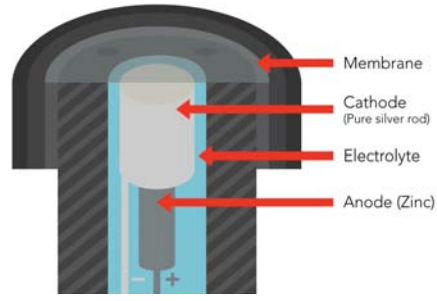
C. Bruk en pipette for å fylle ...

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 skrus på, dermed vet vi at den er full

Pass på å sette lokk på resten av NaOH oppløsningen slik at man unngår at CO_2 i lufta forringer oppløsningen.



Virkemåte[66]

Forrest i membranbeholderen sitter en semi-permeabel membran (PTFE) som kun slipper gjennom oksygen (O_2). Inne i membranbeholderen sitter proben som har en platina-katode og en sink-

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.

7.2.2 Kalibrering av proben.

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
}
```

66.https://atlas-scientific.com/blog/how-do-dissolved-oxygen-probes-work/

```
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-punkskalibrering 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 magnetisk omrører 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);
}
```

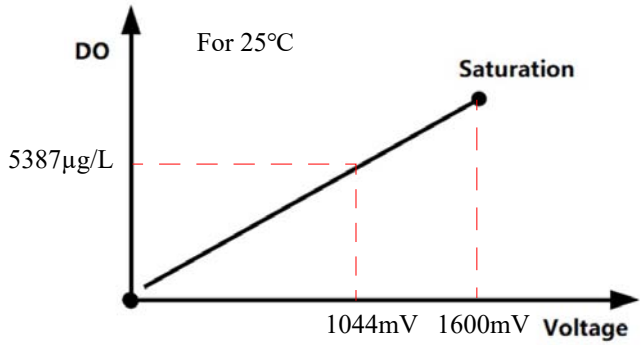
Etter at kalibreringsverdiene fra målingene er lagt inn i programmet og det er kompilert og kjørt, kan vi få ut et resultatet som ligner på denne tabellen:

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 i µg/L. Ved å dele på 1000 får vi mg/L.

Tolkning av ett-punktskalibrering

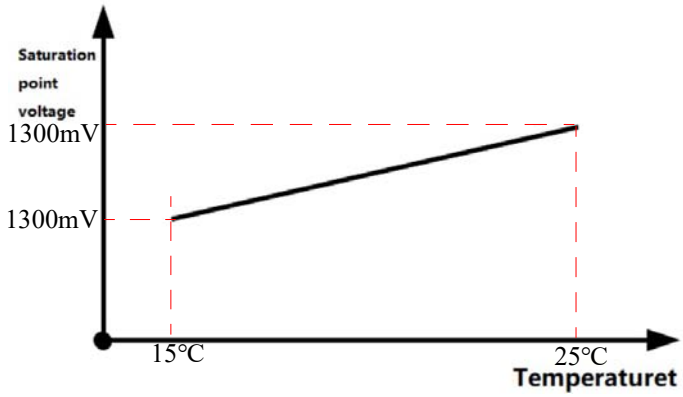
Figuren under viser det målte metningspunktet for oppløst oksygen i vann ved temperaturen 25°C.



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.

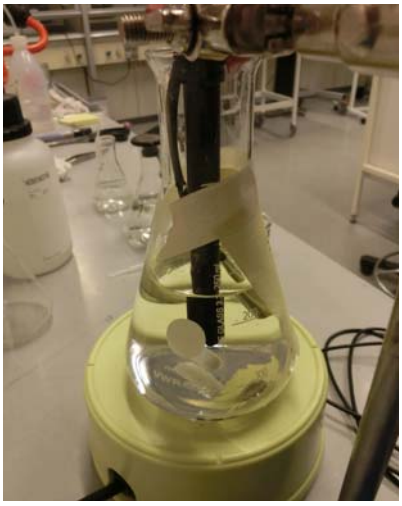


Tabellen under gir en oversikt over metningspunktene som funksjon av temperaturen. Tabellen er integrert i programmet som beregner oppløst oksygen ut fra målt temperatur og spenning.

T ℃	DO mg/L	T ℃	DO mg/L	T ℃	DO 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 til 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 (Na_2SO_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 fordamper. 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.

7.3 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[67].

Sensoren er spesielt beregnet til å måle ledningsevnen i saltvann med et måleområde opp til 100mS/cm (milli Siemens). Sensoren fungerer med spenningskilder fra 3,0 – 5,0V. For å unngå polarisering av proben tilføres den en vekselspenning (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.*

67.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:

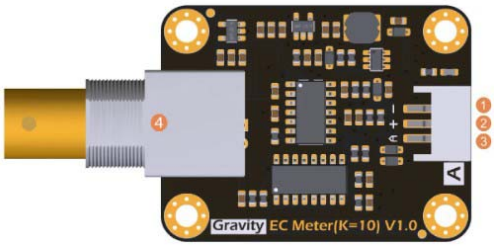
- Type: For laboratoriebruk
- Måleområde: 10 – 100mS/cm
- Cellekonstant: 10 ±2
- Temperaturområde: 0 – 40°C
- Levetid > 0,5 år under normal bruk, som ikke betyr kontinuerlig bruk.
- Kabellengde: 1 meter

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.



Num	Label	Description
1	-	Power GND(0V)
2	+	Power VCC(3.0~5.0V)
3	A	Analog Signal Output(0~3.2V)
4	BNC	Probe Connector

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 signalbehandlings-kortet 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[68]. *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

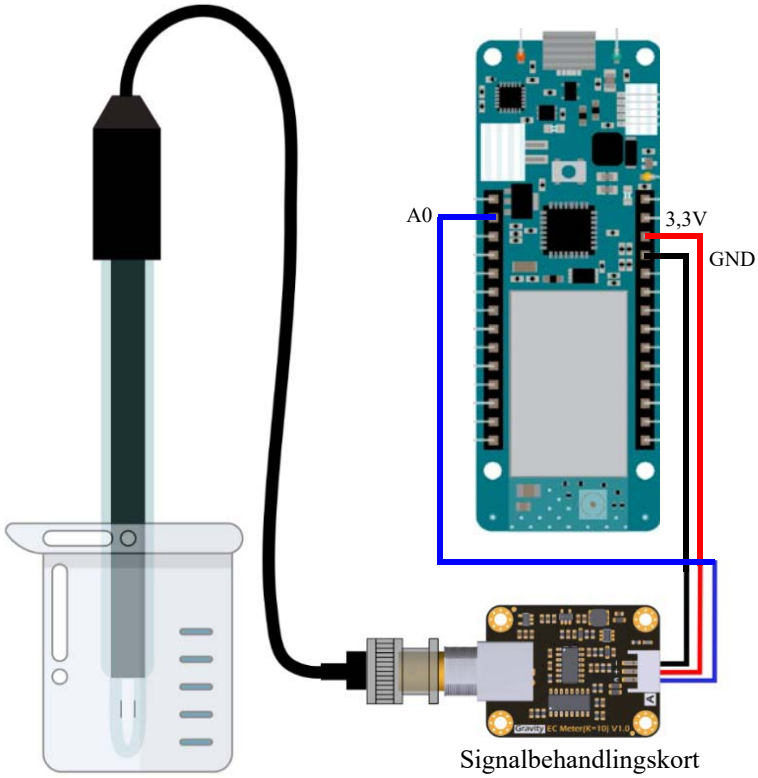
Programvare:

- Arduino IDE
- Last ned og installer DFRobot_EC10 biblioteket[69]
- Eksempelkode for kalibrering (se side)

68.https://atlas-scientific.com/kits/conductivity-k-10-kit/
69.https://codeload.github.com/DFRobot/DFRobot_EC10/zip/master

Oppkobling

Figuren under viser en typisk oppkobling. Signalbehandlingskortet forsynes med 3,3 V og signal-utgangen 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.



Signalbehandlingskort

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 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. Figuren under gir en grafisk framstilling av denne sammenhengen.

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

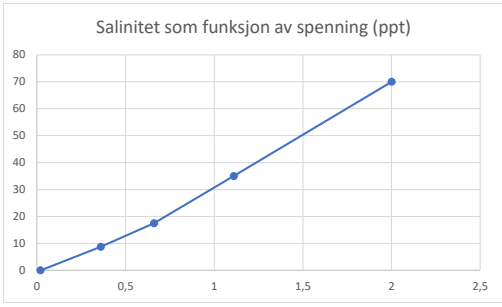
Målinger

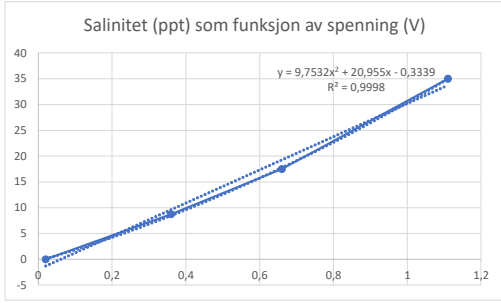
Det er foretatt målinger av spenningen med ulike konsentrasjoner av saltblandinger, angitt i ‰ (promille).

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

I figuren under til venstre har vi tatt med alle målingene og vi ser at kurven er relativt lineær ned til 15 ‰, derifra 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 tre laveste målepunktene. Da kan vi sette opp en matematisk sammenheng som er meget god.





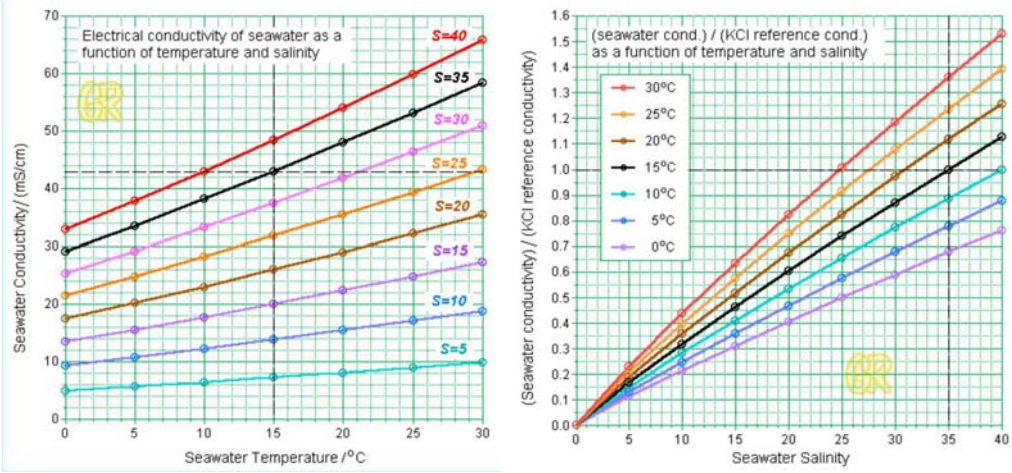
$$\text{Salinitet} = 9{,}7532 * \text{Spenning}^2 + 20{,}955 * \text{Spenning} - 0{,}3339\ [‰] \qquad (7.10)$$

Måling av havvann hentet ute ved Trondheim biologiske stasjon (TBS) er målt til:

Saltholdighet i vannprøve fra Trolla (Trondheimsfjorden) 34,3 ‰ @ 20°C.

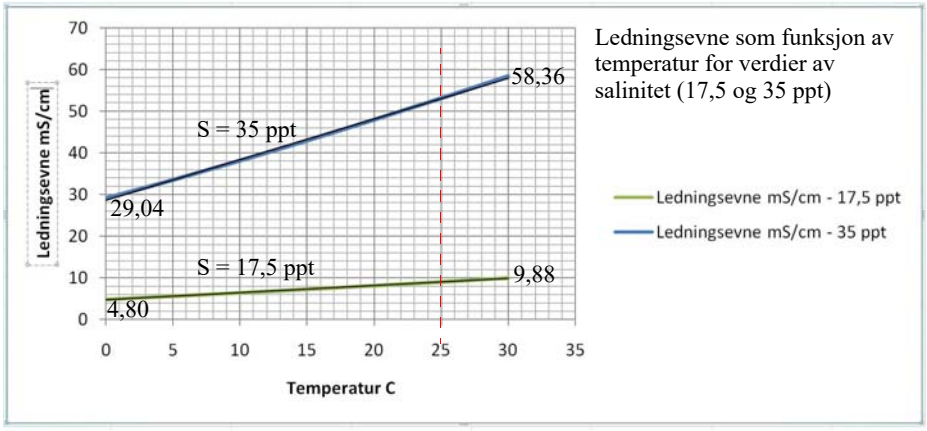
Det vi må huske på er at vi har målt saltholdigheten i sjøvannet referert til standarder laget med NaCl som skal være riktig ved 25°C. Målingene er imidlertid gjort ved 20°C. Siden den aktuelle temperaturen er vesentlig lavere så bør vi kompensere for endring i målt salinitet mht. temperaturen.

Ved bruk av kalkulatorer som disse kan man lage kurveskarer som viser sammenhengen mellom de ulike parametrene som vist på figuren under[70]:



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[71].

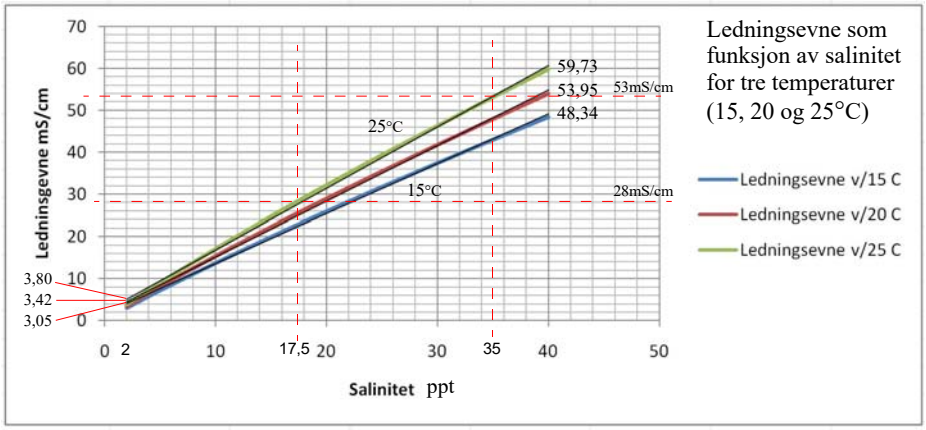
I vårt tilfelle, hvor vi ønsker å måle saliniteten i sjøvann nær verdien 35 ‰, så kan det være interessant å se hvordan ledningsevnen endrer seg med temperaturen under forutsetning av en nær konstant salinitet.



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.

70. http://www.grabovrat.com/handbook/handbookH08g1.html
71. Standard vannprøven er laget ved å å blande ut 32.4356 gram KCl i a1 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.

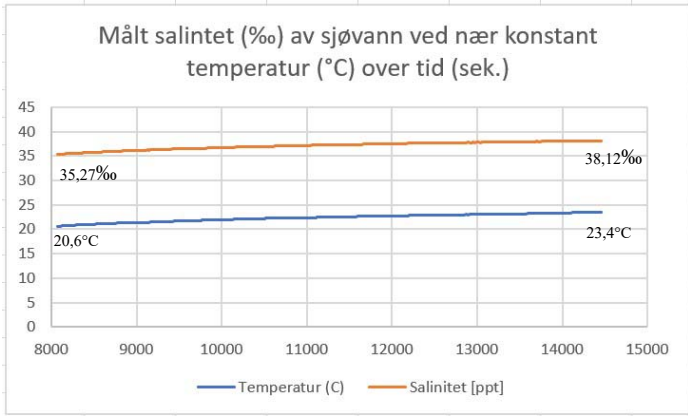


La oss si at vi gjør en kalibrering ved to verdier av salinitet (f.eks. 17,5 ppt og 35 ppt) ved en temperatur (f.eks. 25°C). Er det da mulig å finne en forenklet formel for salinitet som funksjon av en digital måleverdi innen et begrenset temperatur- og salinitets-område. Klarer vi å finne en slik sammenheng 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 C.1 side 193. For ytterligere fordypning av måling av saltholdighet se referanse [2].

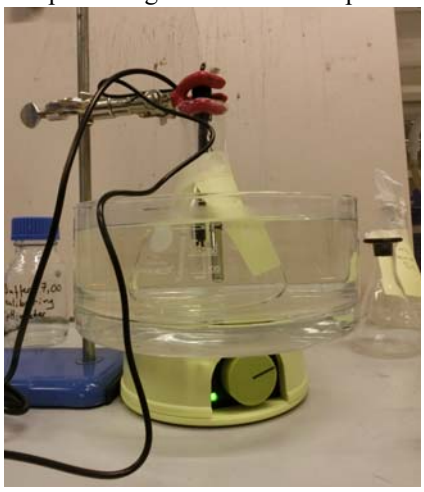
Måling av salinitet som funksjon av temperatur

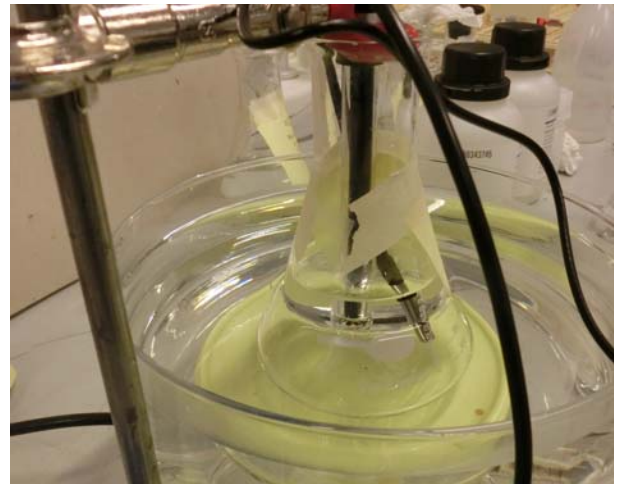
Vi har gjort målinger for å finne ut hvordan måleverdiene av saliniteten til sjøvann endrer seg med temperaturen. Figuren under viser hvordan saliniteten endrer seg over tid med så og si uendret temperatur. Den endringen vi ser skyldes sannsynligvis endring av temperatur i laboratoriet evt. endring på grunn av magnetrøreren som avgir litt varme.



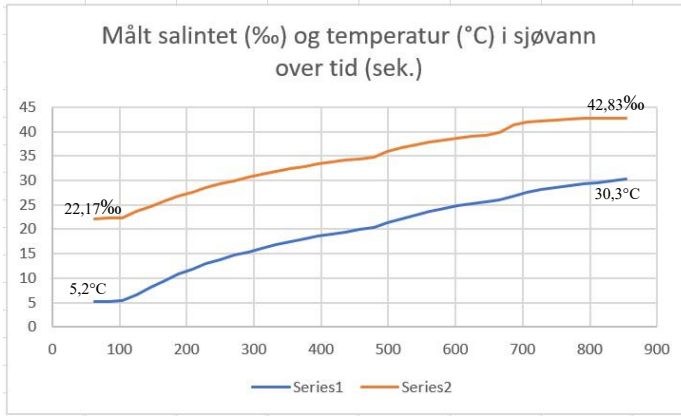
Målingen i figuren over viser at temperaturen har endret seg over måleperioden på ca. 106 minutter. Målingene er gjort med salinitetsproben og sensoren DS18B20 og samlet inn med en Arduino MKR NB 1500. Målingen er gjort i ca. 200 mL saltvann i en erlenmeyerkolbe hentet fra Trolla. Målingen er gjort under konstant omrøring.

For å kunne registrere målt salinitetsverdi som funksjon av temperaturen på en effektiv måte har vi lagt å sette saltvannsprøven i et vannbad med en temperatur på nærmere 40°C. Vannbadet ble satt på en magnetrører som vist på bildet under.

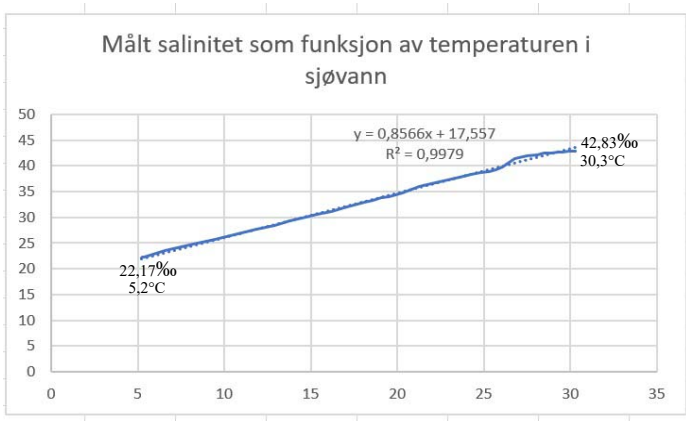




Effekten kommer bedre fram i den neste målingen der sjøvannet er nedkjølt til drøyt 5°C. Dernest ble varmet opp til ca. 30°C, ved å sette kolben i varmt vann.



Figuren over viser den forsterkede trenden av at 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 temperatur på ca. 40°C.



Ved å legge inn trendlinjen ser vi at sammenhengen mellom temperatur og målt salintet er ganske så lineær (R^2 = 99,79), med følgende foreslåtte sammenheng:

$$\text{Salinitet} = 0{,}8566 * \text{Temperatur [C]} + 17{,}557 \text{ for } [5{,}2 – 30{,}3°C] \quad (7.11)$$

Den tilsvarende formelen for nær konstant temperatur er:

$$\text{Salinitet} = 0{,}9836 * \text{Temperatur [C]} + 15{,}095 \text{ for } [20{,}6 – 23{,}4°C] \quad (7.12)$$

Vi antar at målingen gjort over et større temperaturområde er mest nøyaktig så vi bruker den som utgangspunkt for å lage kompensasjon.

Vi tar utgangspunkt i vår salinitetsstandardene framstilt av NaCl-blandingen (på 8,75, 17,5, 35 og 70 ‰ @ 25°C). Med utgangspunkt av disse målinger fant vi en sammenheng mellom målt spenning og salinitet i ligning (7.10) side 141:

$$\text{Salinitet} = 9{,}7532 * \text{Spenning}^2 + 20{,}955 * \text{Spenning} – 0{,}3339 \text{ [‰]} \quad (7.13)$$

Med utgangspunkt i denne gjorde målte vi salinitet i sjøvann til 34,3 ‰ ved 20°C.

Denne ønsker vi å korrigere mht. temperaturen der vi antar at korreksjonen er 0 ‰ @ 20°C. Som korreksjon tar vi utgangspunkt i ligning (7.11) side 145:

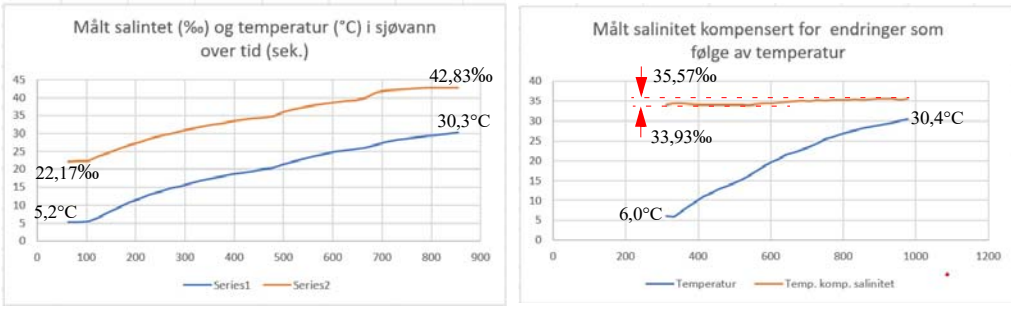
$$\text{tempSalinitet} = 0{,}8566 \text{ [‰/°C]} * \text{Temperatur [°C]} + 17{,}557 \text{ [‰]} \quad (7.14)$$

Vi finner korreksjonen ved å trekke fra

$$\text{deltaSalinitet} = 0{,}8566 \text{ [‰/°C]} * (\text{Temperatur}) \text{ [°C]} + 17{,}557 – 34{,}3 \text{ [‰]} \quad (7.15)$$

$$\text{Temp. kompensert salinitet} = \text{Salinitet} – \text{deltaSalinitet} \quad (7.16)$$

Figuren under viser korrigert og ukorrigert salinitetsmåling:



Eksempelkode for konduktivitetssensor (salinitet)– DFR0300-H

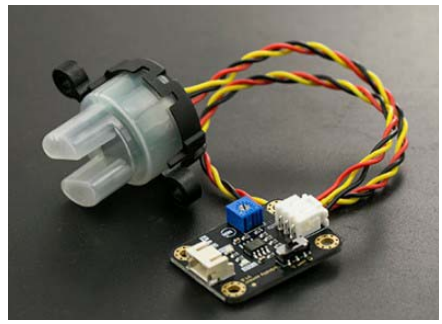
Programmet skissert under er utviklet uten bruk av biblioteker og kan derfor lastes opp og kjøres på MKR NB 1500. Ved hjelp av målingene omtalt foran kompenseres for temperaturvariasjoner. Det antas at innen det aktuelle temperaturområdet er vannets salinitet uforandret.

