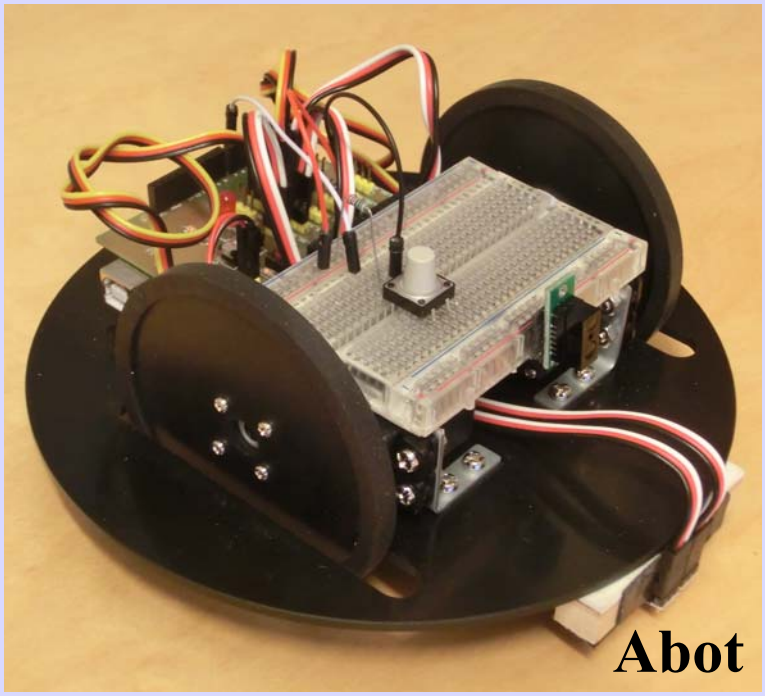
Prøvetrykk 2.2

Nils Kr. Rossing

Arduino ressurshefte – Atmel



NTNU
Trondheim

Program for lærerutdanning

Skolelaboratoriet
for matematikk, naturfag og teknologi

Desember 2014

Arduino ressurshefte – Atmel

Arduino ressurshefte - Atmel

Trondheim 2014

Bidragsytere:
Nils Kr. Rossing, *(nils.rossing@plu.ntnu.no)*Skolelaboratoriet ved NTNU

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

Tekst og bilder Nils Kr. Rossing, Skolelaboratoriet ved NTNU

Faglige spørsmål rettes til:
Skolelaboratoriet for matematikk naturfag og teknologi, NTNU
v/Nils Kr. Rossing, 73 55 11 91
nils.rossing@plu.ntnu.no

Realfagbygget, Høgskoleringen 5
7491 Trondheim

Skolelaboratoriet
Telefon: 73 55 11 43
Telefaks: 73 55 11 40
http://www.ntnu.no/skolelab/

Prøvetrykk 2.2, Rev 2.2 – 13.12.14

NTNU
Det skapende universitet

Prosjektet er gjennomført i et samarbeid mellom
Atmel Norge AS
og
Skolelaboratoriet for Matematikk, Naturfarg og Teknologi ved NTNU

Arduino ressurshefte – Atmel

Nils Kr. Rossing

Torgeir Rossing monterer prototyproboter (Mr. Abot) på vekstedet til sin far (Foto: N.K. Rossing)

Skolelaboratoriet for matematikk, naturfag og teknologi, NTNU

Forord

Vinteren 2012–13 tok ***Ole Petter Håkegård***, Heimdal videregående skole, ***Frode Øren*** og ***Nils Kr. Rossing*** Skolelaboratoriet ved NTNU kontakt med Atmel Norge med spørsmål om de var interessert i å samarbeide om, og støtte et prosjekt i videregående skole knyttet til utvikle av et undervisningsopplegg knyttet til bruk av mikrokontroller konseptet Arduino i faget Teknologi og Forskningslære (ToF). Etter kort tid ble vi enige om at de skulle støttet utvikling og gjennomføring av et lærerkurs og et elevverksted for den nevnte gruppen.

Med utgangspunkt i Sparkfun Inventor's kit ble det i begynnelsen av mars holdt et todagers lærerkurs og i begynnelsen av juni ble det også gjennomført et en dags lærerkurs for ca. 20 elever ved Byåsen videregående skole. Dessuten ble det lånt ut et klassesett av utstyret til Kongsberg videregående skole.

Erfaringene fra disse samlingene var så positive at det ble besluttet å tilby et tredagers kurs høsten 2013 for lærere i ungdomsskolen i faget Teknologi i Praksis. Videre tilbys det et endags elevverksted for ToF- og TiP- elever. Opplegget ble dessuten presentert på en fagdag for TiP-lærere i Trondheim 13. september 2013 og på naturfagkonferansen i Oslo i okt. 2013.

Dessuten ble det høsten 2013 engasjert en IAESTE student ved Atmel Norge AS for å arbeide med et undervisningsopplegg knyttet til Atmel roboten ABOT.

Skolelaboratoriet ved NTNU vil takke Atmel for den positive innstilling firmaet har vist ved å støtte prosjektet både faglig og økonomisk og til ***Torgeir Bjørnvold*** som har gitt oss tillatelse til å benytte bilder av mekanisk oppkobling av Abot. Høsten 2013 engasjerte de IEASTE studenten ***Adam Gajda*** som laget et komplett undervisningsopplegg for roboten Mr. Abot. Den forbindelse laget han et shield-kort og programvare for å linjefølging. Dette er delvis integrert i dette heftet. Videre har ***Torgeir Rossing*** monter 8 prototyper av Mr. Abot for spredning til de 8 regionale vitensentrene i Norge. En tak til ***Åge L. Eriksen*** som bygte opp Vitensenterets egne roboter våren 2014 slik at vi nå kan tilby en aktivitet.

Nils Kr. Rossing

Skolelaboratoriet ved NTNU
Desember 2014

Innhold

1 Innledning 13

2 Bakgrunnen 15

2.1 Mikroprosessorer og mikrokontrollere 15

2.1.1 Arduino og AVR mikrokontrollere - litt historikk 17

2.1.2 Generelt rammeverk 19

3 Grunnleggende byggeklosser 21

3.1 Komponentoversikter 21

3.1.1 "Sparkfun Inventors Kit" 21

3.1.2 Abot 22

3.2 Komponenter – beskrivelse 23

3.2.1 Koblingsbrettet 23

3.2.2 Motstander 24

3.2.3 Sensorer 25

3.2.4 Halvledere 27

3.2.5 Aktuatorer 28

3.2.6 Arduino – mikrokontrollerkortet 30

4 Abot - oppbygging 33

4.1 Mekanisk byggebeskrivelse 33

4.1.1 Montering av hjulene 33

4.1.2 Monter braketter på motorene 34

4.1.3 Montering av halehjulet 34

4.2 Elektrisk oppkobling 35

4.2.1 Oppkobling med shield-kort og 6 V batteripakke 35

4.2.2 Oppkobling med Arduino og 9V batteripakke 44

5 Programmer og programmering 47

5.1 "Sparkfun's Inventors kit" 47

5.2 Installasjon av Arduino programeditor 49

5.2.1 Arduino programeditor 49

5.2.2 Grunnleggende bruk av programeditoren 51

5.3 Programstruktur 52

5.4 Viktige kommandoer 52

5.4.1 Generelle kommandoer 52

5.4.2 Avlesning av sensorer 55

5.5 Programmering av servomotorer 55

6 Hjelpeprogrammer 63

6.1 Fritzing 63

6.1.1 Bruksområde 63

6.1.2 Installasjon 63

6.1.3 Introduksjon til bruk 63

6.1.4 Oppbygging på koblingsbrett (Breadboard) 64

6.1.5 Overføring til koblingsskjema 66

6.1.6 Trykte kretskort 67

6.2 Scratch for Arduino (S4A) 68

6.2.1 Installasjon 69

6.2.2 Mitt første program i S4A (tutorial) 71

6.2.3 Bruk av porter ved bruk av S4A 72

6.3 Ardublocks 73

6.3.1 Installasjon 73

6.3.2 Bruk av Ardublocks 75

7 Oppgavesamlinger 79

7.1 Introduksjonsoppgaver 79

7.1.1 Blinkende lysdiode - Øving 1 A 79

7.1.2 Morsesignalering - Øving 1 B, C, D og E 80

7.1.3 To blinkende lysdioder 81

7.2 Nybegynner kit 82

7.2.1 “Sparkfun Inventors Kit V3” 82

7.2.2 “ The Arduino starter kit” 84

7.3 Halvåpne oppgaver 85

7.3.1 Halvåpen oppgave – Lag en batteritester 85

7.3.2 Halvåpen oppgave – Lag en automatisk blomstervanner 86

7.3.3 Halvåpen oppgave – Lag en automatisk vannvarmer 87

7.3.4 Halvåpen oppgave – Lag en elektronisk terning 88

7.3.5 Halvåpen oppgave – Lag et trafikklys 88

7.3.6 Halvåpen oppgave - Lag et kolorimeter 90

7.3.7 Turbidimeter 94

7.3.8 Lag en stoppeklokke 94

7.4 Oppgaver med Abot – en liten robot 95

7.4.1 Mr. Abot styrt med laserpeker 95

7.4.2 Utforskende oppgaver 97
7.4.3 Problemløsningsoppgaver 99
7.4.4 Mr. Abot følger en sort tape 101
7.4.5 Banen 109
8 Referanser 113
Vedlegg A Organisering av undervisning 114
A.1 Introduksjon for lærere 114
A.2 Introduksjon av Mr. Abot for studenter ved NTNU 115
A.3 Elevaktivitet ved Vitensenteret 118
A.4 Elevverksted ved Skolelaboratoriet 119
A.5 Lærerkurs over 2 dager 122
Vedlegg B Løsningsforslag på introduksjonsoppgaver 126
B.1 Opprinnelige oppgave 126
B.2 Innføring av variabler 126
B.3 Kode som blinker SOS 127
B.4 SOS med variabel hastighet 129
B.5 SOS med lys og lyd 131
B.6 SOS med bruk av funksjoner 133
Vedlegg C Løsningsforslag på halvåpne oppgaver 137
C.1 Batteritester 137
C.2 Automatisk blomstervanner 142
C.3 Vannvarmer med termostat 145
C.4 Elektronisk terning 148
C.5 Trafikklys 150
C.6 Løsningsforslag kolorimeter (turbidimeter) 160
C.7 Løsningsforslag for oppgaven “Lag en stoppeklokke” 161
Vedlegg D Løsningsforslag til oppgaver til Mr. Abot 165
D.1 Styring av Mr. Abot med laserpeker 165
Vedlegg E Informasjons-ark 168
E.1 Kort presentasjon av Mr. Abot 168
Vedlegg F Fargekoding av motstander 183
Vedlegg G Læreplaner 184
G.1 Teknologi og Forskningslære 1 184
G.2 Teknologi og Forskningslære 2 185

G.3 Teknologi i Praksis (TiP) - ungdomsskolen 186
G.4 Forskning i Praksis (FiP) - Ungdomsskolen 187

1 Innledning

Heftet er ikke ment som noen lærebok, men som et ressurshefte for den som ønsker å anvende Arduino som et undervisningsopplegg i faget Teknologi og forskningslære (ToF) i videregående skole eller i Teknologi i praksis (TiP) i ungdomsskolen. Heftet vil gradvis bli utviklet etter som tilbudet utvides og vil både inneholde teknisk bakgrunnstoff og erfaringer fra de tiltakene som blir gjennomført.

I tillegg til tekniske beskrivelser har vi også lagt inn et kapittel som viser hvordan en kan introdusere temaet for lærere, hvordan et verksted for elever kan legges opp og opplegg for et todagers lærerkurs. Dette er kun forslag som viser hvordan Skolelaboratoriet presentert dette stoffet i den innledende fasen av dette prosjektet.

Et betimelig spørsmål er hvorfor elever, enten det er i ungdomsskolen eller i videregående skole skal lære å programmere mikrokontrollere. Greier de seg ikke godt uten? Det finnes få eller ingen kompetansemål i læreplanene for grunnskolen eller i videregående skole som uttrykker at elevene skal lære programmering. Det nærmeste vi kommer er:

> *... bruke sensorer og styringssystemer i forbindelse med forsøk og konstruksjoner.*
> (ToF 1 - "Den unge ingeniøren")

som er hentet fra kompetansemålene for faget ***Teknologi og forskningslære 1***. Selv om programmering av mikrokontrollere med god samvittighet kan legges under dette punktet, så finnes det alternativer.