```
// Salinitetsmåler_DFR0300_H
// Programmet måler og beregner Salinitets verdien til en væske på bakgrunn av
// kaliberingsverdier 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 NTU-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 temeraturendringer
  long sensorValue  = 0;   // Les av analog inngang fra pinne A0:
  float UD_mid      = 0;   // Middelverdi av målt spenning
  float UD_mid_volt = 0;   // Middelverdi av målt spenning

  for(int i=0; i<100; i++)
  {
     sensorValue = sensorValue + analogRead(A0);
  }
  UD_mid = float(sensorValue/100);          // Finner middelverdien

  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);

  //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);
```

7.4 Sensor for måling av turbiditet (SEN-0189)[72]



Turbiditetssensoren måler vannkvaliteten ved å registrere nivået 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 ut som om denne 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

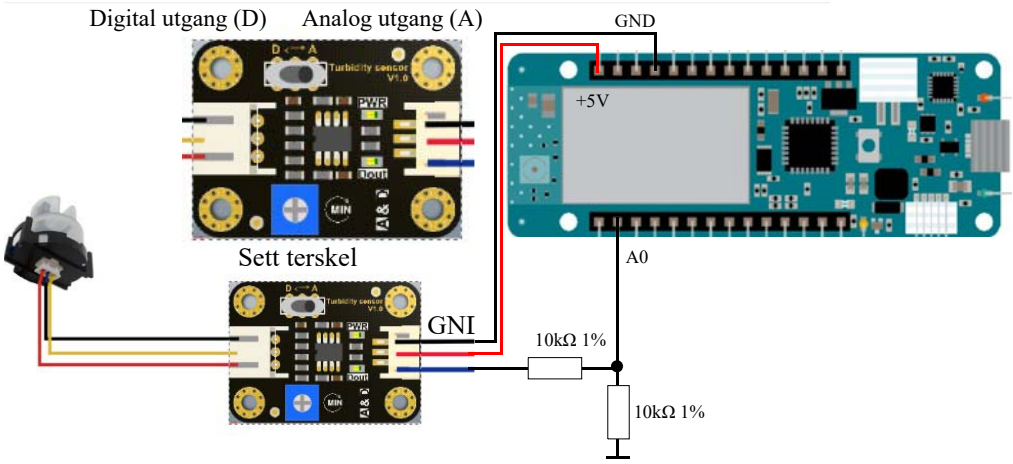
Noen av de viktigste parameterne:

- **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.

72.https://wiki.dfrobot.com/Turbidity_sensor_SKU__SEN0189

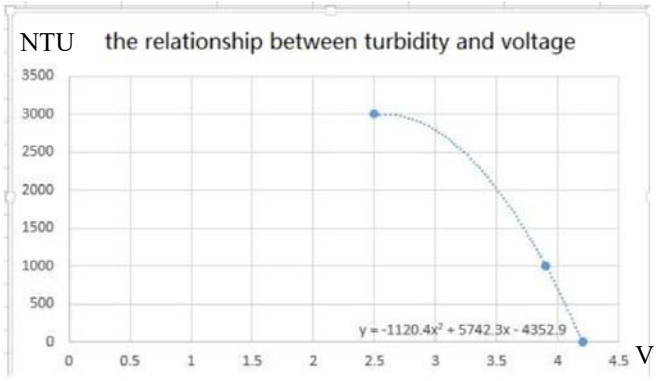
Oppkobling:

Figuren under viser oppkoblingen til en Arduino UNO. En lignende løsning kan brukes for tilkobling til Arduino MKR. En må imidlertid sørge for at analoginngangen til MKR ikke overskrider 3,3V. Dette gjør vi ved hjelp av spenningsdeler som halverer spenningen. Deretter må vio passe på å doble spenningen når vi bruker den i programmet vårt.



Ved hjelp av en bryter D/A kan man 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 spenningen og turbiditeten temmelig ulineær. Det er kun for lave verdier av turbiditet 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.

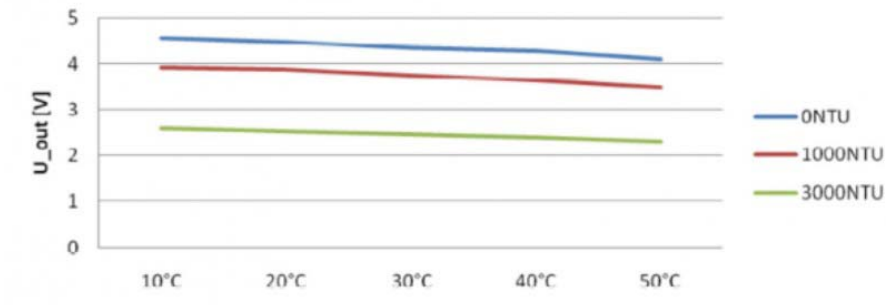
Måleenheten NTU står for Nephelometric Turbidity Units og er vanlig brukt i USA. 1 NTU tilsvarer ca. 1 mg/L. En annen enhet som også er brukt i USA er Jackson Turbidity Unit (JTU) hvor 1 JTU = 1 NTU. I Europa er det vanlig å bruke enheten Formazin Nephelometric Units (FNU).

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.

NTU = 500 NTU = 50 NTU < 0,5

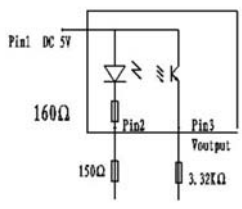
Måleresultatet er også avhengig av temperaturen som vist i figuren under.

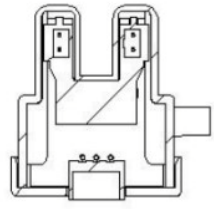


Vi registrerer at spenningen kan falle så mye som 0,5V når temperaturen øker fra 10 – 50°C. Dette skyldes sannsynligvis endringer i elektronikken og ikke forandringer i turbiditet.

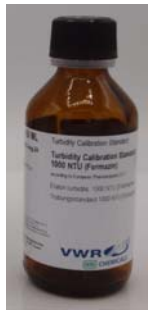
Koblingsskjema og mekanisk utforming

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.





Testbetingelse og kalibrering



Dersom sensoren settes i rent vann med en NTU < 0,5 så skal utgangspenningen være 4,1V±0,3V. Andre standarder kan lages ut fra en 400 NTU standard[73]. 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 og blande ut denne til man får en serie med standardløsninger. Slike kan bl.a. kjøpes hos VWR[74]. 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 sulfate og 100 g/L hexamethylenetetramine med destillert vann.

Men som sagt turbidimeteret SEN-0189 er ikke et ekte turbidimeter, men heller et colorometer som både måler summen av spredning og absorbans.

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 sette opp AD-konverteren til 12 bit, noe vi har anledning til hos Arduino MKR NB 1500.

Eksempel programmet 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:

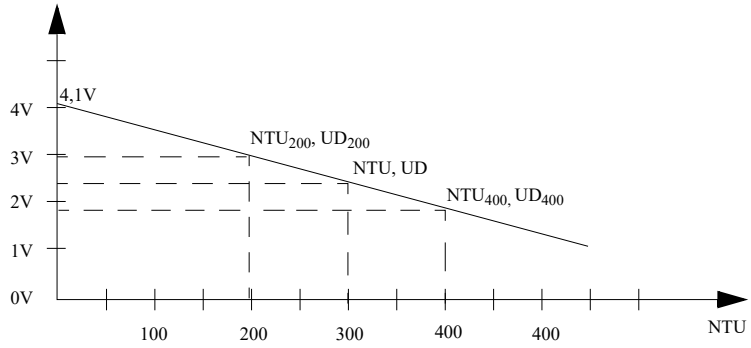
- ... at det er en lineær sammenheng mellom spenningen fra sensoren og turbiditet målt i NTU-enheter.
- ... at turbiditeten for rent destillert vann har en NTU-verdi lik null og gir en spenning på 4,1V
- ... at spenningen levert av mikrokontrolleren er stabil nok

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 Farmazin med NTU-verdier på 200 og 400 under kalibreringen

73. Beskrivelse av hvordan man lager en standard oppløsning på 400 NTU. https://www.boquinstrument.com/how-to-make-turbidity-standard-solution
74. 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:

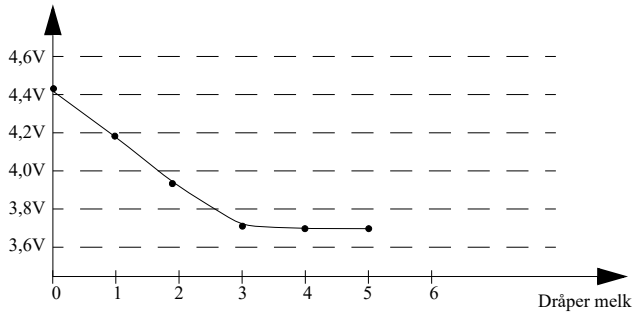
$$NTU = (NTU_{400} - NTU_{200})/(UD_{200} - UD_{400})\ (UD - 4{,}1\ [V]) \tag{7.17}$$

NTU er uten benevning.

Enkel innledende test uten kalibrering

Som en enkel innledende test kan man bruke litt lettmelk 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 avtalt 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. men siden kyvettekammeret er ganske lite så vil en dråpe utgjøre et ganske stort sprang i turbiditet.

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 ugjennomsiktig 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 gjennomfør en måling og noter spenningen
3. Fyll boksen med kalibreringsvæske med NTU 200 gjennomfør en måling og noter spenningen

4. Skriv de fire verdiene 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. 4,3 V.
6. Mål en ukjent prøve å se om det virker rimelig.

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
// kaliberingsverdier 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;   // Middelverdi av målt spenning
```

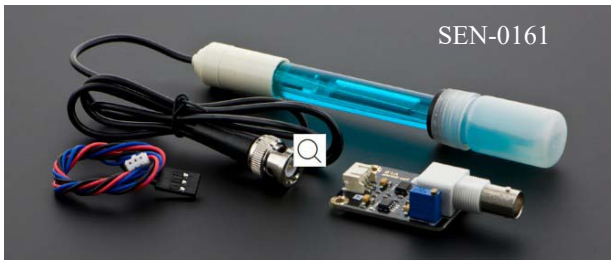
```
  float UD_mid_volt = 0;   // Middelverdi av målt spenning

  for(int i=0; i<100; i++)
  {
     sensorValue = sensorValue + analogRead(A0);
  }

  UD_mid = 2.0 * float(sensorValue/100);          // Finner middelverdien
  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);
}
```

7.5 Sensor for måling av pH (SEN-0161)[75]

pH-sensoren SEN-0161 (bildet under til venstre) er spesielt tilpasset bruk med Arduino og gir en nøyaktighet på ±0,1pH ved 25°C. Det forutsettes da at instrumentet er kalibrert. Arbeidsspenningen er 5,0V. Sensoren er en laboratoriesensor *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[76]),

75.https://www.dfrobot.com/product-1025.html
76.https://www.dfrobot.com/product-1074.html

se bildet under til høyre. Det finnes også en versjon 2 (SEN-0161-V2[77]) av denne proben som også fungerer for arbeidsspenninger ned til 3,3 V. Denne anbefaler imidlertid bruk av et bibliotek som ikke er kompatibelt 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 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°C.

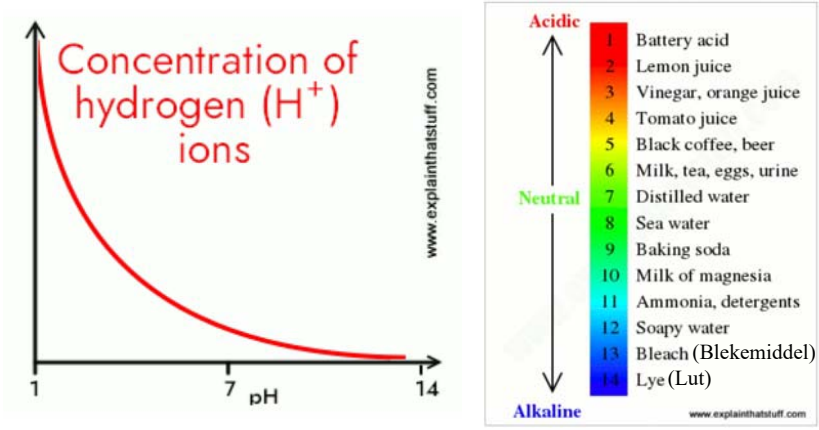
Måling av pH

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 H^+ ioner, mens baser danner OH^- ioner. Ioner er atomer eller molekyler som har avgitt elektroner (f.eks. H^+) eller mottatt elektroner (f.eks. OH^-) og er dermed ladet.

pH er en forkortelse for *potens* of *Hydrogen* (eller *power* of *Hydrogen*). Hvis konsentrasjonen av 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 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}.[78] Vi ser at pH angis som $-\log(10^{-pH}) = pH$.

77. https://www.dfrobot.com/product-1782.html
78. 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 var 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 µmol/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[79].

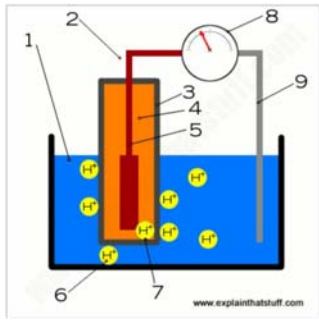


Når vi skal måle pH-verdien i en væske, utnytter vi kunnskapen om at jo syrligere væsken er jo flere H^+ ioner inneholder den.

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

79.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 potensialforskjell en 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 egentlig er 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.

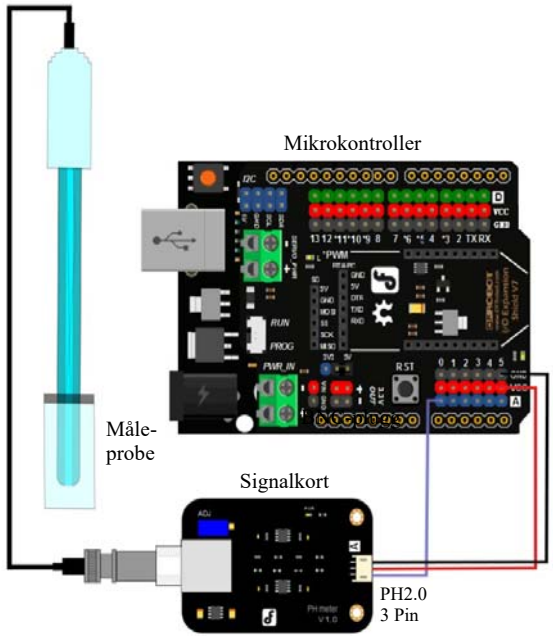
Vi må også være klar over at utvekslingen av ioner er temperaturavhengig, noe det er viktig å kompensere for.

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

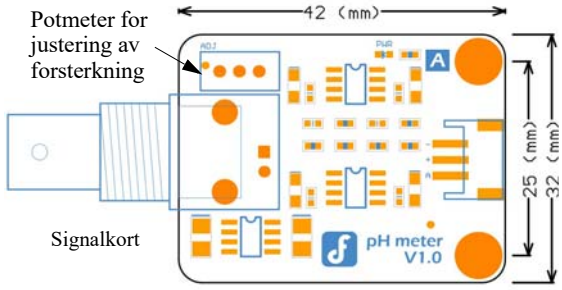
Selve proben kobles til signalkortet (signalkonverteringskortet) med en BNC-plugg. Legg merke til at utgangssignalet fra signalkortet føres inn på en analog inngang hos mikrokontrolleren (her A0).



Pass på følgende:

- Unngå at signalkortet blir fuktig. Et fuktig kort vil endre måle-betingelsene 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 til-koblet signalkortet 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

Spesifikasjoner:



- Forsyningsspenning: 5.0V
- Probekontakt: BNC
- Signalkontakt: PH2.0 – 3P
- Målenøyaktighet: ±0.1@25°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 oppløsning på pH 7,0 og en på pH 4,0 evt. pH 9,18
- 1 x Gravity 3pin Sensorkabel
- 1 x Testoppløsning

Koble opp som vist på figuren over og fjern hetta foran på proben.

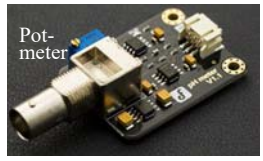
Det er viktig å bruke en stabil spenningskilde på signalkortet. Normalt vil man hente den fra Arduino-kortet, men denne kan være beheftet med støy fra mikrokontrolleren, eller som hos MKR NB 1500 så vil den ikke normalt være tilgjengelig med mindre man lager den selv siden MKR benytter 3,3 V. En power 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 eksempelkoden 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 pH 7,0 ±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
   ```

8. Tørk av proben og stikk den ned i standard bufferløsningen med pH 4,0 og les av verdien med programmet.
9. 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.



10. 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 "forurenses" 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-sennsoren 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):

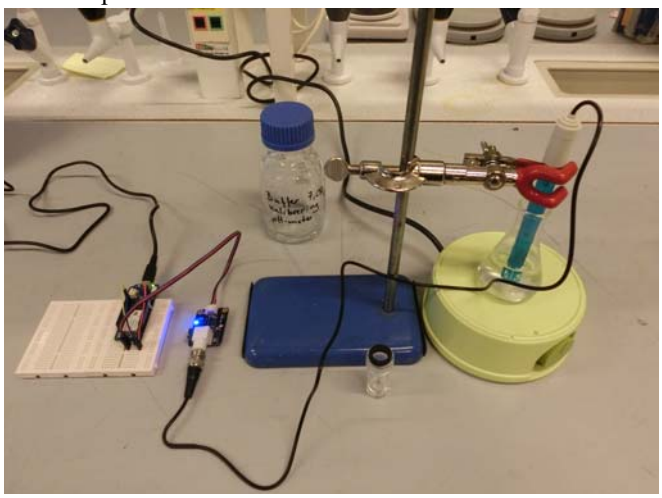
1. 0.1M Na_2HPO_4 (14.2g / l)
2. 0.1M HCl
3. 0.1M NaOH

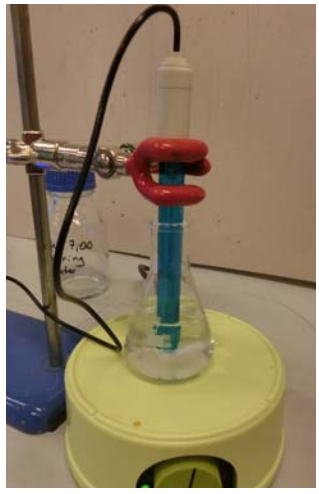
Ulike pH-verdier oppnås ved å blande disse i de mengder som vist i tabellen under

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 røremagnet 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 legge 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 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[80] under.

T (°C)	K_W (mol^2 dm^6)	pH
0	0.114×10^{14}	7.47
25	1.008×10^{14}	7.00
50	5.476×10^{14}	6.63
100	51.3×10^{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[81] til høyre viser dette.

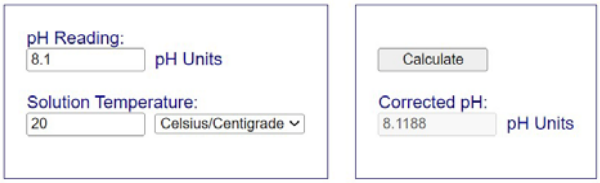
	0°C	25°C	50°C
Acid	2.01	2.00	2.00
Neutral (Water)	7.47	7.00	6.63
Basic	13.80	12.83	12.15

Dersom vi ønsker å inkludere en temperaturkorreksjon i programmet, så vil det være en fordel om sammenhangen 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.



80.https://www.westlab.com/blog/2017/11/15/how-does-temperature-affect-ph
81.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 ($\overline{V}_{\overline{pH}}$) brukes så til å beregne en verdi for pH etter følgende formel:

$$pH = 3{,}5 * \overline{V}_{\overline{pH}} + Offset \tag{7.18}$$

Der

$\overline{V}_{\overline{pH}}$	er den midlede potensialforskjellen målt mellom elektrodene
pH	er den beregnede pH-verdien
Offsett	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: F.4 side 245

Oppbevaring

Når sensoren ikke er i bruk skal toppen settes på proben. Toppen 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

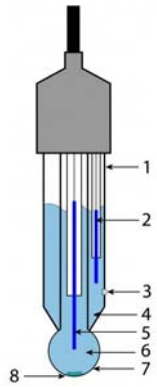
7.6 Sensor for måling av ORP (SEN-0165)[82]

ORP (Oksidation-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.*

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.*

Selve måleelementet er en ORP-kompositttelektrode laget av gull eller platina, som brukes til å måle oksidasjons-reduksjons-potensialet til oppløsningen.

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[83]. 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.

82.https://wiki.dfrobot.com/Analog_ORP_Meter_SKU_SEN0165_
83.https://www.phidgets.com/docs/PH/ORP_Sensor_Primer#How_they_work

8. Dersom måleelektroden er laget av sølvklorid, kan det lett oppstå litt bunnfall av AgCl i bunnen av glasskulen.

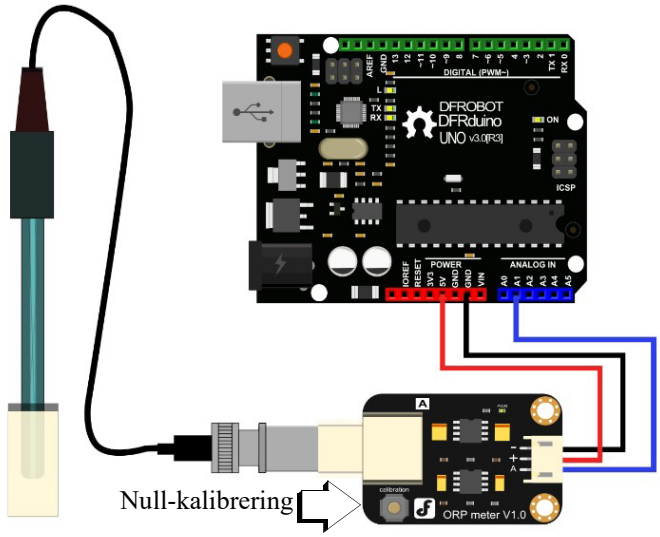
Som nevnt er pH- og ORP-sensorene ganske like, men elektrodene står gjerne i forskjellig elektrolytt (se også 7.5 side 155). Mens pH-sensoren er logaritmisk er ORP-sensoren lineær.

Spesifikasjoner:

- Spenningsforsyning: +5.00V
- Måleområde: –2000mV~2000mV
- Operativ temperatur: 5~70°C (Denne kan være kritisk i havvann om vinteren)
- Nøyaktighet: ±10mv (25°C)
- Responstid: ≤20sec
- 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.



Signalkonverteringskortet har en knapp for null-kalibrering.

Måleverdier i standardløsning med forskjellige temperaturer

Tabellen under til høyre viser hvordan den målte spenningen varierer med temperaturen ved bruk av en standard løsning på 3,5mol/L KCl.

222mV±15mV（25℃）
（3.5mol/L KCL）

℃	mV	℃	mV
10	242	30	215
15	235	35	209
20	227	38	205
25	222	40	201

Framstilling av standard løsning

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 3,5 mol = 261,1 g i 1 liter vann. Dvs. 26,1 g i 1 dL destillert vann.

Brukerveiledning

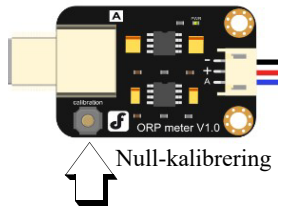
Vær oppmerksom på følgende:

- Det anbefales på det sterkeste å bruk 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 kompenser 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 eksempelkoden 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

Eksempelkoden finnes F.5 side 247

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 opplsning i 24 timer.
- Passivisering kan også skje dersom sensoren blir sterkt forurenset. I så fall må forurensningen fjernes med dertil egnet rensevæske.
- Etter et år med bruk kan det være på tide å skifte probe.

8 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/bla-hefteserie). Under overskriften "ROV (MauSea)" og fanen "ROV med trykk- og temperaturmåling, 2018"

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:

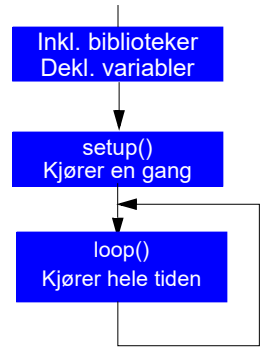
Tabell A.1 Komponentliste

Typebetegnelse	Type komponent	Leverandør	Pris inkl	Nettadresse
MKR NB 1500	Mikrokontroller	RS-online	929,84 (12.03.22)	https://no.rs-online.com/web/p/arduino/1763642
MKR GPS	GPS kort	RS-online	375,36 (12.03.22)	https://no.rs-online.com/web/p/shields-for-arduino/1894522/
MKR ENV	Miljø sensorkort	RS-online	402,19 (12.03.22)	https://no.rs-online.com/web/p/shields-for-arduino/2198445/
MKR Proto Shield	Protypkort for monte-ring sensorer	RS-online	111,75 (04.07.22)	https://no.rs-online.com/web/p/shields-for-arduino/1697587
GSM antenna	m/ UFL connector (868 Mhz)	RS-online	49,78 (12.03.22)	https://no.rs-online.com/web/p/shields-for-arduino/1697591
USB kabel	USB A til micro USB B kabel	RS-online	86,70 (12.03.22)	https://no.rs-online.com/web/p/usb-cables/8762403
DS18B20	Temperatursensor	Mouser Electronics	81,18 (04.07.22)	https://no.mouser.com/ProductDetail/DFRobot/DFR0198
MKR Proto Shield, large	Prototype kort, stort	RS-online	117,11 (17.08.22)	https://no.rs-online.com/web/p/shields-for-arduino/1697587
Li-Po batteri 2950 mAh	Batteri	Biltema	100,00 (17.08.22)	https://www.biltema.no/kontor-teknikk/batterier/oppladbare-batterier/oppladbart-icr18650-batteri-2950-mah-2000037909
Li-Po batteri lader	Batterilader	Biltema	70,00 (17.08.22)	https://www.biltema.no/kontor-teknikk/batterier/batteriladere/batterilader-til-batteri-18650-2000046111
Batteriholder 18650 ELFA nr. 169-14-431	Batteriholder	ELFA	34,73 (17.08.22)	https://www.elfadistrelec.no/no/batteriholder-smd-1x-18650-hullmontert-keystone-1043/p/16914431
Mini Micro JST 2.0 PH	Batteriplugg (100 stk)	Banggood	60,00 (02.09.22)	https://www.banggood.com/Excellway-100Pcs-Mini-Micro-JST-2_0-PH-2Pin-Connector-Plug-With-120mm-Wires-Cables-p-1147298.html?cur_wareho-use=CN&rmmds=search
Andre komponenter				
MKR Mem Shield	Kort for lagring av data på SD-kort	RS-online	207,28 (12.03.22)	https://no.rs-online.com/web/p/shields-for-arduino/1763644
Batteriplugg	Batteriholder med plugg	n00b	20,00 (18.04.22)	https://n00b.no/products/10-x-bbc-micro-bit-bat-teriholder-2xaaa?_pos=3&_sid=f9b19bdad&_ss=r

Vedlegg B Programmering av Arduino

Vedlegget gir noen grunnleggende tips til programmering av Arduino gir en oversikt over de mest brukte kommandoene. MKR NB 1500 programmeres på samme måte som UNO, men pinningen er selvfølgelig annerledes.

B.1 Programstruktur og bruk av funksjoner

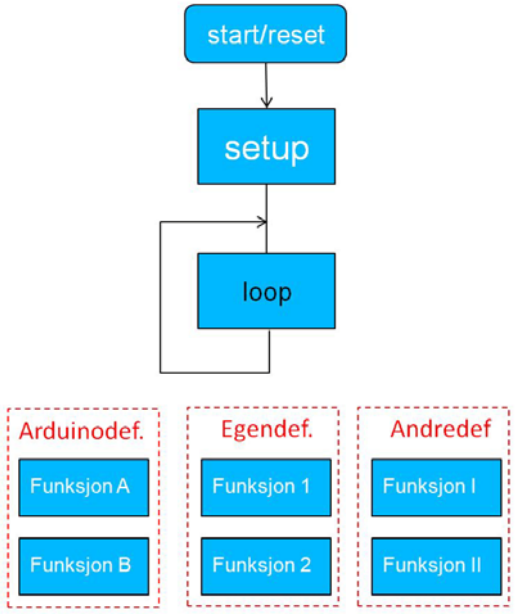


B.1.1 Programstruktur

Alle Arduino-programmer består av to hovedfunksjoner:

- `void setup()` som kjøres bare *en* gang hver gang programmet starter. Dette skjer etter at programmet er lastet opp, når man trykker på restart-knappen eller åpner monitoren.
- `void loop()` er en funksjon som gjentas så lenge Arduino'en har spenning.
- **Biblioteker og deklarasjon** av globale variabler plasseres gjerne helt i starten. ***Globale variabler*** kan brukes i alle funksjoner. ***Lokale variabler*** deklareres i den enkelte funksjon og kan kun brukes innenfor denne funksjonen. Lokale variabler kan også uten problemer gjenbrukes i andre funksjoner.
- Det er også vanlig å skrive ***egne funksjoner*** som gjerne legges på slutten av programmet, utenfor og etter `void loop()`-funksjonen(), men egendefinerte funksjoner kan i prinsippet legges hvor som helst utenfor `void setup()`- og `void loop()`-funksjonene.

```
#include <bibliotek> // Inkluderer spesielle biblioteker
Deklarer globale variable

void setup()
{
    // Koden i setup() kjører bare ved start
.... pinMode(<pin>, in/output);
}

void loop()
{
    // Deklarer lokale variable
    // Programlinjer

    funksjon1( ); // Kaller funksjon 1
...
}

void funksjon1()
{
   // Deklarer lokale variabler
    // kode
}
```



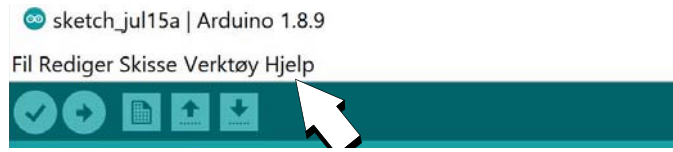
De to hovedfunksjonene i Arduino-programmene omsluttet av *klammeparenteser*. I `void setup()`-funksjonen initieres mikrokontrolleren og ev. sensorer som er tilkoblet, mens selve programmet legges under `void loop()`-funksjonen, som vist på figuren over.

`setup()` og `loop()` er *navnet* til funksjonene, mens de to klammeparentesene {} omslutter *kroppen til funksjonen* hvor selve programmet ligger.

I tillegg til disse to faste funksjonene så bruker vi funksjoner fra Arduino's eget bibliotek (Arduinodef. på figuren over), funksjoner som andre har laget (Andredef.) og vi kan lage våre egne funksjoner (Egendef.). Funksjoner som andre har laget og som de ønsker å tilby fellesskapet, samles ofte i *biblioteker*, som kan lastes ned, installeres og gjøres tilgjengelig for programmerere. For nærmere beskrivelse av hvordan man lager og bruker egne funksjoner, se avsnitt B.2.15 side 185.

B.2 Viktige kommandoer

Referansemanualen til Arduino C++ for bruk ved programmering av Arduino-kontrollerkort finnes på følgende nettadresse: http://arduino.cc/en/Reference/HomePage



Referansemanualen kan også nås via *Hjelp* på menylinjen i programeditoren (figuren over).

B.2.1 Initiering av dataoverføring til PC

Under uttestingen og visning av innhold av variabler kan det være praktisk at data leses tilbake til monitoren i Arduino editoren. Datahastigheten settes opp i setup-funksjonen med kommandoen: Serial.begin(9600); her er datahastigheten satt til 9 600 baud[84] (her ca. 9600 *bit* pr. sekund) som vist under:

```
void setup()
{
   Serial.begin(9600);
}
```

84. Baud rate – Angir hvor mange symboler som kan overføres pr. transmisjonsperiode. For et binært system vil baud raten tilsvare bit pr. sek. For mer komplekse former for modulasjon (f.eks. QPSK) vil hver transmisjonperiode kunne overføre flere bit (2, 4 eller flere), dog på bekostning av følsomheten for støy.

I noen tilfeller kan det være hensiktsmessig å øke overføringshastigheten. 115 200 tegn pr. sekund er vanlig å bruke. Pass på at mottakerhastigheten for monitoren er satt til samme hastighet. Dette gjøres nederst til høyre i monitoren som vist på figuren under.

Kommandoer for å skrive tilbake til PC – til monitoren i programeditoren:

Følgende kommandoer skriver en variabel eller en tekst tilbake til terminalvinduet (monitoren) i programeditoren.

```
Serial.print(a);            // Skriver variabelen a til en linje på skjermen,
                            // neste skrivekommando skriver på samme linje
Serial.println(a);          // Skriver variabelen a til en linje på skjermen og
                            // skifter linje (ln), neste skrivekommando skrives
                            // derfor på ny linje
Serial.println("Hallo");    // Skriver teksten Hallo til en linje på skjer
                            // neste skrivekommando skriver på ny linje
```

Det er også mulig å kombinere tekst og variabler i samme printkommando:

```
Serial.println("Trykk:",a);// Skriver teksten Trykk: til en linje på Arduino
                          // monitoren, etterfulgt av innholdet i variabelen
                          // a, og skifter deretter til ny linje
Serial.println(f, 2);     // Skriver desimalvariabelen f til terminal på PC
                          // med to desimaler
```

B.2.2 Kommentarer:

Kommentarer kan skrives hvor som helst og begynner med //

```
// Dette er en kommentar
```

Kommentarer blir fjernet under kompilering og overføres ikke til mikrokontrolleren. Kommentarene er derfor kun en hjelp for den som skriver programmet, og ev. andre som skal lese og forstå programmet senere.

Ønsker man å kommentere bort flere linjer etter hverandre kan dette gjøres slik:

```
/*
Alle tre linjene
vil bli betraktet
```

```
som kommentarer
*/
```

Midlertidig å kommentere bort programlinjer kan også være et nyttig hjelpemiddel under feilsøking.

B.2.3 Bruk av variabler

Bruk av variabler er en praksis som gjør programmering særdeles slagkraftig.

> *Vi kan betrakte variabler som oppbevaringssteder ("skuffer") for tekst eller verdier med navn og type. Innholdet vil som oftest være ukjent når vi skriver programmet, men vi reserverer plass til verdiene. Navnene på variablene bør gjenspeile hva de representerer slik at det blir lettere og lese og forstå koden.*

La oss deklarere variablene "`tempForskjell`", "`tempStart`" og "`tempSlutt`" som float (desimaltall). Vi kommer da til den andre viktige egenskapen:

> *Vi kan behandle og utføre beregninger på variablene uten at verdiene til variablene er kjente.*

```
tempForskjell = tempSlutt – tempStart;
```

Som variabelnavnene indikerer så ønsker vi å beregne temperaturforskjellen mellom start- og slutttidspunktet. Dette føres oss til den tredje viktige egenskapen:

> *Vi kan bruke variabler som lagringsplass for verdier over tid.*

Deklarasjon av lokale og globale variabler

I programspråkene C og C++ må alle variabler deklareres før de kan brukes. Deklarasjonene må inneholde *type* og *navn* på variabelen og gjøres gjerne i starten av programmet før void setup()-funksjonen. Variabler deklarert utenfor funksjonene blir ***globale variabler***, dvs. at variablene kan brukes i alle funksjonene og beholder innholdet i variablene uavhengig av hvor de brukes.

Deklarering kan også gjøres *innenfor hver* funksjon. Slike variabler gjelder da bare innenfor denne funksjonen og kalles ***lokale variabler***. Under er vist deklarasjon av de vanligste typer variabler. I dette tilfelle vil de bare gjelde innen funksjonen void-loop()-funksjonen:

```
void loop()
{
   Int a;          // deklarasjon av 16 bit heltallsvariabel (word)
   char b;         // deklarasjon av 8 bit karaktervariabel (byte)
   char c, d;      // deklarasjon av to 8 bits karaktervariabler (byte)
   float e;        // deklarasjon av variabelen e som et desimaltall f.eks. 1,65
                   // (32 bit, dobbel word)
   unsigned long f;// deklarasjon av 4 bytes heltallsvariabel f (32 bit)
                   // uten fortegn
   boolean g;      // deklarasjon av en boolsk variabel g
                   // som kan ha verdiene 1 eller 0, "sant" eller "usant"
   // Her følger resten av programkoden i funksjonen loop()-
}
```

Dersom vi deklarerer variabler i begynnelsen av void loop()-funksjonen så vil de bli redeklarert for hver runde i loopen, hvilket betyr at de vil miste verdien for hver gang funksjonen utføres.

B.2.4 Etablering og bruk av array[85]

Noen ganger har vi bruk for å lagre data i en tabell, eller å lage et oppslag. En slik tabell kalles gjerne et *array* i programmering. Et array kan være av en eller flere dimensjoner.

En-dimensjonalt array

Et en-dimensjonalt heltalls-array med navnet *tabell* (fritt valgt) med *n* elementer deklareres slik:

```
int tabell[n];
```

Vi kan også fylle det med verdier idet vi deklarerer det:

```
int tabell[n] {0, 1, 2, 3, 4, 5 ... , n-1};
```

Her har vi fylt arrayet med tallene 0 til n-1, det kunne selvfølgelig ha vært andre heltall. Elementene i arrayet nummereres fra 0 til n-1. Når man ønsker å lese ut verdien fra ett spesifikt element, f.eks. element nr. *i*, så kan man skrive dette slik:

```
int element_i = tabell[i];
```

To-dimensjonalt array

På samme måte kan man definere et to-dimensjonalt array med f.eks. desimaltall (float):

```
float tabell_2D[m][n];
```

Vi kan tenke på dette som en tabell med *m* rader og *n* kolonner, eller om vi vil, med *n* elementer i hver rad. Vi kan fylle tabellen med verdier slik:

```
float tabell_2D[m][n] {
                       {00, 01, 02, 03, 04, ... 0m-1},
                       {10, 11, 12, 13, 14, ... 1m-1},
                        ⋮
                       {n0, n1, n2, n3, n4, ... nm-1}
                      };
```

Når man ønsker å lese verdien fra ett spesifikt element, f.eks. element nr. i, j, så kan man skrive dette slik:

```
int element_ij = tabell[i][j];
```

B.2.5 Strukturer

Mens et *array* er en variabel som inneholder et sett av *elementer av samme type*[86] (f.eks. *enten* int *eller* float osv.), så er en *structure* en variabel som kan inneholde et *sett av elementer av ulike typer* (f.eks. *både* int *og* float med flere). La oss like godt se på et eksempel på hvordan vi kan definere en struktur[87].

85.https://www.tutorialspoint.com/arduino/arduino_multi_dimensional_arrays.htm
86.Et array kalles i noen sammenhenger også en *vektor.*

Definisjon av strukturer

I dette eksempelet skal vi definere en struktur som holder måledata fra sensorer *sensor_measure*, vi kaller denne typen struktur for *measurements*. I den forbindelse ønsker vi å lagre følgende informasjon om målingene[88]:

```
typedef struct measurements {
  char message[32];
  String source;
  int instance;
  float temperature;
  float humidity;
};
```

Hvor:

`typedef struct`	– Viser at dette er definisjon av en struktur
`measurements`	– Navnet vi har gitt til denne *typen* struktur
`char message[32];`	– En tekststreng med informasjon om målingen
`int instance;`	– Målenummer
`String source;`	– Kilden for målingen (f.eks. MAC-adresse til ESP32)
`float temperature;`	– Temperaturen målt i grader Celcius
`float humidity;`	– Relativ luftfuktighet målt i prosent

Nå har vi definert en type struktur som passer for vårt bruk. Nå skal vi bruke denne definisjonen til å *deklarere* våre spesielle målinger.

Deklarasjon av strukturer

Nå når vi har definert en type struktur som vi har kalt *Measurements*, så kan vi lage oss flere strukturer, dvs. samlinger av variabler, med ulike navn av typen Measurements:

```
Measurements packet1;
Measurements packet2;
```

Vi har valgt å lage variablene `packet1` og `packet2` av typen `Measurements`.

Tilordning av innhold medlemmene av strukturen

Vi ønsker nå å legge inn informasjon i de ulike *medlemmene* av strukturen. Dette kan vi gjøre slik for et array av karakterer:

```
packet1.message[32] = "Maaling med BME280";
```

eller for en *int* eller *float* gjør vi slik:

---

87. https://www.tutorialspoint.com/cprogramming/c_structures.htm
88. Det finnes flere måter å definere strukturer på som er like riktig, vi har valgt denne fordi den tydelig indikerer hva dette handler om: Definisjon av en type struktur (typedef struct { ... })

```
packet1.temperature = 23.4;
```

eller slik for *strengvariabler*:

```
packet1.source = String("Dette er en streng");
```

String-kommandoen kan også brukes til å konvertere et tall til en tekststreng. I dette tilfellet omformes et heltall (45) til et heksadesimalt tall (HEX) uttrykt som en "tekststreng", dvs. en streng med bokstaver og tall:

```
packet1.source = String(45, HEX);
```

Tilgang til innhold hos medlemmer av strukturer

Vi skal jo også kunne ta ut innholdet til ett eller flere medlemmer i en struktur og bruke det i ulike sammenhenger som f.eks. ved en utskrift. Dette kan vi gjøre slik:

```
Serial.print(packet.message);     // For et tekstarray
Serial.print(packet.instance);    // For et heltall
Serial.print(packet.source);      // For en streng
Serial.print(packet.temperature); // For en float
```

Vi kan også legge innholdet over i en enkel variabel slik:

```
float T = packet.temperature;
```

Ekstra slagkraftig blir bruken av strukturer når vi skal overføre et sett av variabler "pakket" i en struktur i argumentet i en funksjon.

B.2.6 Pause-kommando

Dersom vi ønsker at programmet skal ta en pause kan vi skrive følgende:

```
delay(1000);              //Stopper programmet i 1000 msek (1 sek)
delayMicroseconds(100);   //Stopper programmet i 100 µs, 0,1 msek (0,0001 sek)
```

Dette medfører imidlertid at heller ingenting annet kan gjøres i denne tiden. Har man ikke råd til å miste denne tiden, bør man finne andre løsninger, f.eks. bruk av `millis();` som vist i neste avsnitt.

B.2.7 Bruk av millis()

Dersom vi vil unngå at programmet stopper helt opp mens vi venter på at et tiden er inne for å utføre en handling, kan vi bruke `millis()` istedet for `delay().millis()` holder tiden i millisekunder fra vi slo på strømmen eller resatte programmet.

La oss anta at vi vil at programmet skal utføre en handling hvert 1000 ms (1 sekund) i tillegg til at programmet skal gjøre en del andre ting mellom disse tidspunktene. I så fall kan vi gjøre det slik:

```
unsigned long naaTidspunkt
```

```
setup()
{
  naaTidspunkt = millis();
  ...
}

void loop()
{
  ...
  if(millis()- naaTidspunkt > 1000)
   {
    // Gjør det som skal gjøres hvert 1000 ms

    naaTidspunkt = millis(); // Sett nytt nåtidspunkt
   }
 // Resten av programmet
}
```

Dette er f.eks. aktuelt der man ønsker å telle og vise sekunder på et klokkedisplay.

Vi definerer en variabel `unsigned long naaTidspunkt`; Grunnen til at vi definerer variabelen som type unsigned long er for å unngå at variabelen skal tildeles verdier som er utenfor maksimal størrelse til variabelen. Bruker vi en `int` vil denne nå grensen for område sitt i løpet av bare ca. 32 sek. For bruk av `long` vil dette ta i underkant av 25 døgn. Bruker vi `unsigned long` vil det ta over 49 døgn.

I setup()-funksjonen setter vi `naaTidspunkt` lik `millis()`. I `void loop()`-funksjonen tester vi om det er gått 1000 ms siden vi tok vare på `naaTidspunkt` sist. Dette gjør vi ved å trekke det sist oppdaterte `naatidspunkt` fra den løpende `millis()`. Dersom differansen er større enn 1000 ms, så utfører vi vår handling samtidig som vi oppdaterer `naaTidspunkt` til løpende `millis()`.

B.2.8 Aritmetiske og logiske operasjoner:

```
sum = a + b;     // Summen av a + b settes i variabelen sum
diff = a - b;    // Differansen av a - b settes i variabelen diff
prod = a * b;    // Produktet av a * b settes i variabelen prod
kvo = a / b;     // Koeffisienten av a / b settes i variabelen kvo
```

Variabelnavnene *sum, diff, prod* og *kvo* er bare valgt som eksempler og må deklareres.

Dersom man ønsker å endre innholdet i en av variablene, kan dette gjøres slik:

```
a = a + b;       // Summen av a + b settes i variabelen a
```

Dette ser ikke lengre ut som en fornuftig "ligning" slik vi kjenner den fra matematikken, og er det heller ikke. Her legges verdien i *a* til *b* før den så legges tilbake til *a*.

I tillegg til de aritmetiske operasjonene har vi også tre logiske funksjoner

```
&&   // Uttrykker logisk "og", brukes når to betingelser må være sanne
```

```
||    // Uttrykker logisk "eller", brukes når en av to betingelser må
      // være sanne for at hele betingelsen skal være sann
!     // Negasjon av logisk verdi, !sant er lik ikke sant
```

B.2.9 Adresser og pekere

Adresser:

I språket C og C++ har vi direkte tilgang til *adressen* i lageret hvor en variabel ligger. I noen tilfeller kan det være praktisk å kjenne adressen til en variabel. Dersom vi definerer en variabel, f.eks.:

```
float f;
```

... så kan vi finne adressen til variabelen som:

```
long adr_f = &f;
```

Ved å sette tegnet "`&`" foran variabelnavnet så får vi adressen der variabelen er lagret.

Pekere:

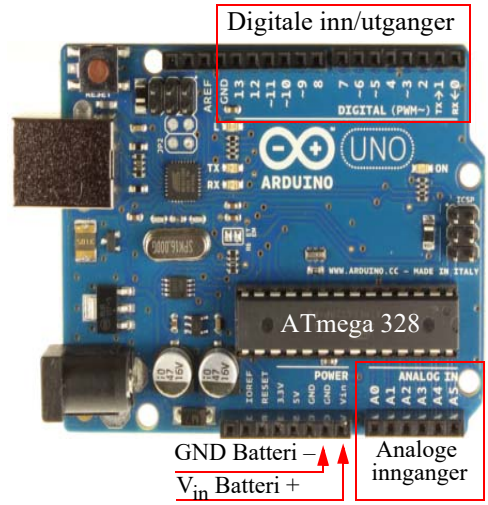
Vi kan også gjøre det motsatte. Vi kan definere en peker. En *peker* er definert som en adresse og angis med en stjerne "*". La oss deklarere følgende:

```
int resultat;      // Deklarerer en heltallsvariabel med navn resultat
int i = 6;         // i er et heltall som gis verdien 6
long *pek_i;       // *pek_i er en peker som kan peke på en adresse

*pek_i = &i;       // *pek_i tilordnes adressen til heltallet "i"
resultat = pek_i;  // pek_i angir innholdet av adressen *pek_i
                   // resultat inneholder verdien 6
```

Vi legger merke til at fjerner vi "*" foran pekeren så angir navnet verdien pekeren peker på.

B.2.10 Digitale porter

Definer digitale porter som inn- eller utganger:



En port er et fysisk terminal på kretsen som kan kobles til eksterne kretselementer, sensorer eller aktuatorer f.eks. LED. Disse kan enten være analoge eller digitale, innganger eller utganger. Siden en strømkrets må være sluttet, vil alle porter forholde seg til jord på kortet (dvs. minus hos strømforsyningen eller batteriet).

Kontrolleren ATmega 328 (Arduino UNO) har en rekke *porter*, 14 digitale (0 – 13) og 6 analoge (0 – 5). De digitale portene kan være innganger eller utganger. Hver port må derfor defineres som en inn- eller utgang. Dette gjøres gjerne i `setup()`-funksjonen:

```
void setup()
{
   pinMode(8,OUTPUT);       // Definerer port (pinne) 8 som utgang
   pinMode(7,INPUT);        // Definerer port 7 som inngang
   pinMode(6,INPUT_PULLUP); // Definerer port 6 som inngang med
                            // pullup-motstand, dette er nyttig for at
                            // inngangen ikke skal henge fritt (sveve)
                            // når den ikke er tilkoblet noe
}
```

Dersom en inngang defineres med *pullup-motstand* betyr det at utgangen er koblet til pluss på batteriet gjennom en "stor" motstand internt i kretsen. Det betyr at dersom porten ikke er tilkoblet noe annet på utsiden så vil den ha verdi 1 (dvs. 5V for Arduino og 3,3 V for ESP32).

Les fra og skriv til en digital port:

Digitale porter som er definert som utganger, kan enten settes til høy eller lav spenning. Digitale porter som er definert som innganger, kan lese av om spenningen på porten er høy eller lav. Dette gjøres med følgende kommandoer:

```
boolean boolsk;          // Definerer den boolske variabelen boolsk kan ha
                         // verdien 0 (usann) eller 1 (sann)
int heltall;             // Definerer en heltallsvariabel heltall,
                         // kan meget vel brukes for 0 og 1
void loop()
{
digitalWrite(8, HIGH);   // Setter port 8 høy (5V ev. 3,3 V på ESP32)
digitalWrite(8, LOW);    // Setter port 8 lav (0V)
boolsk = digitalRead(7); // Leser den digitale verdien på port 7
                         // og setter den inn i variabelen bolsk
heltall = digitalRead(6);// Leser den digitale verdien på port 6
```

```
                        // og setter den inn i variabelen heltall
}
```

Det er imidlertid vanligere å bruke heltallsvariabler (krever 2 byte) for å holde digitale verdier lest fra innganger. Man sparer imidlertid litt lagerplass dersom man bruker boolske variabler (krever 1 byte lagerplass).

B.2.11 Analoge porter

Les verdi fra en analog inngang og skriv til monitor (PC)

Syntaksen for lesing fra en analog inngang er slik:

```
<variabel> = analogRead(<analog port>);//Den analoge porten kan ha verdier fra
                                       // 0 – 5V hos UNO ev. 0 – 3,3V hos ESP32
```

Eksempel 1:

```
Int verdi;              // Deklarerer variabelen verdi
verdi = analogRead(0); //Digitale verdien fra analog inngang 0
                        //leses inn i variabelen verdi
```

Eksempel 2:

```
void loop()
{
   int pressure;             //Deklarerer pressure som en heltalls-variabel
   pressure = analogRead(1); //Leser av trykksensoren på AD-kanal 1
   Serial.println(pressure); //Skriv resultatet tilbake til monitoren (PC)
}
```

Legg merke til at *Serial.print();* skriver *uten* påfølgende linjeskift, mens *Serial.println();* skriver *med* påfølgende linjeskift.

Eksempel 3:

```
void loop()
{
   int digitemp;                 // Deklarerer digitemp som heltallsvariabel
   float anatemp;                // Deklarerer anatemp som float
   digitemp = analogRead(5);     // Leser av temperatursensoren på AD-kanal 5
   anatemp = digitemp * 500/1024;// Regner om til spenning og grader
   Serial.println(anatemp,2);    // Skriv resultatet tilbake til Arduino
                                 // monitoren (PC) med 2 desimaler
}
```

De analoge portene er tilkoblet en analog til digital omvandler (AD-konverter) som gjør om spenningen 0 – 5 V til en digital verdi fra 0 – 1023 (0000000000 – 1111111111). Dvs. AD-konverteren har 10 bits nøyaktighet. Tilsvarende kan ESP32 håndtere analoge spenninger fra 0 – 3,3 V og har AD-konvertere med en oppløsning på 12 bit.

For å regne om fra digital verdi (0 – 1023) til analog spenningsverdi (0 – 5V) benytter man følgende formel:

```
anatemp = 100 * digitemp * 5.0/1024;
```

`anatemp` er en float (desimaltall), mens `digitemp` er den digitale avleste verdien. Vi multiplise-rer med 100 siden en vanlig brukt temperatursensor TMP36 da vil gi resultatet direkte i grader C.

Siden den digitale verdien kan være 0 – 1023 skulle man tro at det ville være riktig å bruke 1023 og ikke 1024 i nevneren på brøken i uttrykket over. Imidlertid er det slik at den målte toppverdien for AD-konverter 1023 tilsvarer en spenning på 4,995 V (og ikke 5,0V), dermed vil det være rik-tigere å skrive 4,995/1023. Dette er imidlertid det samme som 5,0/1024 som er lettere å huske.

B.2.12 Sløyfer

Noen ganger har man behov for å gjenta en operasjon flere ganger, da er en *for-loop* egnet. Andre ganger vil man ha behov for å gjenta en operasjon så lenge en betingelse er oppfylt, i så fall kan man benytte en *while-loop*.

For-loop – For å gjenta mange like operasjoner (sløyfer)

For()-loop'en egner seg spesielt godt til å gjenta den samme operasjonen et bestemt antall ganger, kanskje med ganske små forandringer mellom gjentagelsene. En for()-loop kan skrives slik:

```
for(int x = 0; x < 100; x++)
{
// Her skrives koden som skal gjentas
}
```

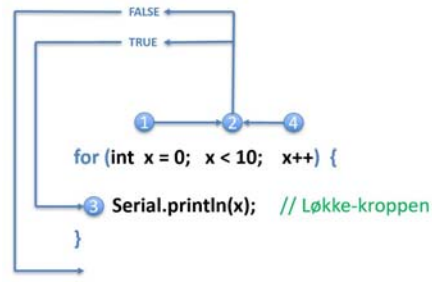
x er en heltallsvariabel (`int x`) som brukes som teller. Denne starter på verdien 0 (`int x = 0;`) økes med 1 (`x++`) for hver runde i loopen og stopper ikke før den når opp til 100 (`x < 100;`). De ulike utrykkene skilles med semikolon. Det som skal gjentas står innenfor klammeparentesene.

Eksempel:

```
for(int x = 0; x < 100; x++)
{
Serial.println(x);
}
```

Denne skriver ut tallene 0 – 99 til monitoren, ett tall for hver linje. Ønsker man å skrive ut tallene 0 – 100 kan man f.eks. skrive:

```
for(int x = 0; x =< 100; x++)
{
  Serial.println(x);
}
```



Figuren til høyre viser rekkefølgen av testen og inkre-mentering av løkkevariabelen[89].

89.Figuren er hentet fra en av Arne Midjos presentasjoner.

Legg merke til at linjen, `for(int x = 0; x =< 100; x++)` ikke avsluttes med semikolon, men etterfølges av { }.

While-loopen –
For å få programmet til å vente i en sløyfe til betingelsen ikke lenger er oppfylt

Det kan for eksempel være tilfelle når man venter på svar fra en sensor. While()-loopen kan skrives slik:

```
while (<logisk betingelse>)
{
// Her skrives koden som skal gjentas mens programmet venter
}
```

Legg merke til at linjen, `while (<logisk betingelse>)` ikke avsluttes med semikolon, men etterfølges av { }.

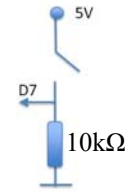
En while()-loop egner seg også for å hindre at en avlest bryter skal medføre et skred av avlesninger av bryteren.

Vanligvis ønsker vi at endring i tilstanden til en bryter skal medføre kun én hending.

I figuren til høyre ser vi en bryter som, når den trykkes inn, forbinder port D7 til +5V (logisk "1"). Når den ikke er trykket inn ligger port D7 til jord via en motstand på 10kΩ. Siden strømmen ut av port D7 til jord er tilnærmet 0, vil spenningen på D7 også være tilnærmet 0V (logisk "0").

While-loop for å unngå mange avlesninger

```
int trykkBryterPin = 7;

void loop()
{
while(digitalRead(trykkBryterPin) == 1)
  {
    trykkBryterVerdi = 1;
  }
If(trykkBryterVerdi == 1)
  {
    // Gjør noe en gang når trykkBryterVerdi er lik 1
    trykkBryterVerdi = 0;
  }
}
```



I programmet ser vi at avlesningen av D7 (`trykkBryterPin`) gjøres inne i argumentet til `while()`-loopen. Så lenge avlesningen er lik "1", så skal programmet bli værende i loopen og gjentatte ganger å sette `trykkBryterVerdi` til "1". Det er først når vi *slipper* bryteren at programmet forlater `while()`-loopen og går videre. Den påfølgende `if()`-setningen vil så sjekke om bryteren har vært trykket og utføre den ønskede handlingen. Siden vi ønsker at handlingen kun skal utføres en gang, må vi huske på å nullstille variabelen `trykkBryterVerdi` før vi forlater `if()`-setningen.

B.2.13 `If()`-setning – For å kunne gjøre "veivalg" i programmet

Noen ganger ønsker vi å gjøre forskjellige ting på bakgrunn av ulike betingelser, da kan vi bruke *`if()`-setninger*. Igjen viser vi et konkret eksempel:

```
if (i < 10)
{
   // Gjør dette dersom innholdet i variabelen i < 10
}
```

```
else if (i == 10)
{
  // Gjør dette dersom innholdet i variabelen i = 10
}
else
{
// Gjør dette dersom noe annet er tilfelle i dette tilfellet i > 10
}
```

Avhengig av verdien til variabelen *i* utfører programmet ulike operasjoner. Parentesen etter `if()` skal inneholde en betingelse, denne kan være enkel, som her, eller ganske sammensatt. Man bruker da *aritmetiske operatorer* for å undersøke om noe er større enn (>), mindre enn (<) eller lik (==). Legg merke til det doble likhetstegnet. På tilsvarende måte kan en skrive større eller lik (>=) og mindre eller lik (<=).

Man kan lage flere betingelser med ulik respons ved å bruke en eller flere *`else if()`* med nye betingelser. Men alt som faller utenfor de definerte betingelse kan samles opp i en *`else`*.

Legg merke til at linjen, *`if (i < 10)`* ikke avsluttes med semikolon men etterfølges av { }. Dvs. at *`if()`* er egentlig en funksjon. Heller ikke klammeparentesene etterfølges med ";" (semikolon). Slik er språket definert.

B.2.14 Switch/Case-kommandoen

Switch/Case kommandoen kan minne om `if()`-setningen, i og med at det er en kommando som styrer retningen til programmet på bakgrunn av verdien til en variabel <`var`>.