Når det gjelder valgfaget ***Teknologi i praksis*** i ungdomsskolen, så favner kompetansemålene så vidt at det ikke er noe problem å inkludere mikrokontrollere i et produkt som elevene skal utvikle på bakgrunnav en prdukt spesifikasjon. Kravet er at oppgavene som løses og produktene som utvikles settes inn i den sammenhengen som planen skisserer.

En kan f.eks. spørre: *Er det nødvendig å forstå virkemåte til en mobilteleforn for å bruke den? Nei, men kunnskap ...*

- *... øker muligheten for å vurdere telefonens kvalitets- og bruksverdi*
- *... gir økt evne til å diskutere med leverandører og selgere*
- *... gir økt respekt for og ønske om å ta vare på telefonen og ...*
- *... gir en indre glede over å forstå og mestre teknologi*

På samme måte kan vi også spørre: *Er det viktig å forstå mangfoldet i naturen for å bruke den? Nei, men ...*

- *... gleden over å oppsøke naturen blir rikere*
- *... respekten for naturen og ønske om å bevare den blir større*

- *... det å forstå og mestre livet i naturen gir i seg selv glede*

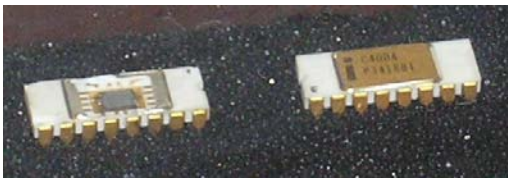
Dessuten ønsker vi å skape nysgjerrighet hos den enkelte, gjerne ved å stimulere til utforskende aktiviteter, som på sikt kan skape en vedvarende fascinasjon og interesse for et fagfelt, og som på sakt kan være en drivkraft og gi indre motivasjon til å forstå mer, hvilket kan føre til læring.

2 Bakgrunnen

2.1 Mikroprosessorer og mikrokontrollere

Vi er vel alle klar over at vi omgir oss med store mengder elektronikk både i skolen, på jobben, i bilen, hjemme, ja overalt hvor vi ferdes. De fleste av oss går rundt bærer på svært avansert elektronikk og kommunikasjonsutstyr. Tenker vi etter så vil de fleste ha med seg en kalkulator som er en liten datamaskin, de vil ha en smarttelefon som ikke bare er ett kommunikasjonshjelpemiddel for sending av tekstmeldinger og telefonsamtaler, men i tillegg tilbyr stadig flere tilleggsfunksjoner som musikkanlegg, navigasjonsutstyr og kart, kamera for opptak av bilder og film, kalendere og arkivsystemer for å holde orden på hverdagen, spill og underholdning, bibliotek med både romaner og lærebøker, og utallige andre funksjoner. I tillegg bærer flere av oss rundt med bærbare PC-er eller nettbrett for større oppgaver.

Figur 2.1 HTC Smartphone



Figur 2.2 Intel mikroprosessor Intel 4004 regnes for den første mikroprosessoren. Lansert 15. nov. 1971

Sentral i alt slikt utstyr er mikroprosessoren som er en miniatyrisert datamaskin som satt på spissen stort sett bare kan addere, sammenligne og flytte på binære tall. Dessuten kan den "Den lille multiplikasjonstabell" for binære tall som er 0 x 0 = 0 og 1 x 0 = 0 og 1 x 1 = 1. I tillegg må den kommunisere med omverdenen ved å omdanne kontinuerlige størrelser, som lyd, lys, temperatur, radiosignaler, gravitasjon osv. til binære tall.

Selv om mikroprosessorer i dag er blitt langt mer avanserte og raskere enn den første mikroprosessoren som Intel lanserte i 1971, så er grunnfunksjonen omtrent den samme.

En mikroprosessor er derfor en temmelig enfoldig og dum elektronisk "duppe dings", men den har noen svært nyttige egenskaper:

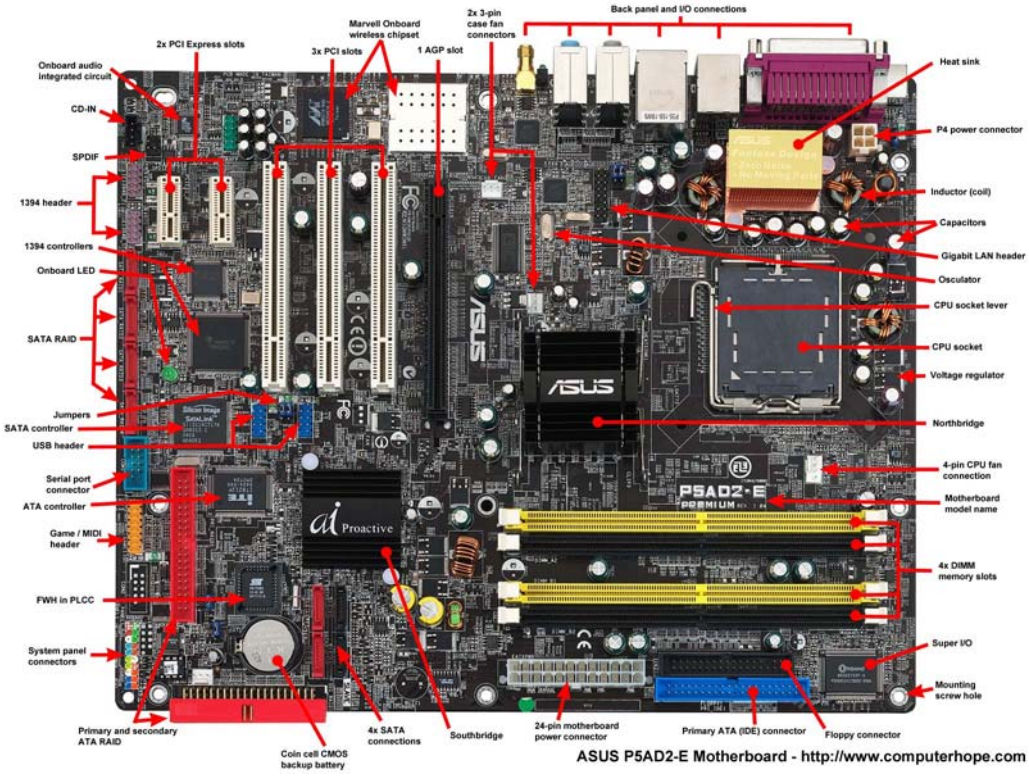
Den arbeider svært fort. I løpet av et sekund kan den utføre mange hundre millioner, ja noen ganger milliarder små enkle oppgaver. Setter man sammen mange nok små enkle oppgaver, kan man gjøre store komplekse oppgaver i løpet av en brøkdel av et sekund.

Den glemmer aldri. Når noe er lagret i hukommelsen til en mikrokontroller så går det ikke "i glemmeboka". Så lenge den virker som den skal så huskes det til "evig tid". Selv om "strømmen går" husker den det som er skrevet inn i hukommelsen.

Den går ikke lei. Dersom den er instruert til å utføre en regneoperasjon riktig, så gjør den det riktig hver gang uten å bomme en eneste gang. Dersom noe går galt så skyldes det som regel at de som har laget mikroprosessoren eller oppskriften (programmet) den følger, har gjort feil eller ikke tenkt på alt. Dessuten er en slik gjenstand sårbar slik at den kan bli ødelagt av gal bruk og uforsiktighet.

Ofte benyttes begrepene mikroprosessor og mikrokontroller. I dette heftet skal vi bare bruke mikrokontrollere. Men hva er egentlig forskjellen?

En mikroprosessor er en meget generell regne- og datamanipuleringsenhet som oftest er plasseres i et miljø med egne kretser for lagring av data og program på utsiden av mikroprosessoren. Likeså foregår all spesialisert kommunikasjon med omverdenen med egne spesialiserte kretser. Åpner du en datamaskin vil du gjerne se et stort kretskort (moderkort) som inneholder mange ulike kretser (se figuren under).

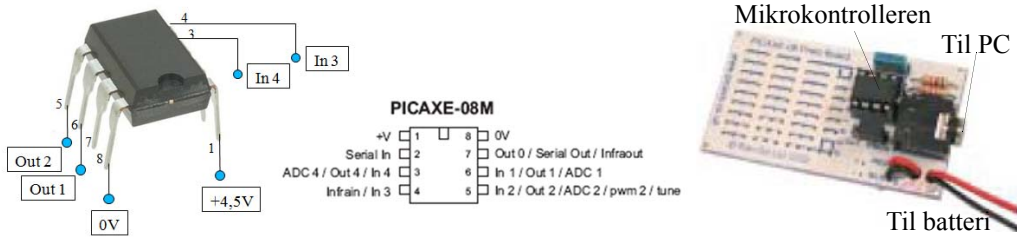


En mikrokontroller inneholder regne- og data-manipuleringsenheten, lager for program og data og det som skal til for å kommunisere med omverdenen, enten det er analoge spenninger eller digitale inn- og utganger, så finnes alt dette inne i den integrerte kretsen. Mikrokontrolleren er med andre ord en komplett datamaskin inkludert i én integrert krets. Prisen en må betale er at den ikke er så kraftig og generell som en mikroprosessor, men som oftest er den kraftig nok til sitt

bruk. Den mangler dessuten muligheter for direkte å koble til PC-tastatur og skjerm. Skal vi kommunisere med en mikrokontroller er vi derfor som oftest avheng av å bruke en PC med tilpasset programvare.

Mens mikroprosessorer gjerne står i datamaskiner, finner vi mikrokontrollere i alt mulig annet utstyr som vaskemaskiner, biler, overvåkning og styringsystemer og annen små-elektronikk. Det er innen dette markedet at Atmel Norge har gjort seg bemerket.

Det finnes en rekke forskjellige familier med mikrokontrollere. Som et eksempel på hvor enkel en mikrokontroller kan være er det på bilde under vist en PIC-kontroller. Så og si en liten datamaskin med 8 bein (til venstre).



Denne vesle kretsen har i alt 5 porter (bein) som kan programmeres som digitale inn- eller utganger, med og uten puls-bredde-modulasjon, analoge innganger, eller innganger som kan fungere som mottakere for IR-kommunikasjon eller som berøringsbrytere. En av utgangene er forberedt for levere toner fra skalaen (i midten på bildet over). Til høyre på figuren over er vist kretsen montert på et lite utviklingskort, med tilkoblingsplugg for batteri og programmering fra PC.

På 1980-tallet oppdaget man at det var mer effektivt å lage mikrokontrollere som hadde få og enkle kommandoer som kunne utføres svart raskt og effektivt, enn å ha mikrokontrollere med mange spesialiserte og ofte kompliserte kommandoer. Denne teknologien kalte man RISK (**R**educe **I**nstruction **S**et **C**omputer) i motsetning til CISK (**C**omlex **I**nstruction **S**et **C**omputer). Atmels mikrokontrollerfamilier er stort sett av RISK typen.

2.1.1 Arduino og AVR mikrokontrollere - litt historikk

Det utstyret som vi skal bruke i undervisningsoppleggene som vil bli omtalt her går under betegnelsen Arduino. La oss se litt på historien bak før vi går videre.

Arduino er et mikrokontrollerkonsept utviklet i den vesle italienske byen *Ivrea* på begynnelsen av dette årtusen. Hensikten var å lage et kontrollerkort som skulle gjøre det enklere og billigere for studenter å lære seg bruk av mikrokontrollere. Det første Arduino-produktet ble utviklet av grunnleggerne ***Massimo Banzi*** og ***David Cuartielles (955–1015)***. De oppkalte prosjektet etter *Arduin of Ivrea* som var den vesle byens historiske heltefigur. Navnet betyr *sterk venn* og burde passe godt for et kraftig kontrollerkort. Det som startet som et lokalt prosjekt for studenter i 2005, hadde i 2010 spredd seg til hele verden. I februar 2010 hadde de solgt 120 000 eksemplarer. Den tilhørende programvaren ble utviklet av studenten ***Hernando Barragán*** ved det lokale universitetet i Ivrea, som en "open source" kode.