```
switch (var) {
    case 1:
      //Gjør noe når variabelen var er lik 1
      break;
    case 2:
      //Gjør noe når variabelen var er lik 2
      break;
    default:
      // Om ingenting stemmer med variabelens verdi, gjør default
      // default er valgfritt
    break;
  }
```

Her ser vi at det er verdien til variabelen <var> som bestemmer hvilket av alternativene (case) som velges. Såfremt verdien til <var> eksisterer i lista over alternativer (case), så utføres det som er knyttet til det aktuelle tilfellet. Så snart handlingen er utført sørger kommandoen break; for at programmet forlater lista over alternativer. Dersom ingen av alternativene finner "match" med verdien til variabelen <var>, så utføres default-alternativet.

```
6  void loop() {
7    // read the sensor:
8    int sensorReading = analogRead(A0);
9    // map the sensor range to a range of four options:
10   int range = map(sensorReading, sensorMin, sensorMax, 0, 3);
11
12   // do something different depending on the range value:
13   switch (range) {
14     case 0:    // your hand is on the sensor
15       Serial.println("dark");
16       break;
17     case 1:    // your hand is close to the sensor
18       Serial.println("dim");
19       break;
20     case 2:    // your hand is a few inches from the sensor
21       Serial.println("medium");
22       break;
23     case 3:    // your hand is nowhere near the sensor
24       Serial.println("bright");
25       break;
26   }
27   delay(1);        // delay in between reads for stability
28 }
```

Dersom man sløyfer `break;` under hvert tilfelle, vil programmet hoppe til det aktuelle tilfellet i henhold til variabelen for deretter å gjennomføre alle etterfølgende tilfeller.

Figuren over til høyre viser et eksempel på bruk av switch/case-kommandoen.

Her gjøres det en avlesning av en lyssensor i linje 8, verdien legges inn i variabelen `sensorReading`. En slik avlesning fra en analog inngang vil f.eks. ha verdier i området 0 – 1023. Dersom vi skulle ha en case for hver verdi, ville det bli et særdeles langt program, i stedet bruker vi en kommando som reskalerer området fra 0 – 1023 ned til et område på fra 0 – 4. Til denne reskaleringen bruker vi funksjonen `map()`:

```
range = map(sensorReading, sensorMin, sensorMax, 0, 3);
```

Hvor `sensorMin` og `sensorMax` er henholdsvis minste og største avlesning av lyssensoren. Funksjonen `map()` vil fordele verdiene mellom `sensorMin` og `sensorMax` på heltallsverdier mellom 0 og 3 som tilordnes variabelen `range`. `range` vil dermed få verdier fra 0 til 3 og vil dermed passe godt som variabel i switch/case- funksjonen som vist i figuren over.

B.2.15 Definisjon av egne funksjoner

I figuren under til høyre ser vi `void setup()`-funksjonen og `void loop()`-funksjonen. Disse er ganske tomme i dette eksempelet, men vil normalt være fylt med kode.

Helt nederst har vi laget vår egen funksjon og kalt den `void funksjon1()`. Denne funksjonen er en bit av programmet som gjør en helt spesiell jobb. Det kan f.eks. være å få det grønne lyset i et trafikklys til å blinke et visst antall ganger.

Vi legger også merke til at vi finner funksjonsnavnet igjen under `void loop()`-funksjonen:

```
funksjon1();
```

Når programmet gjennomløper `void loop()`-funksjonen og kommer til `funksjon1()`, så hopper programmet ned til selve funksjonen nederst og utfører kommandoene i funksjonskroppen for så å hoppe tilbake til hovedprogrammet i `void loop()`.

Figuren til høyre viser et eksempel på en egendefinert funksjon. Funksjonen har vi kalt `blinkFemGanger()`. Det er viktig å gi funksjoner meningsbærende navn, som gjør at det er lettere å forstå hva programmet gjør når vi leser koden.

```
void setup()
  {
    // Gjøres en gang ved oppstart
  }

void loop()
  {
    // Gjentas hele tiden
   blinkFemGanger(); // kall funksjonen som lager blink
  }

void blinkFemGanger()
 {
  for (int i = 0; i < 5; i++)
    {
      digitalWrite(LEDpin, HIGH); delay(500);
      digitalWrite(LEDpin, LOW);  delay(500);
    }
 }
```

Definisjonen av funksjonen har vi lagt nederst i programmet. Her er den definert med et navn og et innhold som gjør nettopp det vi forventer – at det grønne lyset blinker fem ganger.

Bruk av argumenter

Hittil har parentesene etter funksjonsnavnet vært tomme. Her kan vi overføre parametere som påvirker hvordan programmet i funksjonen oppfører seg. Dersom vi ønsker at funksjonen skal blinke 6 istedet for 5 ganger, så kan vi lage en ny funksjon som nettopp gjør det, blinker 6 ganger.

En mer elegant måte å gjøre en funksjon mer generell på er å la det være åpent hvor mange blink funksjonen skal utføre. Dette kan vi f.eks. gjøre ved å gi funksjonen et *argument* som overfører det antallet blink vi ønsker at den skal utføre.

I eksempelet til høyre har vi gitt funksjonen argumentet `N`, som er et heltall. Legg merke til at typen til variabelen `N` deklareres i argumentet (`int N`). Variabelen `N` kan så brukes til å utføre funksjonens oppdrag, dvs. å blinke grønt `N` ganger.

I `void loop()`-funksjonen setter vi inn det ønskede antall blink som argument når vi kaller funksjonen. Her bruker vi variabelen `antallBlink` for å overføre antallet. Vi legger merke til at navnet på variabelen i kallet og variabelen i funksjonsdefinisjonen kan være forskjellige, men det er vanlig at variabelens type er den samme, i dette tilfellet et heltall (`int`).

```
void setup()
  {
    // Gjøres en gang ved oppstart
  }

void loop()
  {
    // Det som skal gjentas hele tiden
   antallBlink = 6;                 // Heltallsvariabel
   blinkNGanger(antallBlink); // kall funksjonen som lager blink
  }

void blinkNGanger(int N)
 {
  for (int i = 0; i < N; i++)
    {
      digitalWrite(LEDpin, HIGH); delay(500);
      digitalWrite(LEDpin, LOW);  delay(500);
    }
 }
```

Funksjoner som returnerer en verdi

Noen ganger vil funksjoner utføre beregninger og vi er interessert i resultatet av beregningene. I det neste eksempelet ønsker vi å lage en funksjon som beregner volumet av en kule.

```
...
void loop()
  {
   // Det som skal gjentas hele tiden
   float volum;
   float radiusKule = 5.0;
   volum = volumKule(radiusKule);    // Kall funksjonen
   Serial.println(volum);
  }

float volumKule(float radius)
  {
    float kulevolum;
    kulevolum = (4/3) * 3.14 * pow(radius, 3);
    return kulevolum;
}
```

Vi har kalt funksjonen `volumKule` med argumentet `radius` som er av typen `float` (desimaltall). Når vi kaller funksjonen har vi brukt argumentet `radiusKule`. Det betyr at i kallet så overføres verdien `radiusKule` til funksjonen via variabelen `radius`. Denne brukes så i beregningen. Kulevolumet returneres så med kommandoen `retur kulevolum;` som i sin tur legges i variabelen `volum` som så skrives ut til monitoren med kommandoen `Serial.println(volum);`. Legg merke til at funksjonens type må være det samme som typen til variabelen vi vil returnere, i vårt eksempel `float`.

Når vi tidligere har brukt typen `void` på funksjoner som f.eks. `void setup()` og `void loop()`. så betyr det at funksjonen ikke returnerer noen verdi. Void betyr tom.

B.2.16 Bruk av interrupt

Interrupt brukes når man ønsker å bryte av programmet for å ta seg av en tilfeldig hendelse. Vanligvis er dette hendelser som kommer utenfra og på tilfeldige tidspunkter, som ikke kan forutsies og ikke kan vente, men må behandles umiddelbart. Hendelsene kan inntreffe ved at tilstanden på en port endrer seg, går fra høy til lav eller omvendt.

De ulike mikrokontroller-kortene i Arduino-familien kan håndtere et ulikt antall interrupt og er tilknyttet ulike porter. Tabellen under gir en oversikt over interrupt hos et par Arduino-produkter og ESP8266. Hos både ESP8266 og ESP32 så kan alle portene fungere som en interrupt innganger.

BOARD	INT0	INT1	INT2	INT3	INT4	INT5	...	INT15
UNO	Pin 2	Pin 3						
MEGA	Pin 2	Pin 3	Pin 21	Pin 20	Pin 19	Pin 18		
ESP8266	GPIO 0	GPIO 1	GPIO 2	GPIO 3	GPIO 4	GPIO 5	...	GPIO 15

Som det framgår av tabellen så har Arduino UNO to hårdvare interrupter som er knyttet til port 2 og 3, mens Arduino MEGA har seks interrupts som er knyttet til portene 2, 3, 21, 20, 19 og 18. Vi legger merke til at interruptets nummer (INT0, INT1, INT2, ...) som vi refererer til i programmet, er forskjellig fra portnummeret.

MKR NB 1500 har 10 porter som kan brukes til interrupt: D0, D1, D4 – D9, A1 og A2.

Interrupt service rutinen: Når et interrupt inntreffer vil alle mellomverdier lagres og programmet hopper til interrupt service rutinen (ISR). Denne rutinen er vanligvis svært kort som f.eks. å sette et flagg. Deretter går den tilbake til programmet, henter tilbake alle mellomverdier og fortsetter å kjøre programmet der det slapp. Ved å teste på interrupt-flagget kan man ferdigstille interruptet i ro og mak, med mindre det haster med å fulgt opp.

ISR begrensninger: Siden interrupt blir slått av når programmet hopper til ISR så vil ikke millis() oppdateres og delay()-funksjonen vil ikke fungere mens man er i interrupt service rutinen. Må man bruke delay så kan man bruke delayMicroseconds() som ikke berøres av at interruptet slås av. Det er heller ikke mulig å overføre verdier via argumenter til-, og heller ikke returnere verdier fra ISR.

Prioritet: Når det gjelder hårdvare-interrupt så er det "første mann til mølla" som gjelder, dvs. hardvare interrupt har ikke prioritet. Idet et interrupt inntreffer og programmet går til interrupt service rutinen, så blokkeres for andre interrupt, disse blir lagt på vent til det første interruptet er behandlet[90]. Når *det* er gjort håndteres ventende henvendelser.

Programvaremessig består interruptet av følgende kommandoer:

Deklarer porten som skal brukes som interrupt-inngang:

```
const byte pin = 3;
```

Dernest tilordnes interruptet en navngitt service rutine (ISR) og en modus (mode):

```
attachInterrupt(digitalPinToInterrupt(pin), ISR, mode)
```

Denne legges gjerne i void setup()-funksjonen. `digitalPinToInterrupt(pin)` vil automatisk finne interrupt nummeret (INT0, INT1 osv) dersom man vet hvilken port som er benyttet. ISR er navnet på servise rutinen og mode angir hvordan hendingen som trigger interruptet skal være. Følgende muligheter finnes: På fallende flanke (FALLING), på stigende flanke (RISE), når et skifte inntreffer (CHANGE), når den er lav (LOW) eller når den er høy (HIGH).

Interrupt kan også kobles fra med følgende kommando:

```
detachInterrupt(digitalPinToInterrupt(pin))
```

Dernest lages en Interrupt Service Rutine (ISR). Denne må ha samme navn som angitt i tilordningen (ISR), men navnet bestemmer man selv. Rutinen eller funksjonen legges hvor som helst i programmet bare utenfor void setup() eller void loop(). Følgende er et enkelt eksempel:

```
void wspeedIRQ()
{
  windClicks++; // Øk antall pulser med 1
}
```

I dette eksempelet ser vi at det eneste som skjer i ISR er å øke et antall, praksis telle antall interrupt.

90.http://www.gammon.com.au/interrupts

Vi kan også slå av og på interrupt med følgende kommandoer i programmet. Når vi gjør dette så utfører vi handlingen for samtlige interrupt.

```
noInterrupts();
```

Bruker vi noInterrupts(); så vil også dette påvirke enkelte funksjoner i programmet, som f.eks. at millis() vil ikke oppdateres og en del kommunikasjon vil heller ikke bli registrert.

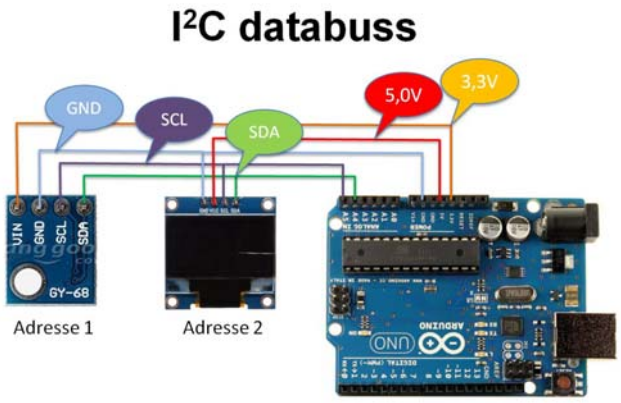
```
interrupts();
```

Denne kommandoen gjenoppretter alle interrupt.

Uansett ulemper så er bruken av interrupt et særdeles nyttig hjelpemiddel for å kunne håndtere tilfeldige eksterne kortvarige hendelser.

B.3 Bruk av I²C buss og biblioteker

I dette avsnittet skal se på hvordan vi kan bruke I²C-[91] buss til å overføre data fra sensorer utstyrt med denne typen kommunikasjonsgrensesnitt, som f.eks. trykkmåleren BMP180/BMP280 eller BME280 og displayet SSD1306. Til dette bruker vi som oftest spesiallagede biblioteker som må installeres dersom de ikke alt finnes blant bibliotekene som følger med Arduino editoren (IDE).



Den såkalte I²C-bussen består at to kommunikasjonslinjer, SDA og SCL, som gjør det mulig å sende data etter hverandre (på serieform) mellom ulike kretser. Siden hver krets som er koblet til bussen har sin unike adresse, kan man adressere meldingene til den komponenten man er interessert i å kommunisere med.

Standardbiblioteket for I²C følger med Arduino programvaren, men krever at man inkluderer "header"-filen: "wire.h" øverst i koden ved å skrive:

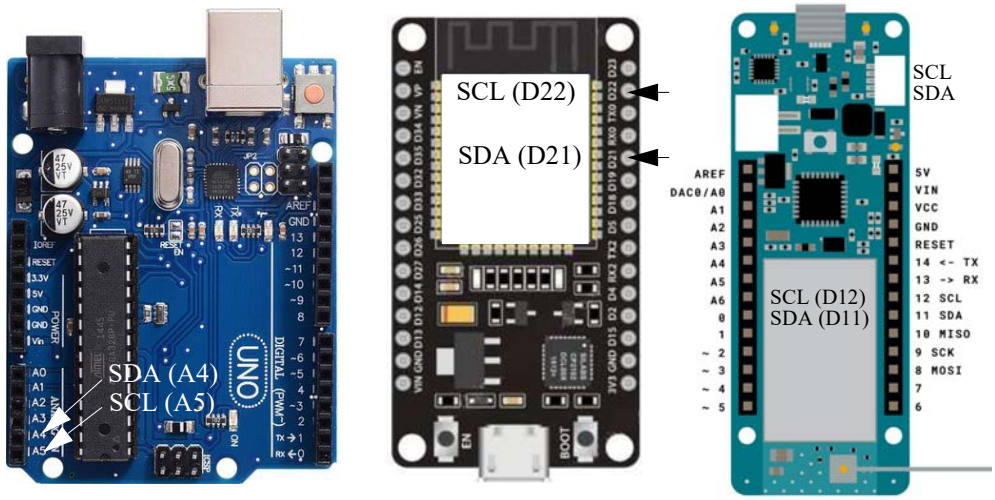
```
#include <wire.h>
```

Dette medfører at programmet får tilgang til en del funksjoner knyttet til kommunikasjon via I²C-bussen.

For mange spesiallagde kretser finnes det spesielle biblioteker med funksjoner som gjør dem lettere å bruke. Slike biblioteker må gjerne hentes ned fra leverandøren eller andre og installeres i programeditoren.

91. I²C står for Inter Integrated Circuit, også forkortelsen I2C og IIC brukes.

For Arduino UNO brukes A4 (SDA) og A5 (SCL) for å kommunisere med I^{2}C-bussen. For ESP32 brukes gjerne D21 (SDA) og D22 (SCL). For MKR NB 1500 kan man enten bruke D11 (SDA) og D12 (SCL) eller en egen kontakt øverst på kortet.



Arduino UNO **ESP32**

I neste avsnitt skal vi se hvordan vi kan laste ned og installere biblioteker generelt og deretter biblioteker for I^{2}C spesielt.

B.3.1 Installasjon av pakkede biblioteker generelt

Som eksempel bruker vi biblioteket som trengs for å lese data fra trykksensoren BMP180 som også inneholder en temperatursensor.

1. Last ned biblioteket i pakket form (*.zip). Denne pakken inneholder header-filer (*.h), biblioteksfunksjoner (*.cpp) og gjerne eksempler på bruk av sensoren. Er man heldig inneholder den også en beskrivelse av biblioteksfunksjonene. Man finner gjerne biblioteket på nett ved søke etter den aktuelle kretsen etterfulgt av ordet *library* om adressen ikke er kjent.

   Fra og med Arduino IDE versjon 1.5 kan man fra menylinjen velge:
   Sketch/Include library/Manage Libraries
   og skrive inn den aktuelle sensoren i søkefeltet. Man får da beskjed om biblioteket er installert eller forslag til hvor man kan hente det fra.

2. **Last ned biblioteket fra ekstern kilde:**
   Dersom biblioteksfilen for komponenten ikke finnes, kan man hente biblioteksfilene til BMP180 fra ulike steder, f.eks. fra SparkFun's hjemmeside:

   https://learn.sparkfun.com/tutorials/bmp180-barometric-pressure-sensor-hookup-/installing-the-arduino-library

   For BMP280 gjelder følgende adresse:

   https://github.com/adafruit/Adafruit_BMP280_Library

og for BME280 gjelder følgende adresse:

https://github.com/adafruit/Adafruit_BME280_Library

Vi har også lagt dem ut under den blå hefteserien:

https://www.ntnu.no/skolelab/bla-hefteserie

3. **Installer biblioteket:**
   Dernest installeres biblioteket i Arduino-programvaren ved å velge:
   *Sketch/Include Library/Add *.ZIP Library* for så å velge den nedlastede zip-pakkede filen.
   Biblioteksfilene blir dermed lagt under *Sketch/include library* og eksempler under: *File/ Eksempler*
4. **Inkluder headerfilen (*.h) i programmet**
   Det aktuelle biblioteket inkluderes i programmet ved å velge:
   Sketch/Include Library/<aktuelle biblioteket>
   Dermed er header-filen på plass i programmet.
   Det vanligste er at man skriver den inn manuelt hvilket forutsetter at man kjenner navnet på h-filen.

B.3.2 Bibliotek for bruk av I²C

Også I²C-bussens bibliotek må lenkes inn i koden når den skal brukes. Bussen fungerer ved at en "master" styrer kommunikasjonen. Vanligvis er dette Arduino mikrokontrollerkortet som automatisk blir definert som master ved kommandoen `Wire.begin();` kommandoen i setup-funksjonen.

Biblioteket – Wire.h:

Biblioteket inneholder noen funksjoner som vi skal oppsummere her:

"Header file":

```
#include <Wire.h>
```

"Header" kommandoen knytter I²C-biblioteket til programmet slik at kompilatoren vet hvilket bibliotek funksjonene skal hentes fra.