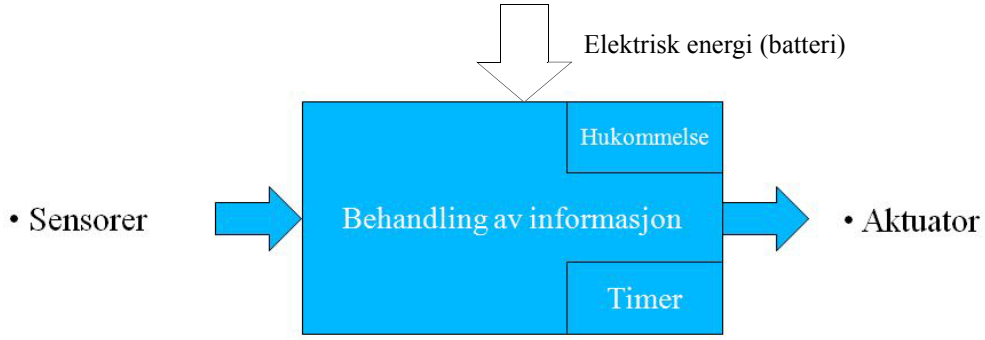
Figur 2.3 Alf Egil Bogen og Vegard Wollan

Kortet var bygget opp omkring ***AVR mikrokontrollere*** fra Atmel (hovedsakelig ATmega8, ATmega168, ATmega328, ATmega1280, and ATmega2560). Kontrollkortet i CanSat anvender ATmega168. Dette er en serie kontrollere som anvender RISK-arkitektur, en svært enkel, men meget effektiv arkitektur. Det er moro å vite at den første kontrolleren i denne serien ble utviklet av studentene ***Alf-Egil Bogen*** og ***Vegard Wollan*** ved NTH på begynnelsen av 1990-tallet. Etter endt studium tok de med seg konseptet inn i firmaet Nordic VLSI (nå NORDIC Semiconductor), hvor det ble videreutviklet. I 1995 gikk de ut av Nordic VLSI og ble snart kjøpt opp av Atmel og driver i dag firmaet Atmel Norge. De sier selv at AVR ikke har noen spesiell betydning, men det er allment akseptert at det opprinnelig sto for **A**lf (Egil Bogen) and **V**egard (Wollan) 's **R**isc processor.

La oss forsøke å sette mikrokontrolleren inn i en sammenheng. I det neste avsnittet skal vi lage et generelt rammeverk som vi kan sette vår elektronikk og mikrokontrollere inn i.

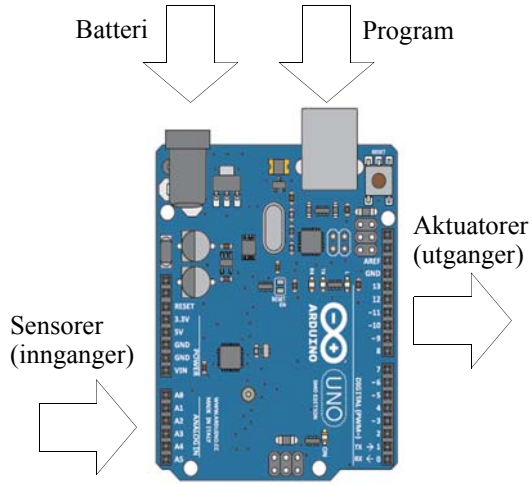
2.1.2 Generelt rammeverk

Elektronisk utstyr, neste uansett hva det er, kan passes inn i følgende beskrivelse (se figuren under).



Utstyret inneholder følgende hoveddeler:

- ***Sensorer***
  Dette er elektroniske enheter som *innhenter* informasjon fra omgivelsene. Eksempler på slike kan være temperatur-, lys-, fuktighets-, bevegelses-, røyksensorer. Det kan også være en enkel bryter av den typen vi bruker til å slå på lyset eller en mikrofon som registrerer lyd eller en uendelighet av andre sensorer.
- ***Aktuatorer***
  Dette er elektroniske enheter som *utfører en oppgave*. Dette kan være en motor som åpner en dør, en lyspære som gir lys, en høyttaler som gir lyd, et varmeelement som gir varme osv. Dvs. en enhet som utfører en aksjon.
- ***Databehandlingsenhet***
  Dette er en elektronisk enhet som kan være svært enkel eller uhyre komplisert og som samler inn informasjon fra omgivelsene ved hjelp av *sensorene*, for så å gi *aktuatorene* beskjed om handling. Sensorene er tilkoblet innganger, mens aktuatorene er tilkoblet utganger. I tillegg tilføres enheten *elektrisk energi* fra batteri eller strømnettet. Noen ganger inneholder enheten en "timer" som holder rede på tiden. Dersom arbeidsoppgavene til enheten er sammensatt må den også inneholde en *oppskrift* (program) som beskriver hvordan den skal behandle informasjonen fra sensorene.



I neste kapittel skal vi skal vi se nærmere på *Sparkfuns Inventors kit* som er den grunnpakken vi har bygget våre undervisningsopplegg på.

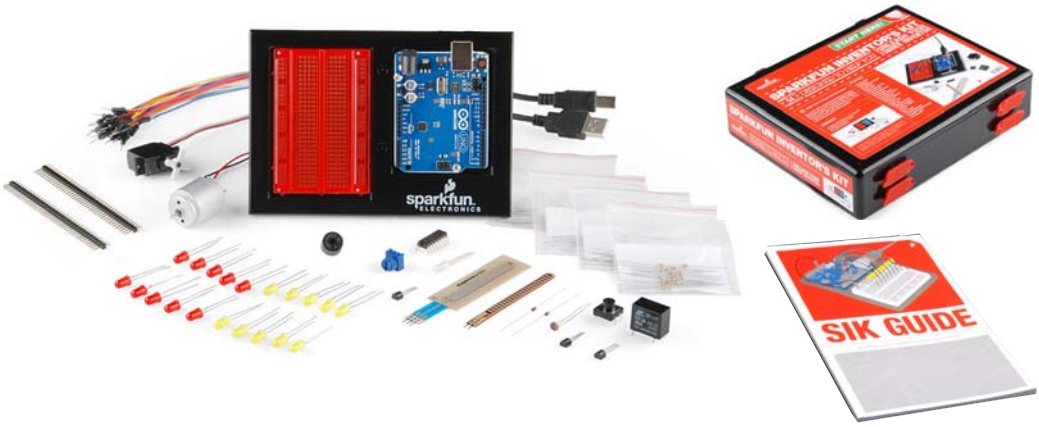
3 Grunnleggende byggeklosser

Dette kapittelet gir en oversikt over de elektroniske byggeklossen vi skal bruke i de følgende undervisningsoppleggene

3.1 Komponentoversikter

3.1.1 "Sparkfun Inventors Kit"

Grunnopplæringen er basert på Sparkfun's nye *Inventors kit*.



Med settet følger et relativt rikholdig utvalg av elektroniske byggeklosser. I dette avsnittet vil vi ganske kort gi informasjon om disse i tillegg til andre aktuelle komponenter. Her er ei liste over innholdet i Sparkfuns Inventors kit:

- 1 stk. Arduino UNO R3
- 1 stk. Plastholder for Arduino og koblingsbrett
- 1 stk. SIK Manual
- 1 stk. Oppbevaringsboks for kittet
- 1 stk Koblingsbrett
- 1 stk Skiftregister – 74HC595
- 2 stk. Transistorer – 2N2222
- 2 stk. Dioder – 1N4148
- 1 stk. DC Motor med ledninger (1,5–3V) – 201-A
- 1 stk. Liten Servo (0–160°) A0090 - 9 g
- 1 stk. Rele 5–12V maks 5A – JZC-11F
- 1 stk. Temperatur sensor – +10mV/K – 0,5 V ved 0 °C – TMP36

- 1 stk. Bøyesensor, Spectra Symbol 25 kOhm v/flat sensor – FS-L-0055-253-ST
- 1 stk. Membran-potensiometer, 100–10kOhm – SP-L-0050-103-ST
- 1 stk. USB kabel, 6’ vanlig A til B USB kabel
- 30 stk. Koblingsledninger, 7” Male/Male
- 1 stk. Fotomotstand, 8 kOhm (10 lux) – 1 MOhm (0 lux) topp ved 540 nm GL5528
- 1 stk. Tri-color LED, Intensitet (RGB): (800, 4000, 900) mcd YSL-R596CR3G4B5C-C10
- 10 stk. Rød lysdiode, 16–18 mA, maks 20 mA, Intensitet 150–200 mcd YSL-R531R3D-D2
- 10 stk. Gul lysdiode16–18 mA, maks 20 mA, Intensitet 40–100 mcd YSL-R341Y3D-D2
- 1 stk. Trimpot med ratt, 10 kOhm
- 1 stk. Magnetisk buzzer 100–10kHz 70–90 dB rel. 20µPa, maks 2048Hz, CEM1203(42)
- 1 stk. Trykkbryter, Big 12mm
- 20 stk. Motstand 330 Ohm 1/6 W
- 20 stk. Motstand 10 kOhm 1/6 W

Mer informasjon komponentene og datablader finnes i avsnitt 3.2 på side 23 eller på følgende nettside: https://www.sparkfun.com/products/11227

Supplement til Inventors kit

- 10 stk. Grønn lysdiode

3.1.2 Abot

- 1 stk. Basisplate
- 2 stk. Servoer 360°
- 2 stk. Store hjul
- 4 stk. vinkeljern m/festeskruer m/mutter
- 1 stk. Kule for halehjul
- 1 stk. Holder for halehjul m/skruer
- 1 stk. Batteriholder 3AA m/plugger
- 1 stk. Avstandsmåler
- Div. Koblingsledninger M/M
- Div. Koblingsledninger H/H

Supplement til Abot kit

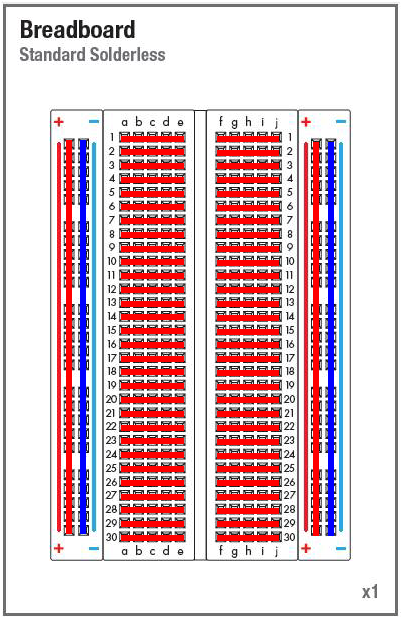
- 1. stk. Batteriholde 6 AA m/plugg

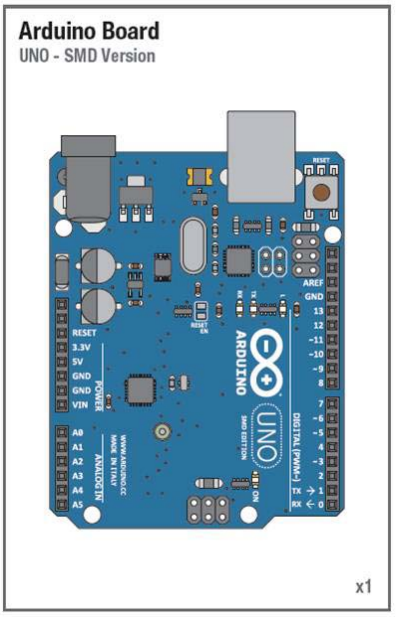
- 1 stk. Lite koblingsbrett
- 1 stk. Plastmontasjeplattform hjemmelaget
- 1 stk. Arduino UNO

3.2 Komponenter – beskrivelse

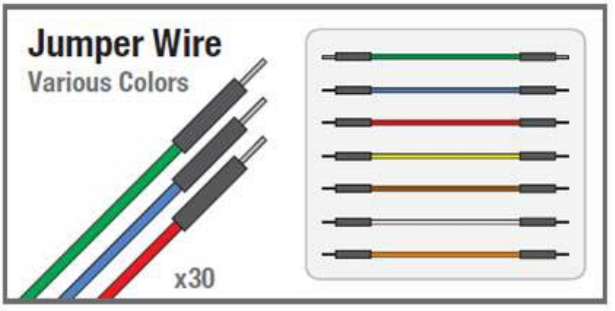
3.2.1 Koblingsbrettet

Deretter kan det være greit å gjennomgå komponentene som ligger i boksen sammen med elevene. La dem studere dem under veis. Normalt vil Arduino UNO og koblingsbrettet være montert på basisplata. Gjennomgå hvordan forbindelseslinjene i koblingsbrettet går.



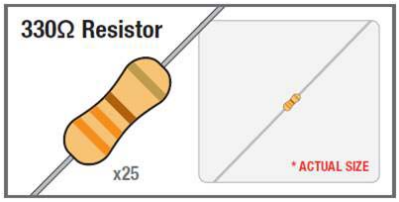


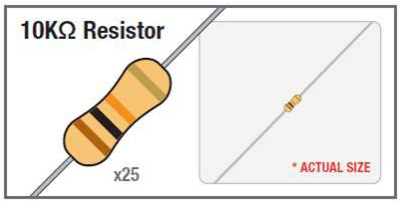
Strekene på koblingsbrettet viser hvilke koblingspunkter som er koblet sammen på undersiden. To komponentbein som er koblet til samme rad i to av hullene a – e eller f – j er koblet elektrisk sammen. Komponentene plasseres på koblingsbrettet og forbindes med Arduino-en ved hjelp av jumperer.

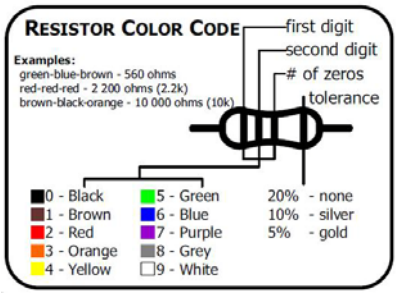


3.2.2 Motstander

Faste motstander







Beskrivelse:

Motstander kommer med mange ulike verdier. Her benyttes kun to: 330 Ohm og 10 kOhm. Fargekoden bestemmer verdien på motstandene. Det er viktig å velge riktig verdi.

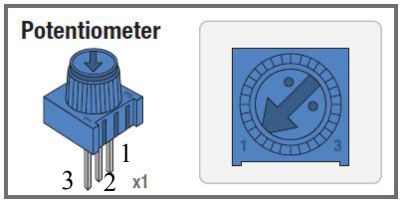
Bruksområder:

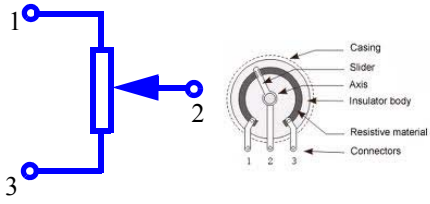
Motstander brukes til å begrense strømmen i en komponent, eller for å etablere et spenningsnivå slik som i spenningsdeleren. I følge Ohms lov vil spenningen over en motstand øke proporsjonalt med strømmen. Dette utnyttes når en ønsker å omdanne en varierende motstandsverdi (resistans) til en varierende spenning. Det har ingen betydning hvilken vei motstanden kobles.

Bestem verdien:

Fargene på ringene bestemme verdien. Hver farge står for et av sifrene 0 til 9. Hold gullringen til høyre og les av fargene fra venstre mot høyre. Første og andre ring angir første og andre siffer i verdien. Tredje ring angir antall nuller.

Potensiometer





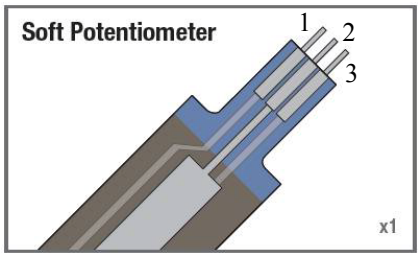
Beskrivelse:

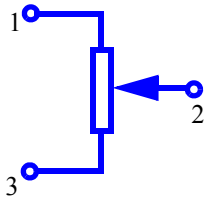
Et potensiometer er en motstand med et variabelt uttak. Ved hjelp av et lite ratt kan uttaket flyttes langs motstanden. På denne måten kan en på pinne 2 ta ut en brøk del av spenningen mellom pinne 1 og 3.

Bruksområder:

Potensiometeret kan etablere en variabel spenning mellom ytterpunktene 1 og 3. Eller fungere som volumkontroll for et signal som sendes inn mellom pinne 1 og 3.

Trykkpotensiometer





Beskrivelse:

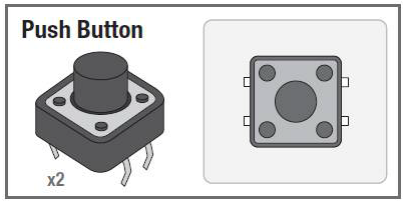
En trykkfølsom motstand fungerer på samme måten som et potensiometer. Istedet for å dreie på en knott, presser man på metallfilmen et sted mellom topp og bunn tilsvarende den spenningen man ønsker at potensiometeret skal gi ut.

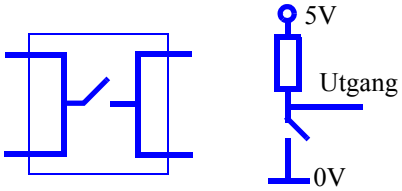
Bruksområder:

Har ikke sett slike potensiometer brukt andre plasser enn i dette settet. Lignende glidepotensiometer finner en f.eks. i miksebord.

3.2.3 Sensorer

Bryter





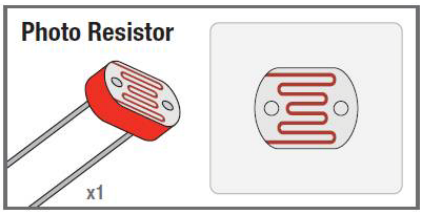
Beskrivelse:

Bryterne har fire bein. De er forbundet med hverandre to og to som vist på tegningen til venstre. Et trykk på knappen vil koble de to parene sammen. Forbindelsen opprettholdes så lenge knappen er trykket inn.

Bruksområder:

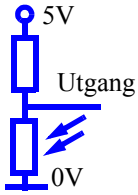
Brytere brukes til å gi en enkel på/av informasjon til kretsen, på samme måte som en lysbryter. For at kretsen skal forstå informasjonen må trykket omdannes til en spenning. Ved å koble bryteren mellom en motstand (10 kOhm) til 5 V og jord vil en på utgangen få en spenning som går fra 5 V til 0 V når bryteren trykkes inn.

Fotomotstand

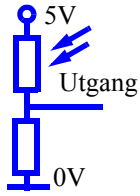


Beskrivelse:

En fotomotstand er en motstand som er avhengig av intensiteten på lyset som treffer den. Jo mer den belyses, jo lavere motstandsverdi (mørke 300 kOhm, sterkt lys 100 Ohm). Det betyr ingen ting hvilken vei den kobles inn i kretsen.



Høy spenning ut ved mørke



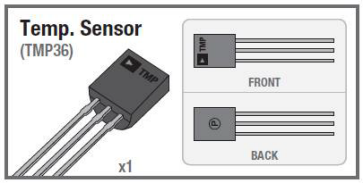
Høy spenning ut ved lys

Bruksområder:

Fotomotstander brukes der en ønsker å styre en funksjon ved hjelp av lysstyrken, som f.eks. tenning av lys når mørket faller på, telleapparater (en lysstråle brytes når noen går gjennom døra) o.l..

Kobles gjerne i en spenningsdeler. Plasseringen bestemmes av funksjonen. Se figuren til venstre.

Fotomotstand

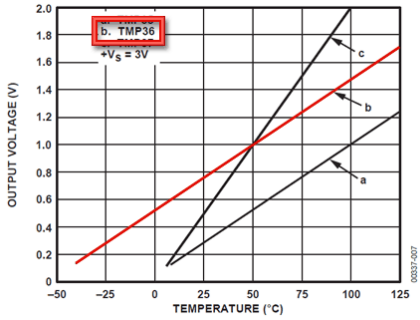


Beskrivelse:

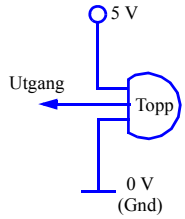
En temperatursensor (TMP36) av denne typen registrerer temperaturen og gir ut en spenning som er proporsjonal med temperaturen. OBS! Unngå å forbytte med transistorene!

Bruksområder:

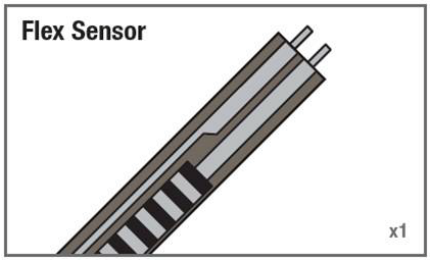
Temperatursensorer brukes i elektroniske termometre eller i termostater for å regulere en varmeovn. Den kan også brukes for å beskytte elektronikk, ved at strømmen brytes når temperaturen overskrider et maksimalt nivå. Hos TMP36 øker spenning med10 mV pr. grad C. Ved 0° C er spenningen 0,5 V.



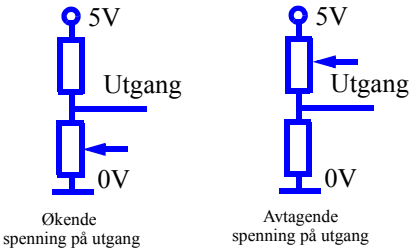
Spenning som funksjon av temperatur (TMP36 rød kurve)



Bøyesensor



Zebrastripenen på motsatt side av trykket



Beskrivelse:

Ved å bøye sensoren vil motstanden endre seg. Bøyes den med sebrastripene på innsiden av bøyen faller motstanden til ca. 25 kOhm. Bøyes den med sebrastripene på utsiden av bøyen øker motstandsverdien til 65 kOhm. Verdiene avhenger av graden av bøyning.

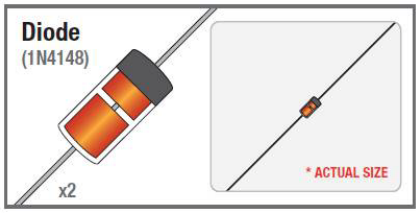
Bruksområder:

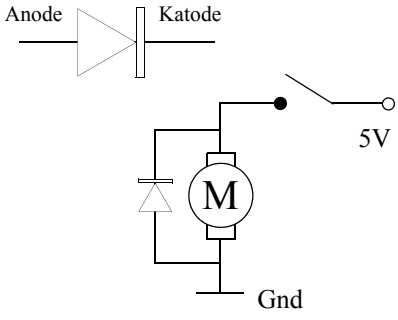
Lignende sensorer brukes i mange sammenhenger for å måle strekk eller sammentrykning i et materiale. Slik deformering kan f.eks. skyldes belastning eller nedbøyning. I denne sammenhengen går en slik sensor under betegnelsen *strekklapp*.

3.2.4 Halvledere

I denne sammenhengen er dette dioder og transistorer.

Diode





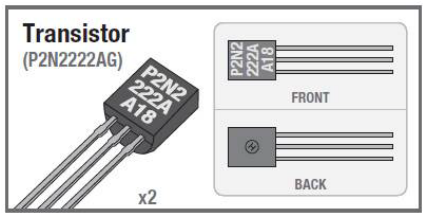
Beskrivelse:

Dioden er en halvleder som kun leder strøm en vei, fra anode til katode.

Bruksområder:

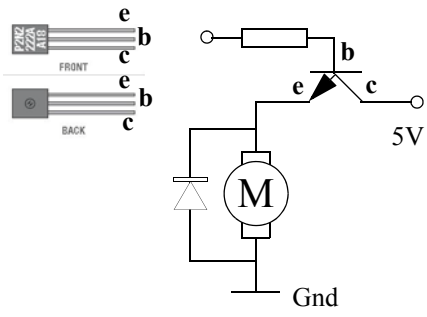
I denne sammenhengen skal vi benytte dioden som en "fly back" diode for å kortslutte strømmer som skyldes at strømmen i en spole i en motor eller et rele skal ødelegge transistoren driver. Den må derfor montere s slik at den normalt stenger for strømmen slik at den kan gå gjennom motoren. Når strømmen i motoren brytes, oppstår en motspennings som forsøker å hindre at motoren stopper. Denne motspenningen kan være så stor at den ødelegger transistoren. Dersom vi kobler en diode som vist på figuren til venstre, vil denne motspenningen i kortsluttet gjennom dioden slik at den ikke når transistoren og ødelegger den.

Transistor



Beskrivelse:

Transistoren har tre terminaler (bein), Fra collector (c) til emitter (e) kan det gå en relativt stor strøm. Størrelsen på collectorstrømmen kan styres av spenningen mellom basen (b) og emitter (e). Når den blir stor nok begynner det å gå en strøm i transistoren.



Bruksområder:

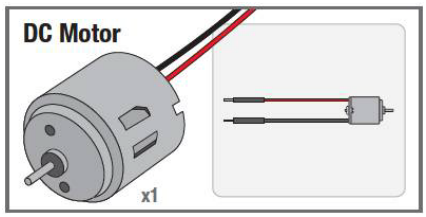
En transistor har gjerne to funksjoner. Som forsterker når basespenningen/strømmen varierer innen et lite område. Som bryter når basespenningen/strømmen er så stor at den slår transistoren på eller av.

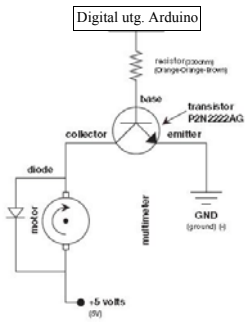
Transistorer som signal forsterker brukes i elektroniske forsterkere, radiosendere og mottakere, TV-er osv. Som brytere i datamaskiner og i styringssystemer.