```
Wire.begin();
```

Denne funksjonen definerer Arduino mikrokontrolleren som master for kommunikasjonen, dvs. at all kommunikasjon med sensorer og andre periferikretser som er koblet til I²C-bussen styres av Arduino'en. Siden Arduino'en er masteren, er det ikke nødvendig å definere en adresse (`Wire.begin(<adresse>);`) for denne. Ev. kan adressen være et tall mellom 0 – 127 som identifiserer masteren på bussen.

```
Wire.beginTransmission(<adresse>);
```

Med funksjonen ***`Wire.beginTransmission(<adresse>);`*** kan man starte en dataoverføring til en enhet på bussen med den spesifiserte adressen.

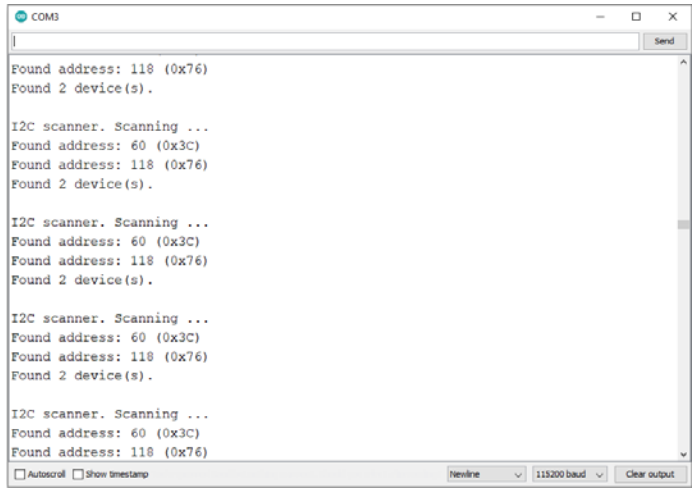
Vi skal senere se hvordan vi anvender dette i praksis.

For tilkobling til I^{2}C hos ESP32 ta gjerne en titt på følgende nettside: https://randomnerdtutorials.com/esp32-i2c-communication-arduino-ide/#1

Program for scanning av adresser på I^{2}C-buss

Dersom man er usikker på hva enhetens adresse er kan man koble enheten til bussen og benytte et programhjelpemiddel for scanning av adressene til de tilkoblete enhetene. Man kobler opp enheten og installerer scanningprogrammet: *i2C_scanning.ino* i Arduino'en. Programmet kan kopieres fra denne siden: https://playground.arduino.cc/Main/I2cScanner/ Programmet vil lete etter adressene til de enhetene som er koblet på bussen og vise resultatet i monitoren som vist på figuren under.

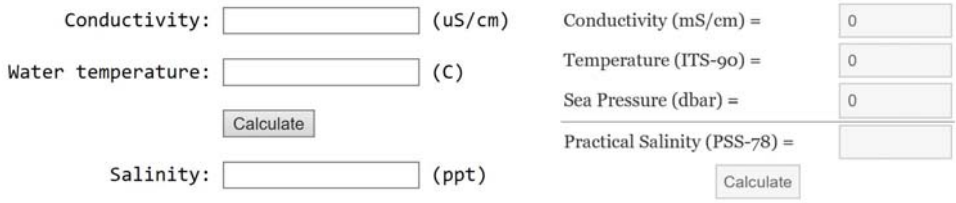
Som vi ser av figuren kan flere enheter med ulike adresser være koblet til bussen, og programmet viser alle som det finner.



Vedlegg C Omregning fra ledningsevne til saltholdighet

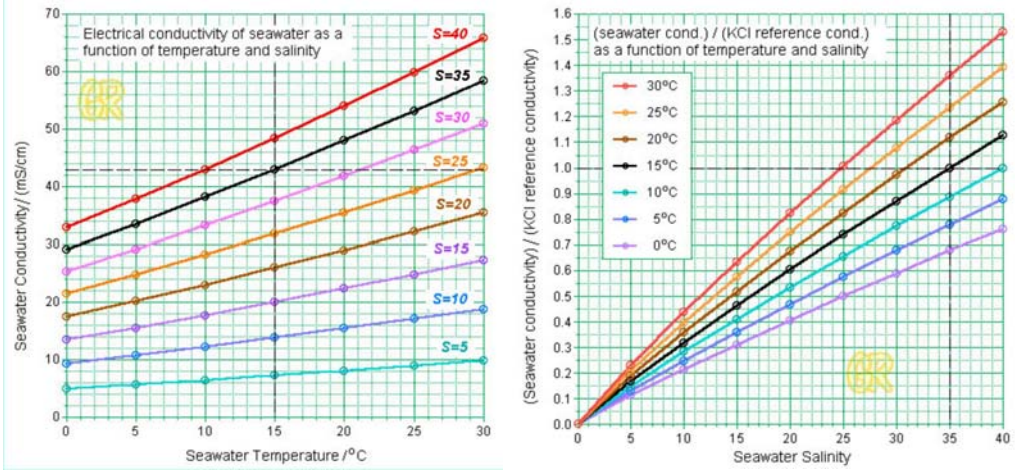
C.1 Ledningsevne som funksjon av saltholdighet, temperatur og trykk

For ev.- å lette utregningen av saltholdighet (salinitet) på bakgrunn av ledningsevnen er det utviklet flere kalkulatorer.



Figuren over viser to eksempler på slike kalkulatorer. Mens den til venstre har ledningsevne og temperatur som parametere[92], så har den til høyre også med trykk (dBar)[93].

Ved bruk av kalkulatorer som disse kan man lage kurveskarer som viser sammenhengen mellom de ulike parametrene som vist på figuren under[94]:



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[95].

For ytterligere diskusjon av salinitet, se N.K. Rossing, "MauSea – trykk- og temperaturmåling under vann" rev. 4.6

92. http://salinometry.com/ctd-salinity-calculator/
93. http://www.fivecreeks.org/monitor/sal.shtml
94. http://www.grabovrat.com/handbook/handbookH08g1.html
95. Standard vannprøven er laget ved å å blande ut 32.4356 gram KCl i a1 liter destillert vann som gir akkurat 35‰ ved 15°C og 1 atm.

Vedlegg D Feilfinning og problemer

I dette vedlegget skal vi samle lure triks for å komme rundt feil som kan oppstå.

D.1 Problemer med opplasting av programmet[96]

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 skjermet 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.

D.2 Problemer med opplasting av programmet pga dvale[97]

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 D.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.

96. En takk til *Pert* https://forum.arduino.cc/t/unbricking-an-audrino-mkr1500/878247
97. En takk til *Pert* https://forum.arduino.cc/t/mrk-1500-stuck-in-deep-sleep/690513

D.3 Problemer med å lese verdier fra MKR GPS sammen med MKR ENV[98]

Vår 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-eksempelet:

```
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);
```

98.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 kopiert ut. Så snart denne ble lagt inn i programmet så sluttet det å skrive ut GPS-måleverdiene. 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 returen fra skal være `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 de som om de kommer umiddelbart og programmet kan gå videre.

Problemet løst.

D.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);
}
```

D.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()) {}
```

D.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 av 926mBar uten GPS-kortet tilkoblet og 2591mBar GPS-kortet er tilkoblet. Dette er ikke helt konsekvent. Vi har likevel observert at dette ikke er helt konsekvent. Det er registrert riktige verdier. Vi har også observert det samme problemet på ulike GPS-kretser. Det er ikke sjekket om problemet

Problemet er ikke løst.

D.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 debuging.

D.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 "vakthund" 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 "vakthunden" får spenning også under dvale.

D.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 "vakthunden".

I et slikt tilfelle må vakthunden 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 vakthunden 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.

D.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 E Ressursprogrammer

Dette vedlegget inneholder noen nyttige programmer som vi kan ha bruk for når vi setter opp prosjektet vårt.

E.1 Hjelpeprogrammer

E.1.1 Scan I²C-bussen etter adresser til påkoblede elementer

Programmet scanner I²C-bussen og lister opp adressene til alle som er koblet på bussen. Programmet er nyttig for å finne I²C-adressen til en sensor eller aktuator.

```
// ESP32 I2C Scanner
// Based on code of Nick Gammon  http://www.gammon.com.au/forum/?id=10896
// ESP32 DevKit - Arduino IDE 1.8.5
// Device tested PCF8574 - Use pullup resistors 3K3 ohms !
// PCF8574 Default Freq 100 KHz

#include <Wire.h>

void setup()
{
  Serial.begin (115200);
  Wire.begin (21, 22);   // sda= GPIO_21 /scl= GPIO_22
}

void Scanner ()
{
  Serial.println ();
  Serial.println ("I2C scanner. Scanning ...");
  byte count = 0;

  Wire.begin();
  for (byte i = 8; i < 120; i++)
  {
   Wire.beginTransmission (i);         // Begin I2C transmission Address
(i)
    if (Wire.endTransmission () == 0)  // Receive 0 = success (ACK
response)
```

```
    {
      Serial.print ("Found address: ");
      Serial.print (i, DEC);
      Serial.print (" (0x");
      Serial.print (i, HEX);     // PCF8574 7 bit address
      Serial.println (")");
      count++;
    }
  }
  Serial.print ("Found ");
  Serial.print (count, DEC);        // numbers of devices
  Serial.println (" device(s).");
}

void loop()
{
  Scanner ();
  delay (100);
}
```

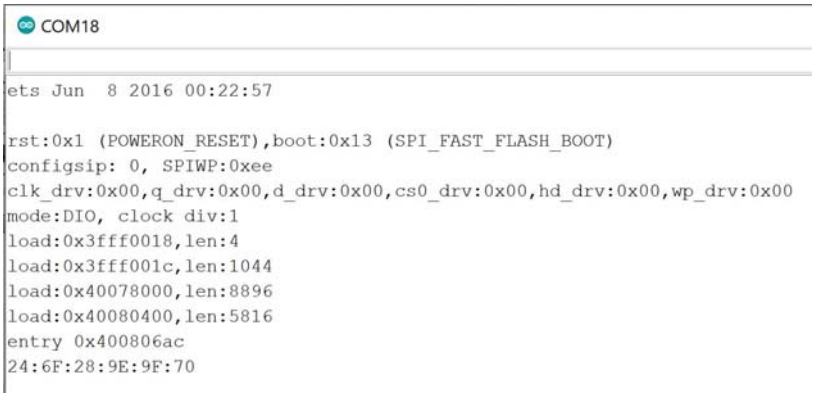
E.1.2 Finn kontrollerens MAC-adresse

Programmet lastes opp på en ESP og vil returnere komponentens MAC-adresse.

```
// Complete Instructions to Get and Change ESP MAC Address: https://
RandomNerdTutorials.com/get-change-esp32-esp8266-mac-address-arduino/

#include "WiFi.h"
#include <esp_now.h>

void setup(){
  Serial.begin(115200);
  WiFi.mode(WIFI_MODE_STA);
  Serial.println(WiFi.macAddress());
}

void loop(){
```

```
}
```

Figuren under viser resultatet av kjøring av programmet på ESP32.



MAC-adressen til denne komponenten er: 24:6F:28:9E:9F:70 Adressen er gitt i heksadesimale siffer.

E.1.3 Program for henting av IMSI og IMEI[99]

```
// Baud rate used for both Serial ports
unsigned long baud = 115200;

void setup() {
  // enable the POW_ON pin
  pinMode(SARA_PWR_ON, OUTPUT);
  digitalWrite(SARA_PWR_ON, HIGH);

  Serial.begin(baud);
  SerialSARA.begin(baud);
}

void loop() {
  if (Serial.available()) {
    SerialSARA.write(Serial.read());
  }

  if (SerialSARA.available()) {
    Serial.write(SerialSARA.read());
  }
}
```

99.https://startiot.telenor.com/tutorials/arduino-dev-kit-coap/#user-content-34-get-imsi-and-imei

E.2 Testprogrammer for shield-kort

Her har vi samlet testprogrammet for uttesting av ulike shield-kort.

E.2.1 Testprogram for MKR ENV[100]

```
#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);
  Serial.println(" kPa");
```

100.https://docs.arduino.cc/tutorials/mkr-env-shield/mkr-env-shield-basic

```
  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);
}
```

E.2.2 Testprogram for MKR ENV skriving til SD-kort[101]

```
/*
  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
const int SD_CS_PIN = 4;
```

101.https://docs.arduino.cc/tutorials/mkr-env-shield/mkr-env-shield-basic

```
// 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);
```

```
}
```

E.2.3 Testprogram for MKR GPS[102]

```
/*
 * 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();

    // Skriv ut GPS verdier
```

102.https://docs.arduino.cc/tutorials/mkr-gps-shield/mkr-gps-basic

```
    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();
}
```

E.2.4 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);
}
```

E.2.5 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 begeregnes 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, registert lav temperatur)
int D_T_Hoy = 710;  // Digital verdi for spenningen på A0, registert 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

// -------------------- 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 tempe-
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);
}
```

E.2.6 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);
```

```
}
```

E.2.7 Testprogram for avlesning av spenning hos sensor for måling av oppløst oksygen

Programmet leser av spenningen fra sensoren som måler oppløst oksygen og brukes under kalibreringen av sensoren.

```
#include <Arduino.h>

#define VREF    3300//VREF(mv) Sjekk gjerne referansespenningen
#define ADC_RES 1024//ADC Resolution Default for MKR er 10 bit

uint32_t raw;

void setup()
{
    Serial.begin(115200);
}

void loop()
{
    raw=analogRead(A1);
    Serial.println("raw:\t"+String(raw)+"\tVoltage(mv)"+
String(raw*VREF/ADC_RES));
    delay(1000);
}
```

E.2.8 Testprogram for måling av oppløst oksygen

Testprogram for måling av oppløst oksygen. Programmet skal inneholde kalibrerte verdier, se programlinjer merket rødt:

```
#include <Arduino.h>

#define DO_PIN A1

#define VREF 3300 //VREF (mv)
#define ADC_RES 1024 //ADC Resolution

//Single-point calibration Mode=0
```

```
//Two-point calibration Mode=1
#define TWO_POINT_CALIBRATION 0

#define READ_TEMP (25) //Current water temperature ?, Or temperature
sensor function

//Single point calibration needs to be filled CAL1_V and CAL1_T
#define CAL1_V (1600) //mv Erstatt 1600 mV med målte verdier
#define CAL1_T (25)   //? Erstatt 25 med den aktuelle temperaturen
//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)   //?

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);
}

void loop()
{
  Temperaturet = (uint8_t)READ_TEMP;
  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);
}
```

E.2.9 Testprogram for kalibrering av sensor for måling av ledningsevne

Programmet følger med som et eksempelprogram når vi installerer biblioteket til ledningsevnesensoren fra DFRobot[103]. NB – Legg merke til at avlesning av temperatursensoren må legges inn under funksjonen `readTemperature()` slik at funksjonen returnerer den avleste verdien. Dersom vi bruker DS18B20-sensoren så bruk deler av programmet i vedlegg E.2.6 side 213.

```
/*
 * file DFRobot_EC10.ino
 * @ https://github.com/DFRobot/DFRobot_EC10
 *
 * This is the sample code for Gravity: Analog Electrical Conductivity
Sensor / Meter Kit(K=10), SKU: DFR0300_H.
 * In order to guarantee precision, a temperature sensor such as DS18B20
is needed, to execute automatic temperature compensation.
 * You can send commands in the serial monitor to execute the calibration.
```

103.https://wiki.dfrobot.com/Gravity_Analog_Electrical_Conductivity_Sensor_Meter_K=10_SKU_DFR0300-H#target_5

```
 * Serial Commands:
 *   enterec -> enter the calibration mode
 *   calec -> calibrate with the standard buffer solution, one buffer
solutions(12.88ms/cm) will be automaticlly recognized
 *   exitec -> save the calibrated parameters and exit from calibration
mode
 *
 * Copyright   [DFRobot](https://www.dfrobot.com), 2018
 * Copyright   GNU Lesser General Public License
 *
 * version  V1.0
 * date  2018-11
 */

#include "DFRobot_EC10.h"
#include <EEPROM.h>

#define EC_PIN A1
float voltage,ecValue,temperature = 25;
DFRobot_EC10 ec;

void setup()
{
  Serial.begin(115200);
  ec.begin();
}

void loop()
{
    static unsigned long timepoint = millis();
    if(millis()-timepoint>1000U)  //time interval: 1s
    {
      timepoint = millis();
      voltage = analogRead(EC_PIN)/1024.0*5000;  // read the voltage
      Serial.print("voltage:");
      Serial.print(voltage);
```

```
     //temperature = readTemperature();  // read your temperature sensor
to execute temperature compensation
      ecValue =  ec.readEC(voltage,temperature);  // convert voltage to
EC with temperature compensation
       Serial.print("  temperature:");
       Serial.print(temperature,1);
       Serial.print("^C  EC:");
       Serial.print(ecValue,1);
       Serial.println("ms/cm");
    }
    ec.calibration(voltage,temperature);  // calibration process by
Serail CMD
}

float readTemperature()
{
  //add your code here to get the temperature from your temperature sensor
}
```

E.2.10 Testprogram for å legge i dyp søvn og vekkes fra en port[104]

```
/*
  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 advan-
tage 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
```

104.https://github.com/arduino-libraries/ArduinoLowPower/blob/master/examples/ExternalWakeup/ExternalWakeup.ino

```
// 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);
    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 ++;
}
```

E.2.11 Testprogram for å legge i dyp søvn og vekkes av intern "time out"[105]

```
/*
```

105.https://github.com/arduino-libraries/ArduinoLowPower/blob/master/examples/TimedWakeup/TimedWakeup.ino

```
  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);
}


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
}
```

E.3 Programmer for testing av overføring av data til server

E.3.1 Testprogram (enkelt) for testing av overføring av data til server

Programmet sender over et minimum av data til serveren. All koden av betydning er lagt i setup()-funksjonen slik at *den kjøres kun en gang* før den stopper. I lopp()-funksjonen ligger kun en funksjon som sjekker om noe tekst er mottatt fra serveren før programmet stopper:

```
/*

  Test of Web client (Eksempel-kode-1.ino)

  This sketch connects to a website through a MKR NB 1500 board. Specifically,
  this example downloads the URL "http://example.org/" and
  prints it to the Serial monitor.

  Circuit:
   - MKR NB 1500 board
   - Antenna
   - SIM card with a data plan

  created 8 Mar 2012
  by Tom Igo
  Modified 30. aug. 2022
  by Håvard Holm/Nils Kr. Rossing
*/

// libraries
#include <MKRNB.h>

// #include "arduino_secrets.h"
// Please enter your sensitive data in the Secret tab or arduino_secrets.h
// PIN Number
const char PINNUMBER[]     = "";

// initialize the library instance
NBClient client;
GPRS gprs;
NB nbAccess;

// URL, path and port (for example: example.org)
char server[] = "sensor.marin.ntnu.no";
char path[]   = "/cgi-bin/tof.cgi?";
```

```
int port = 80; // port 80 is the default for HTTP

void setup() {
  char buffer[256];
  // 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
  }

  Serial.println("Starting Arduino web client.");
  // connection state
  boolean connected = false;

  // After starting the modem with NB.begin()
  // attach to the GPRS network with the APN, login and password
  while (!connected) {
    //if ((nbAccess.begin(PINNUMBER) == NB_READY) &&
    if ((nbAccess.begin("", true, true) == NB_READY) &&
        (gprs.attachGPRS() == GPRS_READY)) {
      connected = true;
    } else {
      Serial.println("Not connected");
      delay(1000);
    }
  }

  Serial.println("connecting...");

// if you get a connection, report back via serial:
  if (client.connect(server, port)) {
    Serial.println("connected");
    // Make a HTTP request:
    client.print("GET ");
    //client.print("/cgi-bin/test.cgi?Gr1,lat=67.8,lon=7.8");
    //sprintf(buffer, "/cgi-bin/tof.cgi?haavard,lat=%lf,lon=%lf", 67.5, 5.6);
    sprintf(buffer, "/cgi-bin/tof.cgi?Gr-no,lat=%lf,lon=%lf", 67.512345,
5.654321);
    client.print(buffer);
    client.println(" HTTP/1.1");
    client.print("Host: ");
    client.println(server);
```

```
    client.println("Connection: close");
    client.println();
  } else {
    // if you didn't get a connection to the server:
    Serial.println("connection failed");
  }
}

void loop() {
  // if there are incoming bytes available
  // from the server, read them and print them:
  if (client.available()) {
    Serial.print((char)client.read());
  }

  // if the server's disconnected, stop the client:
  if (!client.available() && !client.connected()) {
    Serial.println();
    Serial.println("disconnecting.");
    client.stop();

  // do nothing forevermore:
    for (;;)
      ;
  }
}
```