3.2.5 Aktuatorer

En aktuator er en komponent som kan utføre en handling, enten det er å skape mekanisk bevegelse (motorer, pumper, magneter, releer), gi lyd (øretelefoner, høyttaler, sirene) eller lys (LED, lyspærer) eller varme opp en gjenstand eller et rom (glødetråd).

Motor





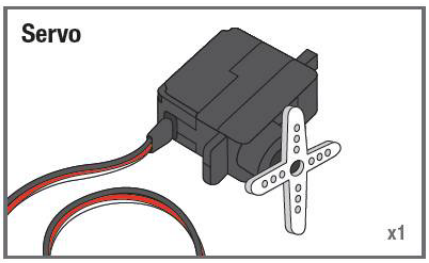
Beskrivelse:

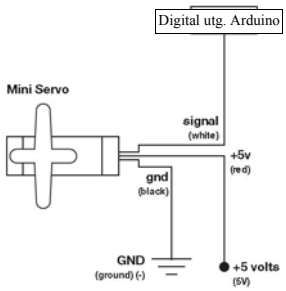
Akslingen begynner å rotere når det tilføres spenning over fra 3 V. Motoren som følger med kitet kan brukes på spenninger opp til 40 V og strømmer på opp mot 200 mA. Siden Arduinoen ikke klarer dete benytter vi en transistor bryter som tåler strømmen. Legg merke til "fly back" dioden over motoren. Denne skal hindre overspenning på transistoren idet motoren stopper.

Bruksområder:

I dag brukes elektriske DC-motorer til det meste der noe skal gå rundt eller bevege seg. Det være seg elektriske kjøkkenartikler, datadisker, leketøy som skal bevege seg, vaskemaskiner, pumper, vifter, vindusviskere og etter hvert elektriske biler.

Servo





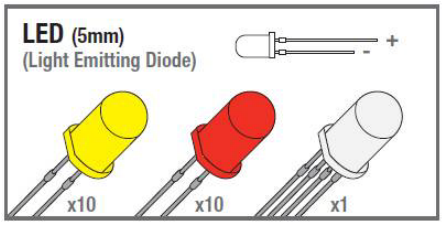
Beskrivelse:

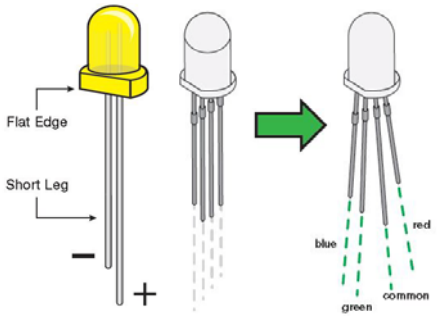
En servo er en slags motor som kan dreie akslingen en bestemt vinkel. Denne servoen kan på kommando fra mikrokontrolleren dreie akslingen en vinkel på fra 0 - 180°. Dreiningen skjer ved at servoen mottar pulser, hvor lengden av pulsen bestemmer dreievinkelen. En pulslengde på 1,5 ms gir en vinkel på 90°. Pulsene må gjentas med jevne mellomrom omtrent som man pulsbreddemodulerer et signal. Vi kobler derfor utgangen til en utgang som har denne funksjonen, f.eks. utg. 9.

Bruksområder:

Servoer av denne typen brukes ofte i modellfly for å styre side- og høyderor og flaps. De er også en viktig komponent i mange roboter hvor som skal bevege en arm e.l.

Lysdioder

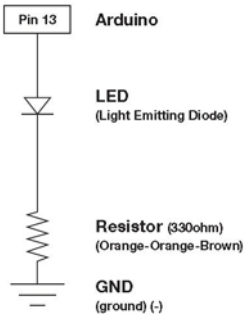




Beskrivelse:

Som navnet leder lysdiodene strøm bare den ene veien fra anoden til katoden. For at den skal lyse må den kobles i lederetning. Når strømmen i dioden overstiger ca. 1,5–2 mA, begynner den å lyse svakt. Lysstyrken øker etter som strømmen øker. For å å begrense strømmen, kobles den gjerne i serie med en motstand på 220–330 Ohm. Uten seriemotstand er det stor sannsynlighet for at dioden går i stykker.

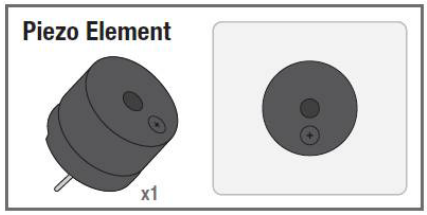
I settet er det vedlagt røde, grønne og gule lysdioder. I tillegg finnes en RGB-diode hvor en rød-, en grønn og en blå lysdioder montert i samme kapsel.

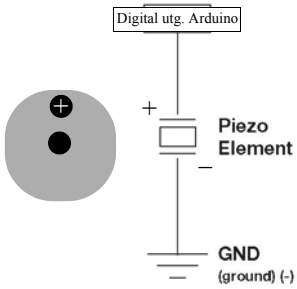


Bruksområder:

Lysdioder eller LED (Light Emitting Diode) brukes til signallamper, men mer og mer også til belysning.

Buzzer





Beskrivelse:

Buzzeren er et piezoelektrisk krystall som trekker seg sammen når det påføres en spenning og det høres et klikk. Når spenningen forsvinner, vil krystallet få tilbake sin opprinnelige form og det høres et nytt klikk. Ved å la spenningen variere fort kan man hør en lyd som i en høyttaler. For at den skal fungere rett, må + og – kobles rett.

Bruksområder:

Buzzeren brukes for å lage lyd og toner med forskjellig frekvens. Den kan også brukes i alarmer som f.eks. røykvarslere.

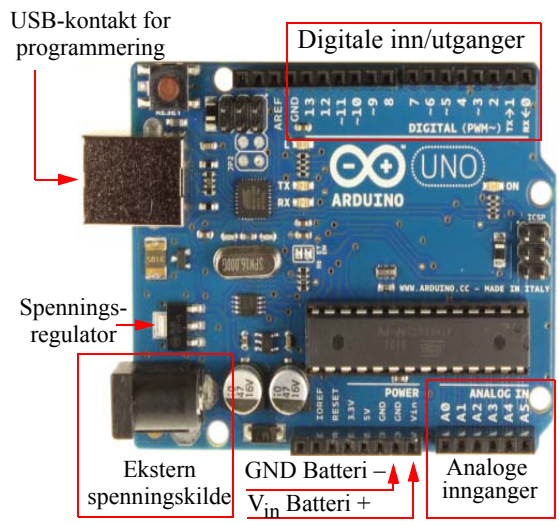
3.2.6 Arduino – mikrokontrollerkortet

Arduino UNO-kortet er databehandlingsenheten med sine inn- og utganger for sensorer og aktuatorer. Den har også en intern timer og lager for programmet og data.

I tillegg har kortet en USB-inngang for å legge inn programvare og en plugg for batteri eller batteriadapter.

Det Arduino-kortet som følger med settet er Arduino UNO R3, som er ett av de mest populære i mikrokontrollerkortene i Arduino-familien, og dermed leveres til en overkommelig pris (Robonors pris kr. 249,- inkl. MVA + frakt eller ELFA pris kr. 209,- inkl. MVA)

Kortet er bygget opp omkring Atmel mikrokontrolleren ATmega328P med en klokkefrekvens på 16 MHz og et flash lager på 32 kbyte, SRAM 2 kbyte og EEPROM 1 kbyte.

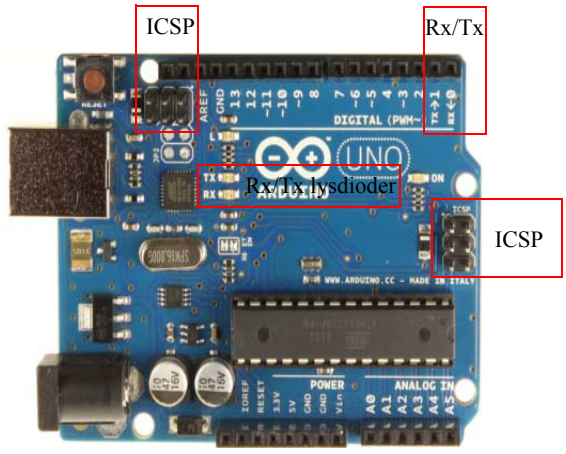


Kortet har dessuten følgende inn- og utganger:

- **Digitale I/O-porter**
  Kortet har 14 digitale inn/utporter (I/O-porter) som kan programmeres til enten å være en inn- eller en utgang. Seks av disse (3, 5, 6, 9, 10 og 11) kan *pulsbreddemoduleres* (pwm), disse er merket med ~ på kortet. Maksimal strøm på I/O portene er 40 mA.
- **Analoge innganger**
  Kortet har 6 analoge innganger
- **USB-kontakt** for direkte tilkobling av PC, for programmering av kortet. I tillegg kan data overføres mellom monitoren på PC-en og kortet. Under programmeringen tilføres kortet spenning fra USB-kontakten. Dersom denne belastes med mer enn 500 mA vil strømforsyningen bli brutt inntil strømtrekket reduseres under denne grensen.
- **Strømtilførsel**
  Tilkoblingspunkter for batterieliminator anbefalt spenning 7 – 12 V (grenseverdier 6 – 20 V). Batteri kan enten tilkobles eliminatorpluggen (2.1 mm + senter) eller via V_{in} (+) og GND (–). 5V utgangen lever spenning til f.eks. datainnsamlingskortet. Enkelte komponenter trenger lavere spenning som ev. kan leveres fra 3.3 V utgangen.
- **Reset**
  Kortet inneholder en RESET-knapp som resetter programmet som

Kantkontaktene er montert slik at tilleggskort kan monteres rett ned på Arduino-kortet.

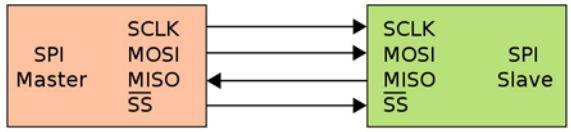
Kortet støtter ulik data kommunikasjon med omverdenen.

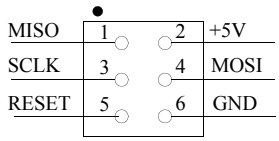


- **Rx/Tx**
  Kortet støtter UART seriell datakommuniaksjon via Rx og Tx portene (I/O-port 0 og 1). Det er også disse som benyttes for programmering av kortet. Disse er også tilkoblet to dioder som vil blinke når kortet kommuniserer med USB/PC. Tilsvarende vil skje når data overføres til programeditoren for monitorering av data på PC-skjermen.
- **I^2C-databus**
  I ^2C står for Inter IC-bus, og er ment å være akkurat det, da den ble utviklet av *Philips Semiconductor* tidlig på 80-tallet. Bussen er svært enkel med sine to linjer (klokke og

datalinje).Videre er hver krets langs bussen er adresserbar. Bussen er dessuten utstyrt med kollisjonsdeteksjon[1]. I starten var den definert med en hastighet på 100 kb/s. Senere, etter som en trengte raskere dataoverføring, er *Fast mode* - 400 kb/s og High speed - 3,4 Mb/s definert.

SPI-databus





Seriel Peripheral Interface buss (SPI-bus). Er en firelinjes databuss brukt for å overføre data til utstyr utenfor mikroprosessoren (periferutstyr). Databusen ble opprinnelig utviklet av Motorola. I en slik dataoverføring fungerer den ene modulen som master (sjefen), og den andre som slave (tjener). På figuren over er enheten til venstre master, og enheten til høyre slave. Master tar initiativet til dataoverføringen ved hjelp av SS-signallinjen, og sender sine data på linjen MOSI (Master Output Slave Input) linjen, mens slaven mottar på sin MOSI linje. Master mottar så svar på sin forespørsel til slaven som sender data tilbake på linjen MISO (Master Input Slave Output) som mottas av master på linjen MISO (Master Input Slave Output). SCLK er klokkesignalet som bestemmer takten til dataoverføringen. SS signalet (Slave select) varsler slaven om at nå ønsker master kontakt. Dersom det er flere slaver i systemet trengs flere SS-linjer, en til hver slave[2].

For den interesserte så kan kretsskjema for Arduino UNO hentes fra følgende nettadresse: http://arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf.

For mer informasjon om Arduino UNO R3 se: http://arduino.cc/en/Main/ArduinoBoardUno/

1. For mer informasjon se: http://www.i2c-bus.org/
2. http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus

4 Abot - oppbygging

Abot er en enkel robot konstruert av Atmel.

4.1 Mekanisk byggebeskrivelse

Denne byggebeskrivelsen er hentet fra "Abot chassis assambly guide" [3][3].

Nødvendig verktøy:

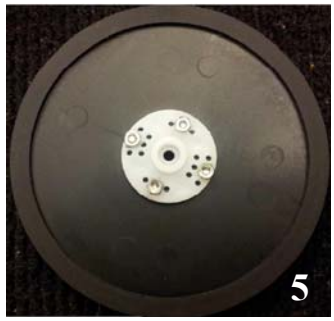
- Skrutrekkere, stjerne
- Boremaskin, med 2mm bor

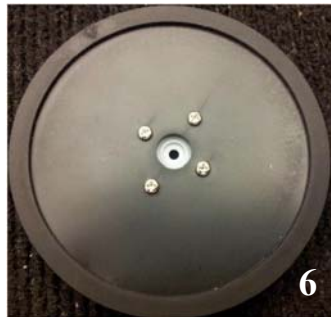
4.1.1 Montering av hjulene

Pakk ut servomotorene og demonter de hvite hjulholderne ved å skru opp senterskruen (1, 2, 3).

Hjulholderne har fire hull. Størrelsen til disse hullene økes til 2 mm ved hjelp av boret (4).

Legg merke til at hullene i hjulholderen ikke passer helt til hullene i det svarte hjulene. Dette kan løses ved at tre av hullene i hjulene tilpasses tre av hullene i hjulholderen. Deretter benyttes boret for å tilpasse det fjerde hullet (5, 6)[4].



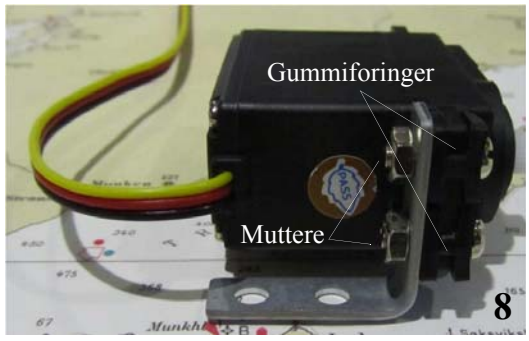


3. Bildene er gjengitt med tillatelse av Torgeir Bjørnvold, Atmel Norge AS
4. I nyere utgaver av Mr. Abot er det kun nødvendig å justere hullstørrelsen til skruene.

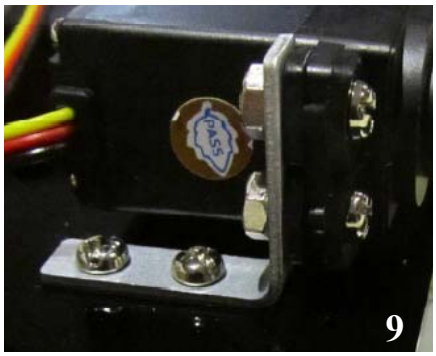
Bruk 4 x 2 mm skruer med tilhørende mutere.

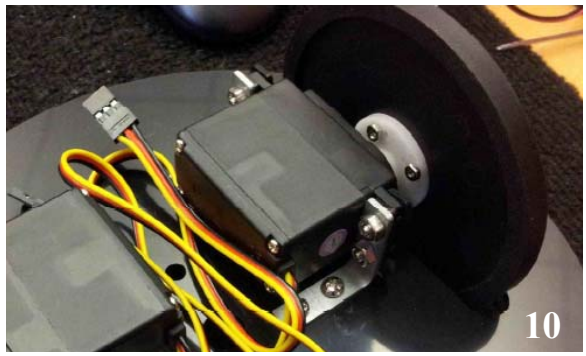
4.1.2 Monter braketter på motorene

Brakettene (vinkeljern) monteres, en på hver side av motorene, ved hjelp av to skruer med tilhørende muttere og gummiforinger (7, 8).



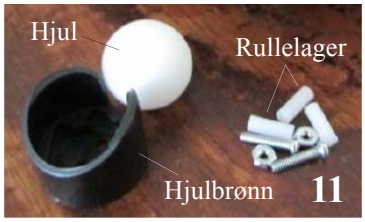
Dernest festes de to motorene til den svarte karosseriplaten ("chassis"), en på hver side, slik at hjulene går gjennom de to spaltene i plata. Brakettene fastes med to skruer med tilhørende muttere. Pass på at hjulene har god klaring til sidene av spaltene før skruene strammes.

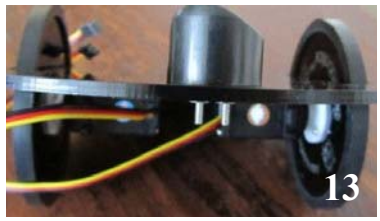




4.1.3 Montering av halehjulet

Halehjulet består av en kule som går på tre rullelager som ligger i bunnen av hjulbrønnen.





Hjulbrønnen festes i bakkant av karosseriplaten ved hjelp av to 2 mm skruer med mutter (11, 12, 13). Legg merke til at skruehodene skal være på innsiden av hjulbrønnen.

Dernest legges rullelagrene inn i fordypningene i bunnen av hjulbrønnen (14), før kulehjulet presses på plass i brønnen (15).

4.2 Elektrisk oppkobling

Dette kan gjøres på flere ulike måter. Her har vi vist to varianter:

- Oppkobling med shield-kort og 6 V batteripakke
- Oppkobling med Arduino og 9V batteripakke

4.2.1 Oppkobling med shield-kort og 6 V batteripakke

Denne oppkoblingen er utviklet av ***Adam Gajda*** høsten 2013. Hovedhensikten var å lage en robot som kunne følge en svart linje. Forenklet kan oppkoblingen også brukes til enklere innledende øvelser uten lyssensorer eller ved oppkoblinger på koblingsbrettet som sitter på roboten.

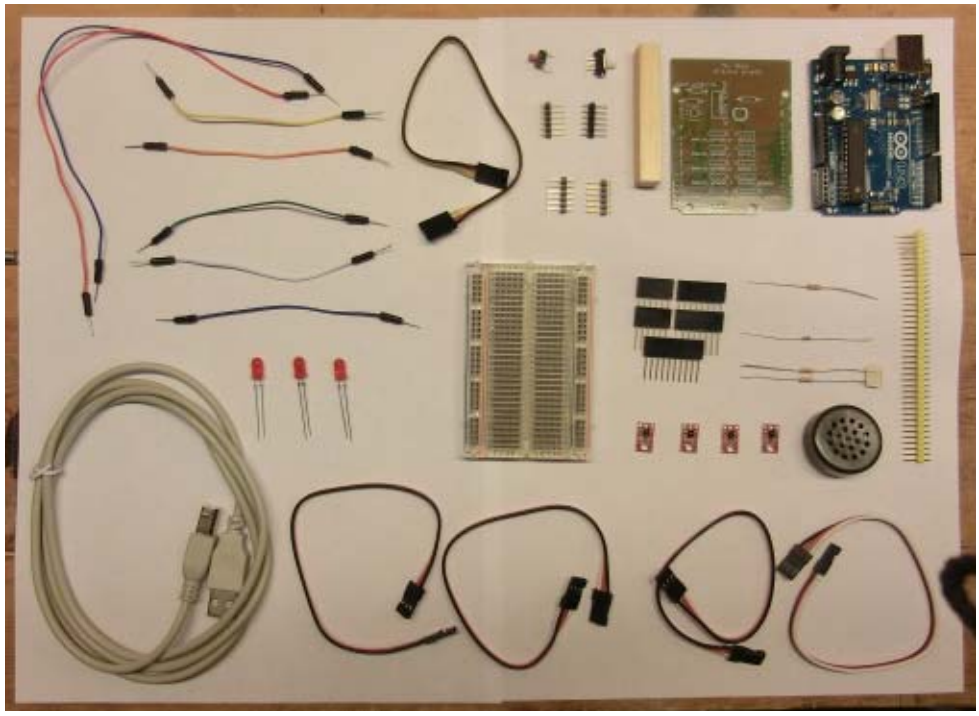
Oversikt

Lista under gir en oversikt over komponenter i tillegg til det som følger med Mr. Abot.

Komponenttype	Leverantør	Ant.	Pris
Mr. Abot	http://store.atmel.com/PartDetail.aspx?q=p:10500327#tc:description	1	99 ($)
QTR-1A Reflectance Sensor	http://www.pololu.com/catalog/product/958	4	1,76 ($)
3 PIN DUAL-FEMALE JUMPER WIRE - 20CM	http://imall.iteadstudio.com/prototyping/cable-and-wires/im120530011.html	4	0,81 ($)
4 PIN DUAL-FEMALE JUMPER WIRE - 20cm (PARALLEL)	http://imall.iteadstudio.com/prototyping/cable-and-wires/im120530012.html	1	1,08 ($)
Arduino UNO	ELFA 10-389-9	1	167,00 (NOK)
Breadboard (8.3 x 5.5cm)	http://imall.iteadstudio.com/prototyping/breadboard/im120530015.html	1	3,80 ($)
Jumper wire (75 stk pr. pakke)	http://imall.iteadstudio.com/prototyping/cable-and-wires/im120530005.html	7	0,5 ($)
USB to mini	http://imall.iteadstudio.com/prototyping/cable-and-wires/im120530007.html	1	0,8 ($)
LED	http://no.farnell.com/vishay/tlhr5400/led-5mm-he-red/dp/1045473	2	0,67 (NOK)

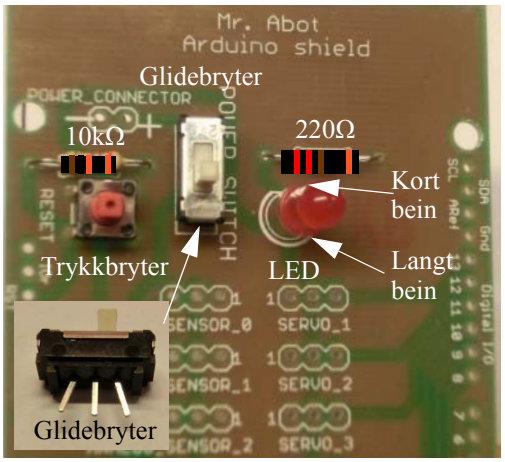
Komponenttype	Leverantør	Ant.	Pris
220 ohm resistor	http://no.farnell.com/te-connectivity/cfr16j220r/resistor-carbon-220r-0-25w-5/dp/2329500	2	0,169 (NOK)
Høyttaler (100 Ohm)	http://www.pololu.com/product/1261	1	1.79 ($)
Boks	Clas Ohlson, Smart Store Classic 12, 28 x 28 x 17 (34-1552-12)	1	29,00 (NOK)
Skrin	Clas Ohlson, Skrin (40-7523)	1	26,00 (NOK)
Arduino Abot Shield (PCB order)	http://imall.iteadstudio.com/open-pcb/pcb-prototyping/im120418003.html	1	2,5 ($)
Stackable female headers	http://www.pololu.com/product/1035	1	1.95 ($)
Slide switch	http://no.farnell.com/alps/stsss9121/slide-switch-sp-2-pos-vert/dp/1123875	1	6,56 (NOK)
LED	http://no.farnell.com/vishay/tlhr5400/led-5mm-he-red/dp/1045473	1	0,67 (NOK)
220 ohm resistor	http://no.farnell.com/te-connectivity/cfr16j220r/resistor-carbon-220r-0-25w-5/dp/2329500	1	0,169 (NOK)
10k ohm resistor	http://no.farnell.com/te-connectivity/cfr16j10k/resistor-carbon-10k-0-25w-5/dp/2329474	1	0,143 (NOK)
Tactile switch	http://no.farnell.com/alps/skhhdga010/switch-tactile-6x6mm-vert-red/dp/2056821	1	1,07 (NOK)
Gold pins (40 pin)	http://no.farnell.com/fci/77311-818-36lf/header-pin-2-54mm-36way-1row/dp/1097955	1	20,95 (NOK)

Figuren under viser komponentene. I tillegg kommer Mr. Abot med avstandssensor og batteripakke.



Montering av "shield"-kort

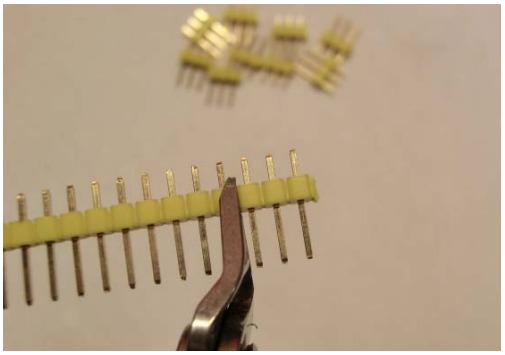
Shield-kortet monteres på "ryggen" av Arduino UNO kortet og tilrettelegger kontaktene slik at de lett lar seg tilkoble med standard flatkabler til sensorer og servomotorer. I tillegg inneholder kortet en glidebryter for å slå på strømmen, en trykk-bryter for å resette kortet og en rød lys-diode som indikerer at strømmen er slått på.