E.3.2 Testprogram for testing av overføring av dummy-data til server

Programmet sender over dummy-data til serveren. Noen av dataene er tilført random-verdier for å simulere variasjon. Programmet sender 5 påfølgende måleserier før det stopper. Programmet er ment å være grunnlaget for å legge inn virkelige data fra MKR ENV, MKR GPS i tillegg til ev. andre sensorer.

```
/*
  Eksempel-kode-3

 Dette kodeeksempelet kobler opp mot en webserver ved hjelp av et MKR NB 1500 kort.
  Koden lager dummy-data med noe random tillegg.
  Koden legger dataene på URL-adressen "http://sensor.marin.ntnu.no"
  samtidig som den skriver måledatene til Seriemonitoren. Koden kjøres 5 ganger
før den
```

```
  avsluttes.

  Circuit:
   - MKR NB 1500 board
   - Antenna
   - SIM-kort

  Utviklet av Tom Igor 08.03.12
  Modifisert av Håvard Holm 2021
  Modifisert av Nils Kr. Rossing 30.08.22

*/

// libraries
#include <MKRNB.h>

// PIN Number
const char PINNUMBER[]     = "";

// initialize the library instance
NBClient client;
GPRS gprs;
NB nbAccess;

// URL, path og port
char server[] = "sensor.marin.ntnu.no";
char path[] = "/cgi-bin/tof.cgi?";
int port = 80; // port 80 er default for HTTP

// Deklarasjon av sensorvariabler fra ENV-kort
float temperature = 0;  // Målt lufttemperatur i C
float humidity    = 0;  // Målt luftfuktighet i %
float pressure    = 0;  // Målt lufttrykk i mBar
float illuminance = 0;  // Målt lysintensitet i Lux

// Deklarer sensorvariable fra vannsensorer
float w_temperature = 0;// Målt vanntemperatur

// Deklarasjon av sensorvariabler fra GPS-kort
float latitude;         // Breddegrad
float longitude;        // Lengdegrad
```

```
float altitude;          // Høyde over havet
float speed;             // Beregnet hastighet
float timeSec;           // Antall sekunder fra 1.1.1980 - Epoketid
int   satellites;        // Antall satellitter
long  no = 0;            // Måling nummer

// Deklarasjon av buffer for dataoverføring
char buffer[256];
boolean connected = false; // Status oppkobling

void setup() {

  Serial.begin(9600);   // Initialiser monitoren
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }
}

void loop()
{
  for(int i=1; i<6; i++)
  {
  no=i;
  readGPSdata();           // Les GPS-posisjon mm
  readENVdata();           // Les ENV-målinger
  readWaterData();         // Les av sensorer i vann
  printDataToMonitor();    // Skriv måleverdier til monitor i samme format som
datafila
  connectToGPRS();       // Koble opp til GPRS nettverket med APN, login og passord

  Serial.print("Kobler til og overfører data: "); Serial.println(no);
  connectToServer();       // Overfør data til server
  //checkServerMessage();   // Sjekk om det er noen melding fra serveren til
sensornoden

  delay(5000);
  }
  for(;;)                  // Stop programmet
  ;
}

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);
    }
  }
}

void connectToServer()
{
// Koble opp til server og send over data, varsler om oppkobling mislykkes

  if (client.connect(server, port))
  {
    Serial.println("Tilkoblet server");

    client.print("GET "); // Gjør et HTTP request:
    sprintf(buffer, "/cgi-bin/tof.cgi?Gr-
no,no=%d,time=%.2f,lat=%.6f,lon=%.6f,C_luft=%.1f,C_vann=%.1f,hum=%.1f,pres=%.1f,
light=%.1f", no, timeSec, longitude, latitude, temperature, w_temperature, humi-
dity, pressure, illuminance); // Legg måleverdier inn i bufferet
    client.print(buffer);
    client.println(" HTTP/1.1");
    client.print("Host: ");
    client.println(server);

    client.println("Connection: close");
    client.println();
  }
  else
```

```
  {
    Serial.println("Mislykket tilkobling til server"); // Dersom tilkobling
mislyktes
  }
}

void checkServerMessage()
{
   // Dersom det er innkommende data fra serveren, les dem og skriv dem ut
  if (client.available())
  {
    Serial.print((char)client.read());
  }

  // Dersom serveren er frakoblet stopp klienten:
  if (!client.available() && !client.connected())
  {
    Serial.println();
    Serial.println("Kobler fra klienten");
    client.stop();

  // do nothing forevermore:
    for (;;)
      ;
  }
}

void readGPSdata()
{
  Serial.println("Leser GPS-data");
  latitude   = 63.426269 + random(0, 9)/100.0;
  longitude  = 10.454045 + random(0, 9)/10.0;
  altitude   = 65        + random(0, 4);
  speed      = 0         + random(0, 9);
  satellites = 6;
  timeSec    = millis()/1000.0;
}

void readENVdata()
{
  Serial.println("Leser miljo-data");
  temperature = 21.0;
```

```
  humidity    = 45.5;
  pressure    = 1089.3;
  illuminance = 148.5;
}

void readWaterData()
{
  Serial.println("Leser vanntemperatur");
  w_temperature = 12.0;
}

void printDataToMonitor()
{
  sprintf(buffer, "Gr-1,
no=%d,time=%.2f,lat=%.6f,lon=%.6f,C_luft=%.1f,C_vann=%.1f,hum=%.1f,pres=%.1f,lig
ht=%.1f", no, timeSec, longitude, latitude, temperature, w_temperature, humidity,
pressure, illuminance); // Legg måleverdier inn i bufferet
  Serial.println(buffer);
}
```

E.4 Programmer for uttesting av RTC-klokken DS3231

Denne kretsen kan brukes dersom man ønsker å spare maksimalt med strøm ved at denne klokka slår på strømmen når det skal foretas en måling for deretter å sende over data.

E.4.1 Program for å stille klokka hos DS3231 (setTimeDS3231Man-1)

I dette programmet setter man tidspunktet når klokka starter for deretter å skrive ut tidspunkt og dato for hvert sekund.

```
// Sett data og tid manuelt ved bruk av DS3231 RTC tilkoblet I2C og Wire
biblioteket


#include <Wire.h>
#include "Sodaq_DS3231.h"

char weekDay[][5] = {"Søn", "Man", "Tirs", "Ons", "Tors", "Fre", "Lør" };

//År, Måned, Dato, Time, Minutt, Sekund og ukedag (går fra 0 til 6)
DateTime dt(2022, 07, 31, 20, 33, 45, 0);



void setup ()
```

```
{
    Serial.begin(57600);
    Wire.begin();
    rtc.begin();
    rtc.setDateTime(dt); //Registrer tiden i DS3132
}

void loop ()
{
    // Skriv ut tiden i monitoren
    DateTime now = rtc.now(); //Henten tiden
    Serial.print(now.year(), DEC);
    Serial.print('-');
    Serial.print(weekDay[now.dayOfWeek()]);
    Serial.print("dag ");
    Serial.print(now.date(), DEC);
    Serial.print('/');
    Serial.print(now.month(), DEC);
    Serial.print(' ');
    Serial.print(now.hour(), DEC);
    Serial.print(':');
    Serial.print(now.minute(), DEC);
    Serial.print(':');
    Serial.print(now.second(), DEC);
    Serial.println();
    delay(1000);
}
```

E.4.2 Sett opp en enkel alarm (*setIntAlarm.ino*)

Programmet viser et eksempel på hvordan man kan sette opp en alarm.

```
// Programmet illustrerer hvordan man kan sette opp et interrupt fra
DS3231
// Når det oppnåes match mellom et på forhånd definert tidspunkt og
nåtiden,
// legges interuptutgangen lav (SQW). En kan også bestemme om match
gjelder
// bare timer, minutter eller sekunder, ev. en kombinasjon av disse.
```

```
#include <Wire.h>
#include "ds3231.h"

#define BUFF_MAX 256

// Tid for oppvekking
uint8_t wake_HOUR = 15;
uint8_t wake_MINUTE = 49;
uint8_t wake_SECOND = 59;

// Hyppigheten for oppdatering
int pinRTCINT = 3;

void setup()
{
    Serial.begin(9600);
    pinMode(pinRTCINT,INPUT);
    Wire.begin();
    DS3231_init(DS3231_INTCN);
    DS3231_clear_a1f();
    set_alarm();
}

void loop()
{
     printTime();
     delay(1000);
     if (digitalRead(pinRTCINT) == 0)
     {
        Serial.println("Alarmen er utløst");
        // Resett interrupt-linjen (high-z)
        delay(5000);
        DS3231_clear_a1f();
      }
}
```

```
void printTime()
{
  struct ts t;
  DS3231_get(&t); // Hent tiden fra DS3231
  Serial.print(t.mday); Serial.print("/");
  Serial.print(t.mon); Serial.print(" ");
  Serial.print(t.year+100); Serial.print(" ");
  Serial.print(t.hour); Serial.print(":");
  Serial.print(t.min); Serial.print(":");
  Serial.print(t.sec); Serial.println("");
}

void set_alarm(void)
{

    // flags[] angir hvilke komponenter i dato og tid som skal sjekkes
for å gi alarm
    // A1M1 (seconds) (0 to enable, 1 to disable)
    // A1M2 (minutes) (0 to enable, 1 to disable)
    // A1M3 (hour)    (0 to enable, 1 to disable)
    // A1M4 (day)     (0 to enable, 1 to disable)
    // DY/DT          (dayofweek == 1/dayofmonth == 0)
    //Sjekker {sek, min, timer, dag, dag i uke/måned}
    uint8_t flags[5] = { 0, 1, 1, 1, 1 };

    // Sett Alarm1
    DS3231_set_a1(wake_SECOND, wake_MINUTE, wake_HOUR, 0, flags);

    // Aktiver Alarm1
    DS3231_set_creg(DS3231_INTCN | DS3231_A1IE);
}
```

E.4.3 Program for å sette opp en gjentatt alarm (*setIntAlarm-1*)

Programmet gir mulighet for å sette opp en alarm med jevne mellomrom, enten med intervaller med et visst antall sekunder (10 – 60), eller med intervaller med et visst antall minutter (1 – 60), eller med intervaller med et visst antall timer (1 – 24).

```
// setIntAlarm-1


// Programmet illustrerer hvordan man kan sette opp et interrupt fra
DS3231
// med jevne intervaller i sekunder (10 - 6=), minutter (1 - 60 eller
timer 1 - 24.
// Når match oppnås legges interuptutgangen lav (SQW).


#include <Wire.h>
#include "ds3231.h"

#define BUFF_MAX 256

// Tid for oppvekking
uint8_t wake_HOUR = 0;
uint8_t wake_MINUTE = 0;
uint8_t wake_SECOND = 0;
uint8_t wake_interval_HOUR = 0;
uint8_t wake_interval_MINUTE = 0;
uint8_t wake_interval_SECOND = 10;

// Hyppigheten for oppdatering
int pinRTCINT = 3;

void setup()
{
    Serial.begin(9600);
    pinMode(pinRTCINT,INPUT);
    Wire.begin();
    DS3231_init(DS3231_INTCN);
    DS3231_clear_a1f();
    set_alarm();
```

```
    Serial.print("setIntAlarm-1");
}

void loop()
{
     printTime();
     delay(1000);
     if (digitalRead(pinRTCINT) == 0)
     {
        Serial.println("Alarmen er utløst");
        // Resett interrupt-linjen (high-z)
        delay(2000);
        set_alarm();
        DS3231_clear_a1f();
      }
}

void printTime()
{
  struct ts t;
  DS3231_get(&t); // Hent tiden fra DS3231
  Serial.print(t.mday); Serial.print("/");
  Serial.print(t.mon); Serial.print(" ");
  Serial.print(t.year+100); Serial.print(" ");
  Serial.print(t.hour); Serial.print(":");
  Serial.print(t.min); Serial.print(":");
  Serial.print(t.sec); Serial.println("");
}

void set_alarm(void)
{

    // Beregne neste tidspunkt
    wake_SECOND = wake_SECOND + wake_interval_SECOND; if(wake_SECOND >=
60) {wake_SECOND = wake_SECOND - 60;}
```

```
    wake_MINUTE = wake_MINUTE + wake_interval_MINUTE; if(wake_MINUTE >=
60) {wake_MINUTE = wake_MINUTE - 60;}
    wake_HOUR   = wake_HOUR   + wake_interval_HOUR;   if(wake_HOUR   >=
24) {wake_HOUR   = wake_HOUR   - 24;}


    // flags[] angir hvilke komponenter i dato og tid som skal sjekkes
for å gi alarm
    // A1M1 (seconds) (0 to enable, 1 to disable)
    // A1M2 (minutes) (0 to enable, 1 to disable)
    // A1M3 (hour)    (0 to enable, 1 to disable)
    // A1M4 (day)     (0 to enable, 1 to disable)
    // DY/DT          (dayofweek == 1/dayofmonth == 0)
    //Sjekker {sek, min, timer, dag, dag i uke/måned}
    uint8_t flags[5] = { 0, 1, 1, 1, 1 };


    // Sett Alarm1
    DS3231_set_a1(wake_SECOND, wake_MINUTE, wake_HOUR, 0, flags);


    // Aktiver Alarm1
    DS3231_set_creg(DS3231_INTCN | DS3231_A1IE);
}
```

E.4.4 Program for å sette opp en gjentatt alarm og bruk av power booster (*setIntAlarm-3*)

Programmet setter opp en alarm som gjentar seg etter et gitt tidsintervall. Når alarmen går, gis mikrokontrolleren og evt. shield-kort, spenning, ellers er kortene helt avslått. I stedet for å bruke monitoren brukes den interne lysdioden (`LED_BUILTIN`) til å indikere når programmet befinner seg i setup()-funksjonen (ett langt blink) og når alarmen detekteres i loop()-funksjonen (fem korte blink).

```
// setIntAlarm-3


// Programmet illustrerer hvordan man kan sette opp et interrupt fra
DS3231
// med jevne intervaller i sekunder (10 - 60), minutter (1 - 60 eller
timer 1 - 24.
// Når match oppnås legges interuptutgangen lav (SQR). I denne har vi
sløyfet Serial.print
#include <Wire.h>
```

```
#include "ds3231.h"

#define BUFF_MAX 256

// Tid for oppvekking
volatile uint8_t wake_HOUR = 0;
volatile uint8_t wake_MINUTE = 0;
volatile uint8_t wake_SECOND = 0;
uint8_t wake_interval_HOUR = 0;
uint8_t wake_interval_MINUTE = 0;
uint8_t wake_interval_SECOND = 20;

// Hyppigheten for oppdatering
int pinRTCINT = 3;

void setup()
{
    delay(1000);
    pinMode(pinRTCINT,INPUT);
    pinMode(LED_BUILTIN,OUTPUT);
    digitalWrite(LED_BUILTIN, HIGH);
    Wire.begin();
    DS3231_init(DS3231_INTCN);
    set_alarm();
    delay(2000);
    digitalWrite(LED_BUILTIN, LOW);
}

void loop()
{
     if(digitalRead(pinRTCINT) == 0)
     {
       set_alarm();
       for(int i=0; i<5; i++)
       {
         digitalWrite(LED_BUILTIN, HIGH);
```

```
        delay(50);
        digitalWrite(LED_BUILTIN, LOW);
        delay(50);
      }
      DS3231_clear_a1f();
    }
}

void set_alarm(void)
{

    // Beregne neste tidspunkt
    wake_SECOND = wake_SECOND + wake_interval_SECOND; if(wake_SECOND >=
60) {wake_SECOND = wake_SECOND - 60;}
    wake_MINUTE = wake_MINUTE + wake_interval_MINUTE; if(wake_MINUTE >=
60) {wake_MINUTE = wake_MINUTE - 60;}
    wake_HOUR   = wake_HOUR   + wake_interval_HOUR;   if(wake_HOUR   >=
24) {wake_HOUR   = wake_HOUR   - 24;}

    // flags[] angir hvilke komponenter i dato og tid som skal sjekkes
for å gi alarm
    // A1M1 (seconds) (0 to enable, 1 to disable)
    // A1M2 (minutes) (0 to enable, 1 to disable)
    // A1M3 (hour)    (0 to enable, 1 to disable)
    // A1M4 (day)     (0 to enable, 1 to disable)
    // DY/DT          (dayofweek == 1/dayofmonth == 0)
    //Sjekker {sek, min, timer, dag, dag i uke/måned}
    uint8_t flags[5] = { 0, 1, 1, 1, 1 };

    // Sett Alarm1
    DS3231_set_a1(wake_SECOND, wake_MINUTE, wake_HOUR, 0, flags);

    // Aktiver Alarm1
    DS3231_set_creg(DS3231_INTCN | DS3231_A1IE);
}
```

Vedlegg F Programmer for bruk av sensorer

Programmene i dette vedlegget er testprogrammer, programmer for kalibrering og bruk av sensorer for måling av væskeparmetere.

F.1 Kalibrering og måling av sensor for oppløst oksygen SEN0237-A

F.1.1 Avlesning og 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);
}
```

F.1.2

```
// 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()
{
  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);
}
```

F.2 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
// kaliberingsverdier 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 NTU-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 temeraturendringer
  long sensorValue  = 0;   // Les av analog inngang fra pinne A0:
  float UD_mid      = 0;   // Middelverdi av målt spenning
  float UD_mid_volt = 0;   // Middelverdi av målt spenning

  for(int i=0; i<100; i++)
  {
     sensorValue = sensorValue + analogRead(A0);
  }
  UD_mid = float(sensorValue/100);          // Finner middelverdien

  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);
}
```

F.3 Kalibrering og måling Turbiditet SEN-0189

F.3.1 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 spen-
ning (0 - 4.1 +/- 0,3 V ved 0 NTU (Rent vann)):
  Serial.println(voltage); // Skriv ut den avleste verdien:
  delay(500);
}
```

F.3.2 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
}
```

F.3.3 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
// kaliberingsverdier 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;   // Middelverdi av målt spenning
  float UD_mid_volt = 0;   // Middelverdi av målt spenning

  for(int i=0; i<100; i++)
  {
     sensorValue = sensorValue + analogRead(A0);
  }

  UD_mid = 2.0 * float(sensorValue/100);          // Finner middelverdien
  UD_mid_volt = UD_mid * (3.3 / 4096.0);  //
  // Konverterer den analoge avlesningen (som går fra 0-4095) til en spen-
ning (0 - 4.1 +/- 0,3 V ved 0 NTU (Rent vann)):

  Serial.print("Maalt midlere spenning: "); // Skriv ut den avleste ver-
dien 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);
}
```

F.4 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;
}
```

F.5 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
        }
      }//if
    }//for
    avg = (double)amount/(number-2);
  }//if
```

```
  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));
  }
}
```

Vedlegg G Løsningsforslag

G.1 Program som leser av og sender ENV-data til serveren (Sea-buoy-2)

Programmet sender over avleste sensordata fra ENV-kortet og sender disse til serveren. GPS-data og vanntemperatur er foreløpig kun dummy-konstanter.