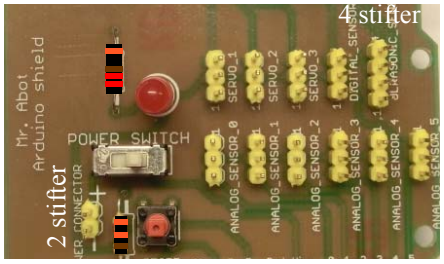
1. *Monter følgende på "shield"-kortet:*

- Motstand 220 Ω (rød, rød, brun, gull)
- Motstand 10 kΩ (brun, sort, oransje, gull)
- Glidebryter (hvit). Beina må sprike for å tilpasses hullene (se innfelt figur til høyre).
- Trykkbryter (rød)
- Lysdiode (merk plassering av langt og kort bein).

2. Del opp stiftlisten ("gold pins") i:
   - 10 grupper med 3 stifter (se fig. til høyre)
   - 1 gruppe med 4 stifter
   - 1 gruppe med 2 stifter

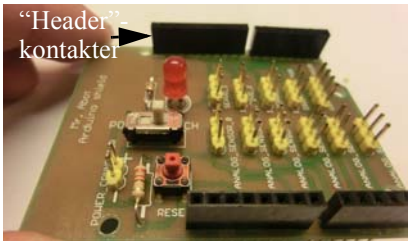


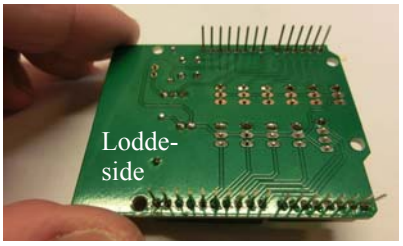
3. Plasser stiftene i hullene på midten av plata som vist på figuren under. Kontaktene loddes på loddesiden (undersiden).



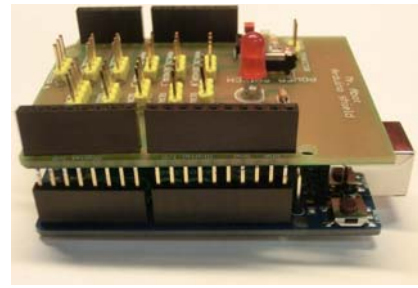
4. Monter fire "header"-kontakter for å montere "shield"-kortet på "ryggen" av Arduino UNO kortet. Velg de fire av de fem som passer best til kontaktene på Arduino UNO kortet. Disse har lange bein og skal stikkes gjennom "shield"-kortet og ned i "header"-kontaktene på Arduino kortet. Her er det viktig at beina treffer kontaktene.

Det er derfor særdeles viktig å lodde dem loddrett på kortet



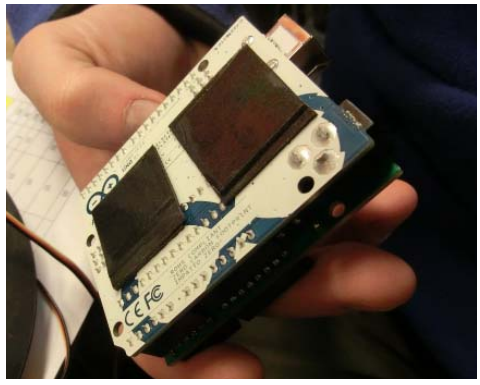


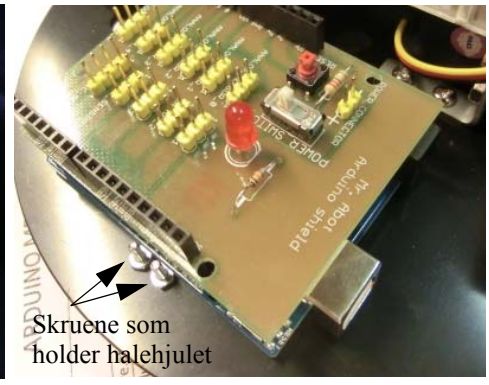
Figuren til høyre viser hvordan stiftene fra "header"-kontaktene passer til kontaktene på Arduino UNO kortet. Pass på at stiftene presses helt ned i kontaktene, og *ikke* slik som vist på figuren til høyre.



Montering av Arduino UNO kortet m/"shield"-kort.

5. Arduino UNO festes med dobbelsidig tape bak på Mr. Abot, men foran skruene som holder halehjulet som vist til høyre på figuren under. Det kan være nødvendig å legge på dobbelt med tape slik at pinnene som stikker ut under Arduino-kortet kommer fri fra den svarte monteringsplata til Mr. Abot.

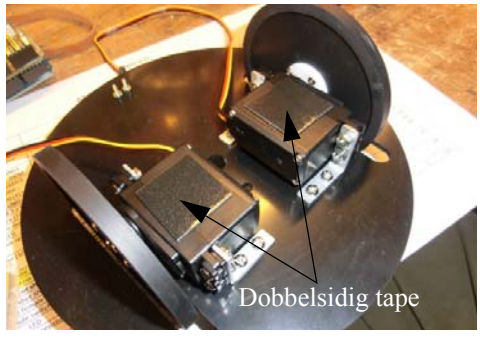


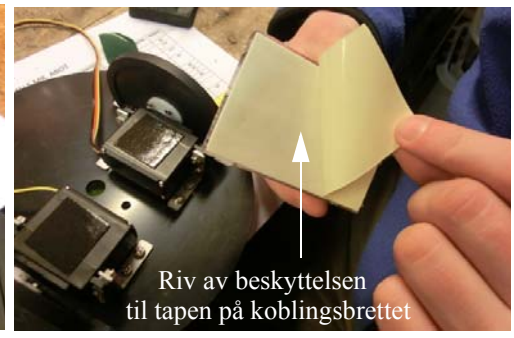


Montering av koblingbrettet

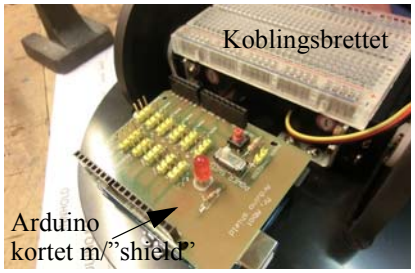
Vi har valgt å inkludere et koblingbrett på roboten. Dette gir både mulighet for å lage enkle oppkoblinger som introduksjonsøvelser før man starter med å arbeide med Mr. Abot, eller man kan inkludere tilleggskretser til roboten. Dette kan f.eks. inkludere lys, lyd eller andre tilleggsfunksjoner.

6. *Monter koblingsbrettet.*
   Dette kan gjøres ved å benytte den dobbelsidige tapen som er på koblingsbrettet. Vi valgte i tillegg å legge to tapeputer på hver av servomotorene for å heve koblingsbrettet opp fra brakettene som holder motorene, som vist på figuren under.





7. Monter koblingsbrettet som vist på figuren til høyre.

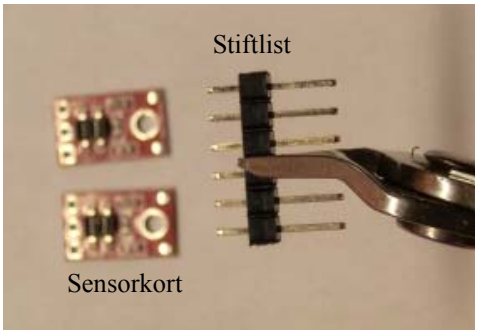


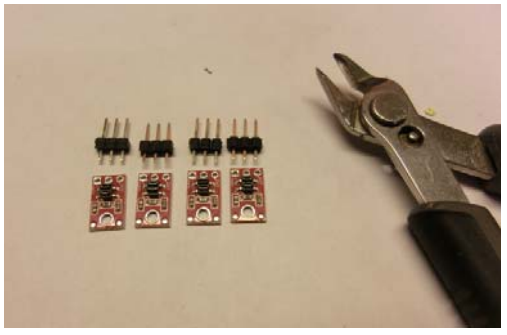
Montering av sensorer

Det skal monteres fire reflektans-sensorer, som måler lyshetsgraden til underlaget, dvs. hvor hvitt eller svart underlaget er. Disse sensorene kan brukes til å følge en svart/hvit kant (svart stripe mot lyst underlag).

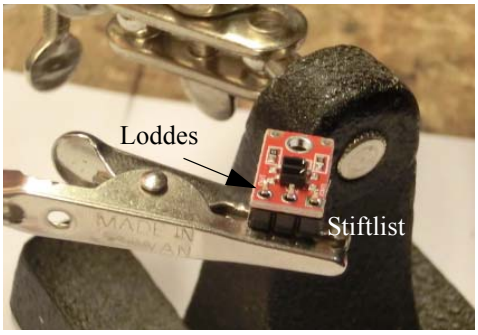
Det skal i tillegg monteres en avstandssensor som "ser" framover og som kan brukes til å detektere hindringer foran Mr. Abot.

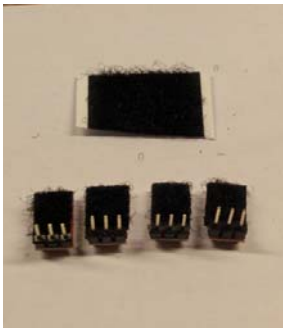
8. *Montering av fire reflektans-sensorer.*
   Disse kommer i poser på to og to. Det må monteres stiftlister med tre stifter på hvert sensorkort. Det følger med både bøyde og rette stiftlister. Her velger vi å benytte de rette, som deles i to siftlister, hver med tre stifter, som vist på figuren under.





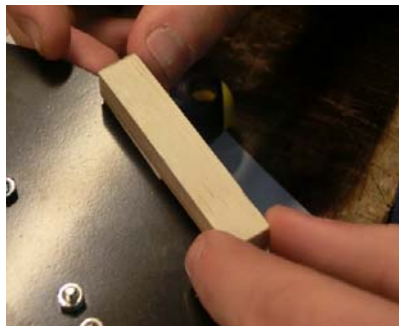
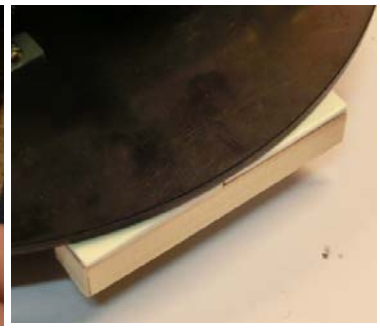
Stiftlistene monteres på sensorkortene og loddes på samme side som sensoren





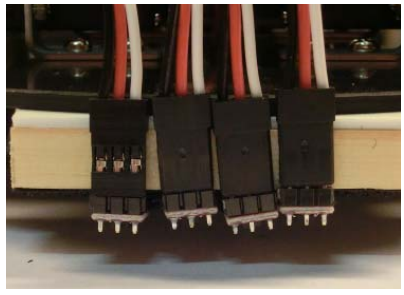
9. *Montering av borrelås på reflektans-sensorene*
   Disse sensorene skal kunne festes foran på Mr. Abot slik at de peker ned mot underlaget. Vi velger å lime "løkke"-delen av borrelåsen på sensorkortet, som vist til høyre på figuren over. Her bør en alternativt vurdere å benytte dobbelsidig tape, da borrelåsen kan bli et for svakt feste. Dessuten kan flatkabelen til sensorene festes med dobbelsidig tape foran på bjelken (se under) slik at sensoren blir stødigere.

10. *Montering av bjelke for å holde reflektanssensorene*
Vi har valgt å bruke en firkantlist (10 x 12 x 60 mm) som festes under og helt foran på Mr. Abot, slik at midtpunktet tangerer kanten av den svarte monteringsplata til roboten.

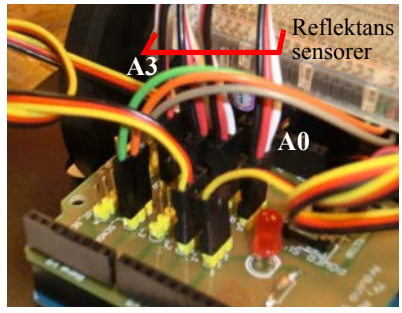
På undersiden av bjelken festes en stripe av borrelås, den delen med kroker, slik at reflektans-sensorene kan festes til denne (som omtalt foran).

11. *Montering av reflektans-sensorene*
Reflektanssensorene monteres med borrelås på undersiden av bjelken. Legg merke til hvilken vei ledningene skal monteres. Her er det lett å ta feil da det ikke alltid er slik at ledningene er montert samme vei ned i kontaktene. Sørg for at ledningene legges under koblingsbrettet. Dette gjør at de ikke skygger for avstandssensoren som skal monteres på koblingsbrettet.