```
/*
  Web client


  Dette kodeeksempelet kober opp mot en webserver ved hjelp av et MKR NB
1500 kort.
  Denne koden legger dataene på URL-adressen "http://
sensor.marin.ntnu.no"
  samtidig som den skriver måledatene til Seriemonitoren


  Circuit:
   - MKR NB 1500 board
   - Antenna
   - SIM-kort


  Utviklet av Tom Igor 08.03.12
  Modifisert av Håvard Holm 2021
  Modifisert av Nils Kr. Rossing 08.09.22

*/

// libraries
#include <MKRNB.h>

// PIN Number
const char PINNUMBER[]     = "";

// initialize the library instance
NBClient client;
GPRS gprs;
NB nbAccess;
```

```
// URL, path og port
char server[] = "sensor.marin.ntnu.no";
char path[] = "/cgi-bin/tof.cgi?";
int port = 80; // port 80 er default for HTTP

// Globale variabler

void setup() {
  char buffer[256];

  Serial.begin(9600);      // Initialiser monitoren
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }
  Serial.println("Starter Arduino web-client.");

  boolean connected = false; // Status oppkobling

  // Koble opp til GPRS nettverket med APN, login og passord
  while (!connected)
  {
   if ((nbAccess.begin("", true, true) == NB_READY) && (gprs.attachGPRS()
== GPRS_READY))
    {
      connected = true;
    }
    else
    {
      Serial.println("Ikke tilkoblet");
      delay(1000);
    }
  }

  Serial.println("Kobler til...");

// Om det oppnås forbindelse rapporter tilbake til monitor:
```

```
  if (client.connect(server, port))
  {
    Serial.println("Tilkoblet");

    client.print("GET "); // Gjør et HTTP request:
    sprintf(buffer, "/tof/tof.cgi?nils,lat=%1f,lon=%1f", 63.426269,
10.454045); // Leggmåleverdier inn i bufferet
    client.print(buffer);
    client.println(" HTTP/1.1");
    client.print("Host: ");
    client.println(server);

    client.println("Connection: close");
    client.println();
  }
  else
  {
    Serial.println("Mislykket tilkobling"); // Dersom tilkobling
mislyktes
  }
}

void loop()
{
  // Dersom det er innkommende data fra serveren, les dem og skriv dem ut
  if (client.available())
  {
    Serial.print((char)client.read());
  }

  // Dersom servene er frakoblet stopp klienten:
  if (!client.available() && !client.connected())
  {
    Serial.println();
    Serial.println("Kobler fra klienten");
    client.stop();
```

```
  // do nothing forevermore:
    for (;;)
      ;
  }
}
```

G.2 Program som leser av og sender GPS- og ENV-data til server (Sea-buoy-3)

Programmet er testet og synes å fungere tilfredsstillende så fremt GPS-kortet er koblet opp med kabel og ikke som shield-kort montert på toppen av MKR NB 1500 og MKR ENV-kortet. Vanntemperaturen er foreløpig dummy-data.

```
/*
  Sea-buoy-3


  Dette kodeeksempelet kobler opp mot en webserver ved hjelp av et MKR
NB 1500 kort.
  Koden henter data fra både ENV-kortet, GPS-kortet og lager noe dummy-
data for vanntemp.
  Koden legger dataene på URL-adressen "http://sensor.marin.ntnu.no" sam-
tidig som den skriver måledatene til Seriemonitoren. Koden kjøres 5
ganger før den
  avsluttes.


  Circuit:
   - MKR NB 1500 board
   - MKR ENV
   - MKR GPS
   - Antenna
   - SIM-kort


  Utviklet av Tom Igor 08.03.12
  Modifisert av Håvard Holm 2021
  Modifisert av Nils Kr. Rossing 08.09.22

*/


// libraries
```

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

// PIN Number
const char PINNUMBER[]     = "";

// initialize the library instance
NBClient client;
GPRS gprs;
NB nbAccess;

// URL, path og port
char server[] = "sensor.marin.ntnu.no";
char path[] = "/cgi-bin/tof.cgi?";
int port = 80; // port 80 er default for HTTP

// Deklarasjon av sensorvariabler fra ENV-kort
float temperature = 0;
float humidity    = 0;
float pressure    = 0;
float illuminance = 0;

// Deklarer sesnsorvariable fra vannsensorer
float w_temperature = 0;

// Deklarasjon av sensorvariabler fra GPS-kort
float latitude;
float longitude;
float altitude;
float speed;
unsigned long epochTime;
int   satellites;
long  no = 0;

// Deklarasjon av buffer for dataoverføring
```

```
char buffer[256];
boolean connected = false; // Status oppkobling

void setup() {

  Serial.begin(9600);      // Initialiser monitoren
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }
  Serial.println("Sea-buoy-3");

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

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

void loop()
{
  for(int i=1; i<6; i++)
  {
  readGPSdata();           // Les GPS-posisjon mm
  readENVdata();           // Les ENV-målinger
  readWaterData();         // Les av sensorer i vann
  no=i;
  sprintf(buffer, "Gr-no,
no=%d,time=%d,lon=%.6f,lat=%.6f,C_luft=%.1f,C_vann=%.1f,hum=%.1f,pres=%
.1f,light=%.1f", no, epochTime, longitude, latitude, temperature, w_tem-
perature, humidity, pressure, illuminance); // Legg måleverdier inn i
bufferet
  Serial.println(buffer);
```

```
  connectToGPRS();   // Koble opp til GPRS nettverket med APN, login og
passord
  Serial.print("Kobler til og overfører data: ");
  Serial.println(no);
  connectToServer(); // Om det oppnås forbindelse overføres data
  checkServerMessage();

  delay(5000);
  }
  for(;;) // Stop programmet
  ;
}

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);
    }
  }
}

void connectToServer()
{
```

```
// Koble opp til server og send over data, varsler om oppkobling mislykkes

  if (client.connect(server, port))
  {
    Serial.println("Tilkoblet server");

    client.print("GET "); // Gjør et HTTP request:
    sprintf(buffer, "/cgi-bin/tof.cgi?Gr-
no,no=%d,time=%d,lon=%.6f,lat=%.6f,C_luft=%.1f,C_vann=%.1f,hum=%.1f,pre
s=%.1f,light=%.1f", no, epochTime, longitude, latitude, temperature,
w_temperature, humidity, pressure, illuminance); // Legg måleverdier inn
i bufferet
    client.print(buffer);
    client.println(" HTTP/1.1");
    client.print("Host: ");
    client.println(server);

    client.println("Connection: close");
    client.println();
  }
  else
  {
    Serial.println("Mislykket tilkobling til server"); // Dersom tilkob-
ling mislyktes
  }
}

void checkServerMessage()
{
   // Dersom det er innkommende data fra serveren, les dem og skriv dem ut
  if (client.available())
  {
    Serial.print((char)client.read());
  }

  // Dersom serveren er frakoblet stopp klienten:
  if (!client.available() && !client.connected())
```

```
  {
    Serial.println();
    Serial.println("Kobler fra klienten");
    client.stop();

  // do nothing forevermore:
    for (;;)
      ;
  }
}

void readGPSdata()
{
    while (!GPS.available()) {}
    // read GPS values
    latitude   = GPS.latitude();
    longitude  = GPS.longitude();
    altitude   = GPS.altitude();
    speed      = GPS.speed();
    epochTime  = GPS.getTime();
    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();
}

void readENVdata()
{
  Serial.println("Leser miljo-data");
  temperature = ENV.readTemperature();
  humidity    = ENV.readHumidity();
  pressure    = ENV.readPressure() * 10; // Fra kPa til mBar
  illuminance = ENV.readIlluminance();

  // Skriv ut hver av sensor måleverdiene

  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(" mBar");

  Serial.print("Illuminance = ");
  Serial.print(illuminance);
  Serial.println(" lx");

  // print an empty line
  Serial.println();
}
```

```
void readWaterData()
{
  Serial.println("Leser vanntemperatur");
  w_temperature = 12.0;
}
```

G.3 Program leser av og sender GPS-, ENV- og vanntemperatur til serveren (Sea-buoy-4)

Programmet inkluderer i tillegg til innhenting av data fra sensor- og GPS-kort også måling av vanntemperatur.

```
/*
  Sea-buoy-4


  Dette kodeeksempelet kobler opp mot en webserver ved hjelp av et MKR
NB 1500 kort.
  Koden henter data fra både ENV-kortet, GPS-kortet og DS18B20 som måler
vanntemp.
  Koden legger dataene på URL-adressen "http://sensor.marin.ntnu.no"
  samtidig som den skriver måledatene til Seriemonitoren. Koden kjøres 5
ganger før den
  avsluttes.


  Circuit:
   - MKR NB 1500 board
   - MKR ENV
   - MKR GPS
   - DS18B20 (vanntemp)
   - Antenne
   - SIM-kort


  Utviklet av Tom Igor 08.03.12
  Modifisert av Håvard Holm 2021
  Modifisert av Nils Kr. Rossing 08.09.22

*/


// libraries
```

```
#include <MKRNB.h>
#include <Arduino_MKRENV.h>
#include <Arduino_MKRGPS.h>
#include <DS18B20.h>
#include <OneWire.h>

// Lager objekt for måling av vanntemperatur
DS18B20 ds(0); //Sensoren er koblet til pinne D0
uint8_t address[] = {40, 250, 31, 218, 4, 0, 0, 52};
uint8_t selected;

// PIN Number
const char PINNUMBER[]     = "";

// initialize the library instance
NBClient client;
GPRS gprs;
NB nbAccess;

// URL, path og port
char server[] = "sensor.marin.ntnu.no";
char path[] = "/cgi-bin/tof.cgi?";
int port = 80; // port 80 er default for HTTP

// Deklarasjon av sensorvariabler fra ENV-kort
float temperature = 0;
float humidity    = 0;
float pressure    = 0;
float illuminance = 0;

// Deklarer sesnsorvariable fra vannsensorer
float w_temperature = 0;

// Deklarasjon av sensorvariabler fra GPS-kort
float latitude;
float longitude;
```

```
float altitude;
float speed;
unsigned long epochTime;
int   satellites;
long  no = 0;

// Deklarasjon av buffer for dataoverføring
char buffer[256];
boolean connected = false; // Status oppkobling

void setup() {

  Serial.begin(9600);      // Initialiser monitoren
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }

  Serial.println("Sea-buoy-4");

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

  // Initialiserer MKR GPS-kortet
  if (!GPS.begin()) {
    Serial.println("Failed to initialize GPS!");
    while (1);
  }

  // Initialiserer DS18B20 - måling av vanntemperatur
  selected = ds.select(address);
}

void loop()
```

```
{
  for(int i=1; i<6; i++)
  {
  readGPSdata();           // Les GPS-posisjon mm
  readENVdata();           // Les ENV-målinger
  readWaterData();         // Les av sensorer i vann
  no=i;
  sprintf(buffer, "Gr-no,
no=%d,time=%d,lon=%.6f,lat=%.6f,C_luft=%.1f,C_vann=%.1f,hum=%.1f,pres=%
.1f,light=%.1f", no, epochTime, longitude, latitude, temperature, w_tem-
perature, humidity, pressure, illuminance); // Legg måleverdier inn i
bufferet
  Serial.println(buffer);


  connectToGPRS();   // Koble opp til GPRS nettverket med APN, login og
passord
  Serial.print("Kobler til og overfører data: ");
  Serial.println(no);
  connectToServer(); // Om det oppnås forbindelse overføres data
  checkServerMessage();


  delay(5000);
  }
  for(;;) // Stop programmet
  ;
}


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);
    }
  }
}

void connectToServer()
{
// Koble opp til server og send over data, varsler om oppkobling mislykkes

  if (client.connect(server, port))
  {
    Serial.println("Tilkoblet server");

    client.print("GET "); // Gjør et HTTP request:
    sprintf(buffer, "/cgi-bin/tof.cgi?Gr-
no,no=%d,time=%d,lon=%.6f,lat=%.6f,C_luft=%.1f,C_vann=%.1f,hum=%.1f,pre
s=%.1f,light=%.1f", no, epochTime, longitude, latitude, temperature,
w_temperature, humidity, pressure, illuminance); // Legg måleverdier inn
i bufferet
    client.print(buffer);
    client.println(" HTTP/1.1");
    client.print("Host: ");
    client.println(server);

    client.println("Connection: close");
    client.println();
  }
  else
  {
    Serial.println("Mislykket tilkobling til server"); // Dersom tilkob-
ling mislyktes
```

```
  }
}

void checkServerMessage()
{
   // Dersom det er innkommende data fra serveren, les dem og skriv dem ut
  if (client.available())
  {
    Serial.print((char)client.read());
  }

  // Dersom serveren er frakoblet stopp klienten:
  if (!client.available() && !client.connected())
  {
    Serial.println();
    Serial.println("Kobler fra klienten");
    client.stop();

  // do nothing forevermore:
    for (;;)
      ;
  }
}

void readGPSdata()
{
    while (!GPS.available()) {}
    // read GPS values
    latitude   = GPS.latitude();
    longitude  = GPS.longitude();
    altitude   = GPS.altitude();
    speed      = GPS.speed();
    epochTime  = GPS.getTime();
    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();
}

void readENVdata()
{
  Serial.println("Leser miljo-data");
  temperature = ENV.readTemperature();
  humidity    = ENV.readHumidity();
  pressure    = ENV.readPressure() * 10; // Fra kPa til mBar
  illuminance = ENV.readIlluminance();

  // Skriv ut hver av sensor måleverdiene

  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(" mBar");

  Serial.print("Illuminance = ");
  Serial.print(illuminance);
  Serial.println(" lx");

  // print an empty line
  Serial.println();
}

void readWaterData()
{
  w_temperature = ds.getTempC();

  Serial.print("Vanntemperature er: ");
  Serial.print(ds.getTempC());
  Serial.println(" C");
}
```

G.4 Program leser av og sender GPS-, ENV- og vanntemperatur til serveren (Sea-buoy-5A) i kortformat

Programmet er testet og synes å fungere tilfredsstillende. ENV-kortet er plugget inn i MKR NB 1500 mikrokontrolleren, mens mikrokontrollerkortet er plugget ned i sensorkortet for måling av vanntemperaturen. I tillegg er GPS-mottakeren koblet til I^2C-bussen.

```
/*
  Sea-buoy-5A


  Dette kodeeksempelet kobler opp mot en webserver ved hjelp av et MKR
NB 1500 kort.

  Koden henter data fra både ENV-kortet, GPS-kortet og DS18B20 som måler
vanntemp.

  Koden legger dataene på URL-adressen "http://sensor.marin.ntnu.no" sam-
tidig som den skriver måledatene til Seriemonitoren. Koden kjøres helt
til den stoppes.

  Data skrivs inn kunn skilt med komma og med topptekst.
```

```
  Circuit:
   - MKR NB 1500 board
   - MKR ENV
   - MKR GPS
   - DS18B20 (vanntemp)
   - Antenne
   - SIM-kort

  Utviklet av Tom Igor 08.03.12
  Modifisert av Håvard Holm 2021
  Modifisert av Nils Kr. Rossing 08.09.22

*/

// libraries
#include <MKRNB.h>
#include <Arduino_MKRENV.h>
#include <Arduino_MKRGPS.h>
#include <DS18B20.h>
#include <OneWire.h>

// Lager objekt for måling av vanntemperatur
DS18B20 ds(0); //Sensoren er koblet til pinne D0
uint8_t address[] = {40, 250, 31, 218, 4, 0, 0, 52};
uint8_t selected;

// PIN Number
const char PINNUMBER[]     = "";

// initialize the library instance
NBClient client;
GPRS gprs;
NB nbAccess;

// URL, path og port
```

```
char server[] = "sensor.marin.ntnu.no";
char path[] = "/cgi-bin/tof.cgi?";
int port = 80; // port 80 er default for HTTP

// Deklarasjon av sensorvariabler fra ENV-kort
float temperature = 0;
float humidity    = 0;
float pressure    = 0;
float illuminance = 0;

// Deklarer sesnsorvariable fra vannsensorer
float w_temperature = 0;

// Deklarasjon av sensorvariabler fra GPS-kort
float latitude;
float longitude;
float altitude;
float speed;
unsigned long epochTime;
int   satellites;
long  no = 0;

// Deklarasjon av buffer for dataoverføring
char buffer[256];
boolean connected = false; // Status oppkobling

void setup() {

  Serial.begin(9600);      // Initialiser monitoren
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }

  Serial.println("Sea-buoy-5A");

  // Initialiserer MKR ENV-kortet
```

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

  // Initialiserer MKR GPS-kortet
  if (!GPS.begin()) {
    Serial.println("Failed to initialize GPS!");
    while (1);
  }

  // Initialiserer DS18B20 - måling av vanntemperatur
  selected = ds.select(address);

  // Skriv ut topptekst
  connectToGPRS();
  Serial.println("Kobler til og overfører topptekst");
  connectToServerWriteHeaderText();
  checkServerMessage();
  delay(2000);
}

void loop()
{
  no++;
  readGPSdata();           // Les GPS-posisjon mm
  readENVdata();           // Les ENV-målinger
  readWaterData();         // Les av sensorer i vann

  sprintf(buffer, "Gr-no,
no=%d,time=%d,lon=%.6f,lat=%.6f,C_luft=%.1f,C_vann=%.1f,hum=%.1f,pres=%
.1f,light=%.1f", no, epochTime, longitude, latitude, temperature, humi-
dity, pressure, illuminance, w_temperature); // Legg måleverdier inn i
bufferet
  Serial.println(buffer);
```

```
  connectToGPRS();   // Koble opp til GPRS nettverket med APN, login og
passord
  Serial.print("Kobler til og overfører data: ");
  Serial.println(no);
  connectToServer(); // Om det oppnås forbindelse overføres data
  checkServerMessage();

  delay(5000);
}

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);
    }
  }
}

void connectToServer()
{
// Koble opp til server og send over data, varsler om oppkobling mislykkes

  if (client.connect(server, port))
```

```
  {
    Serial.println("Tilkoblet server");

    client.print("GET "); // Gjør et HTTP request:
    sprintf(buffer, "/cgi-bin/tof.cgi?Gr-
no,%d,%d,%.6f,%.6f,%.1f,%.1f,%.1f,%.1f,%.1f", no, epochTime, longitude,
latitude, temperature, humidity, pressure, illuminance, w_temperature);
// Legg måleverdier inn i bufferet
    client.print(buffer);
    client.println(" HTTP/1.1");
    client.print("Host: ");
    client.println(server);

    client.println("Connection: close");
    client.println();
  }
  else
  {
    Serial.println("Mislykket tilkobling til server"); // Dersom tilkob-
ling mislyktes
  }
}

void checkServerMessage()
{
   // Dersom det er innkommende data fra serveren, les dem og skriv dem ut
  if (client.available())
  {
    Serial.print((char)client.read());
  }

  // Dersom serveren er frakoblet stopp klienten:
  if (!client.available() && !client.connected())
  {
    Serial.println();
    Serial.println("Kobler fra klienten");
    client.stop();
```

```
  }
}

void readGPSdata()
{
    while (!GPS.available()) {}
    // read GPS values
    latitude   = GPS.latitude();
    longitude  = GPS.longitude();
    altitude   = GPS.altitude();
    speed      = GPS.speed();
    epochTime  = GPS.getTime();
    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();
}

void readENVdata()
{
```

```
  Serial.println("Leser miljo-data");
  temperature = ENV.readTemperature();
  humidity    = ENV.readHumidity();
  pressure    = ENV.readPressure() * 10; // Fra kPa til mBar
  illuminance = ENV.readIlluminance();

  // Skriv ut hver av sensor måleverdiene

  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(" mBar");

  Serial.print("Illuminance = ");
  Serial.print(illuminance);
  Serial.println(" lx");

  // print an empty line
  Serial.println();
}

void readWaterData()
{
  w_temperature = ds.getTempC();

  Serial.print("Vanntemperature er: ");
  Serial.print(ds.getTempC());
  Serial.println(" C");
}
```

```
void connectToServerWriteHeaderText()
{
  // Koble opp til server og sender over topptekst

  if (client.connect(server, port))
  {
    Serial.println("Tilkoblet server");

    client.print("GET "); // Gjør et HTTP request:
    sprintf(buffer, "/tof/tof.cgi?Gr-1,Nr.,Tid,Lengdegrad,Bredde-
grad,Lufttemp.,Luftfukt.,Lufttrykk,Vanntemp."); // Legg overskrift inn i
bufferet
    client.print(buffer);
    client.println(" HTTP/1.1");
    client.print("Host: ");
    client.println(server);

    client.println("Connection: close");
    client.println();
  }
  else
  {
    Serial.println("Mislykket tilkobling til server"); // Dersom tilkob-
ling mislyktes
  }
}
```

Prosjektet “Miljøovervåking med havbøye” 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. 4G 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. Det ble i den forbindelse utført en mulighetsstudie ved Skolelaboratoriet som ble videreført av to studenter samme sommer.

En kan tenke seg flere løsninger ved at man i større eller mindre grad bruker etablerte tjenester som den f.eks. Telenor tilbyr, 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.

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 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

ISBN
Rev. 3.11 - 20.12.22

Trykk: NTNU Grafisk senter

NTNU
Trondheim
Institutt for fysikk
Skolelaboratoriet
for matematikk, naturfag og teknologi

Tlf. 73 55 11 43
https://www.ntnu.no/skolelab