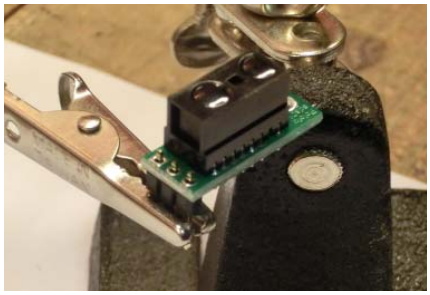

Ledningene må også monteres rett på "shield"-kortet. Her er det slik at alle ledninger fra sensorer og motorer, skal kobles med den svarte (–) ledningen inn mot midten av kortet, som vist på figuren til høyre. Reflektans-sensorene skal monteres i posisjonene:
- Analog sensor_0 (A0)
- Analog sensor_1 (A1)
- Analog sensor_2 (A2)
- Analog sensor_3 (A3)

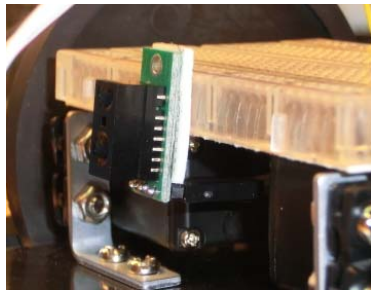
fra høyre mot vestre sett bakfra.



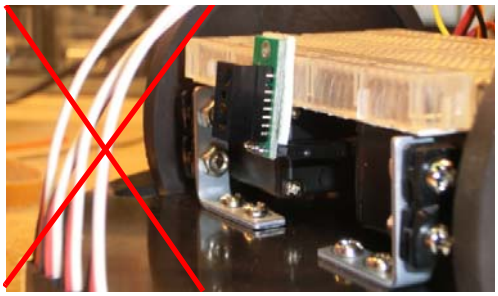
12. *Montering av digital avstandssensor*
Denne sensoren skal peke framover og kan monteres relativt høyt på roboten. Sensoren følger med Mr. Abot, og vi velger å montere den ved hjelp av dobbelsidig tape.

Vi monterer en rett, trepins stiftlist på sensor-kortet. Også her velger vi å lodde stiftlisten på samme side som sensoren er montert. Vi bruker tre av jumperledningen levert sammen med Mr. Abot for å forbinde avstandssensoren til "shield"-kortet. Grå (–), oransje (+) og grønn (signalering), som vist på det midterste bildet under.





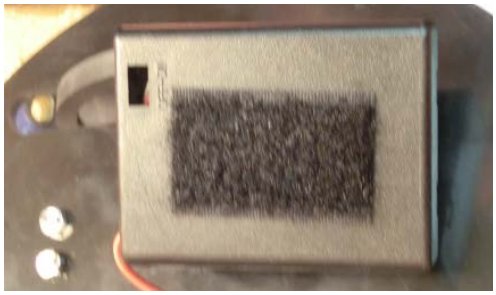
Sensoren kan festes til kanten av koblingsbrettet. Det er, som nevnt, fristende å la kablene fra reflektans-sensorene passere over koblingsbrettet. Dette vil imidlertid skygge for avstands-sensoren som vil ha problemer med "se" veggen for bare ledninger som vist på figuren under.



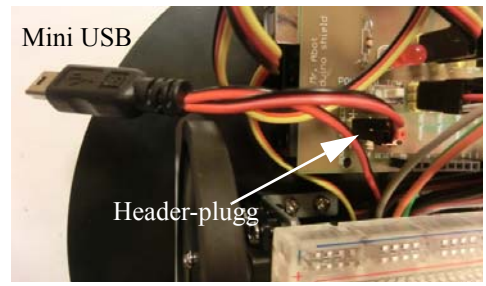

Montering av batteriholderen

13. Batteriholderen er vedlagt Mr. Abot og leverer 6 V med fire standard AAA batterier. En ulempe med denne batterikassetten er at den kun leverer 6 V. Dersom vi skulle ønske å benytte ladbare batterier, vil disse levere en lavere spenning som kan bli temmelig marginal mht. til drift av roboten.

    Batteripakken monteres på undersiden foran halehjulet som vist på figuren under. Vi har valgt å benytte borrelås som følger med Mr. Abot slik at batterikassetten lett kan byttes ut.




Batterieledningene, som har to kontakter, en "header"-kontakt for tilkobling til stiftlist og en mini USB-plugg, stikkes gjennom hullet midt på monteringsplata til roboten. Vi velger å benytte "header"-kontakten og kobler den til stiftene på shield"-kortet merket *Power connector*



En bedre batteripakke er den som er vist på figuren til høyre. Pluggen passer til batterieliminatorkontaken på Arduino UNO kortet. Pakken kan kjøpes fra: http://www.ebay.com/itm/6XAA-6xAA-6-AA-9V-Battery-Holder-Box-Case-Wire-5-5-2-1mm-Plug-good-/370992745346 for 1$. Batteriholderen kan monteres på samme sted på undersiden som den lille kassetten som følger med Mr. Abot.

Montering av høyttaler

Arduino UNO gir rike muligheter til å skape lyd. Det er derfor vedlagt en 100Ω høyttaler. Denne burde kunne kobles rett til en digital utgang, ev. i serie med en motstand.

Det kan imidlertid være noe utfordrende å montere høyttaleren på koblingsbrettet slik at det blir mulig å koble seg til med jumpere. Det finnes imidlertid en måte å gjøre det på.

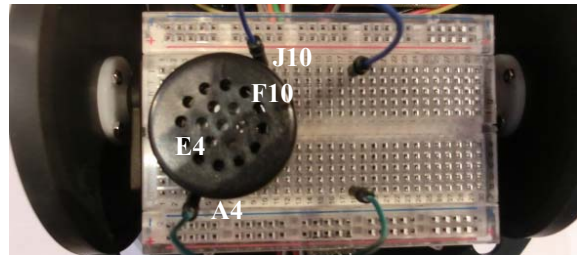
Høyttalerens pinner stikkes ned i: E4 og F10. Vi kan nå disse tilkoblingspunktene ved å kobble jumperne til A4 og J10.

FE23100PCW-03 LF

1. Characteristics

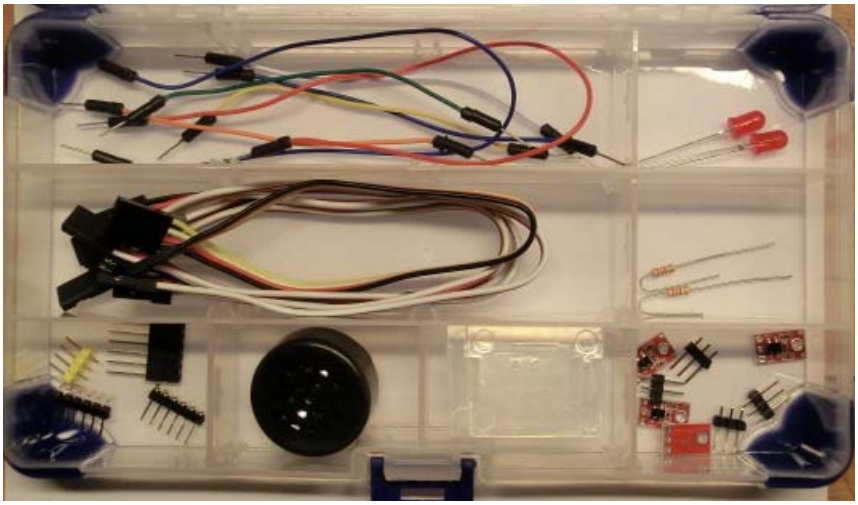
1,1 Electrical and Mechanical Characteristics

No.	Item	Specifications
1-1)	Impedance (1V,*2kHz)	100 ±10 % Ω
1-2)	Rated Input power	0.15W
1-3)	Max Imput	0.3W
1-4)	Frequency response	1000,.,4000Hz
1-5)	Sound pressure level	≥90dB(at 0.25W/10cm/1000Hz square wave)
		≥95dB(at0.25W/10cm/2000Hz square wave)
1-6)	Buzzes and Rattles	Must be normal
		sine wave 3.87V
1-7)	Operating temperature	-40,.,+85℃
1-8)	Weight	9g

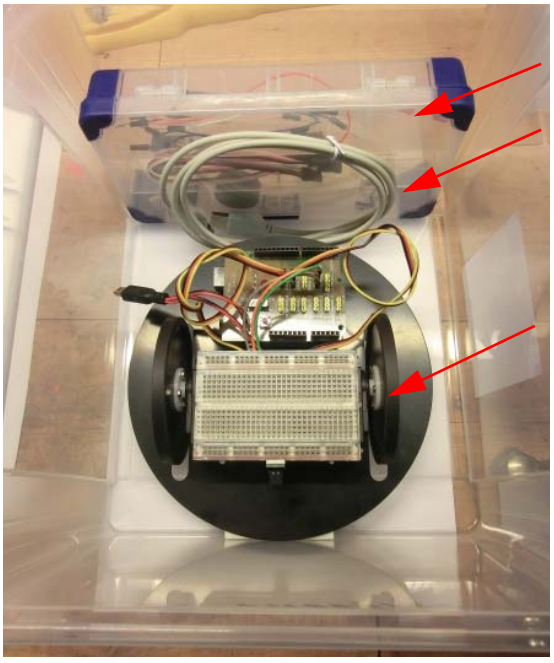


Pakking

14. De resterende delene samt reflektans-sensorene med tilhørende ledninger pakkes i det lille skrinet. Skilleveggene kan fjernes slik at størrelsen på rommene kan tilpasses, som vist på figuren under (merk at bildet er tatt før reflektans-sensorene har fått påmontert stiftlisten)



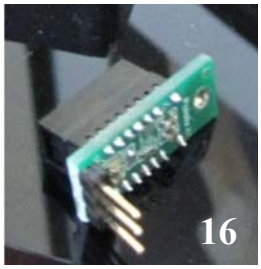
Roboten, kabelen og skrinet legges i plastboksen som vist under.

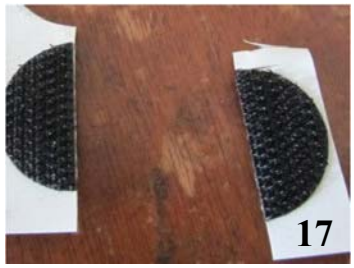


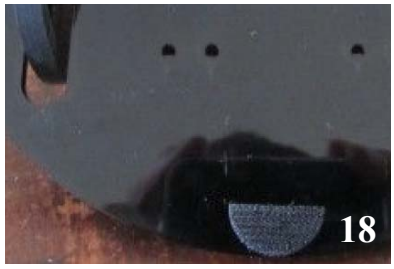
4.2.2 Oppkobling med Arduino og 9V batteripakke

Montering av avstandsensor

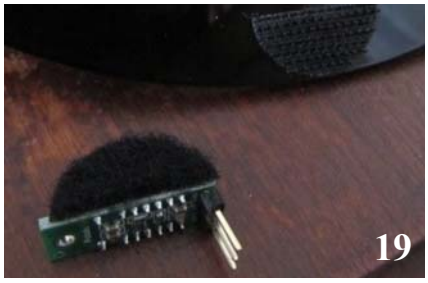
En avstandsensor (16) er en innretning som gir ut et signal når roboten og sensoren møter en hindring. Denne bør derfor monteres i forkant av roboten (18).

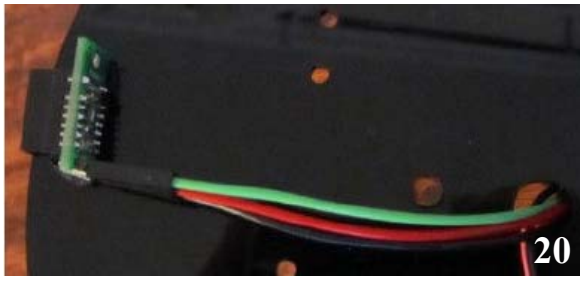

16


17


18

En borrelås benyttes for å feste sensoren foran på karosseriplaten (18, 19).


19


20

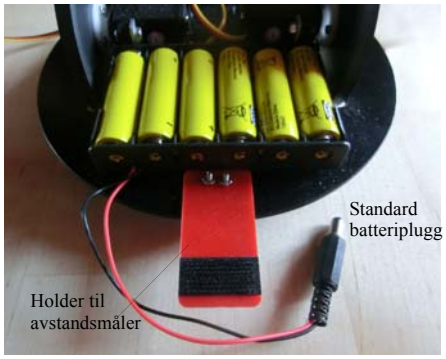
Ledningen fra sensoren føres opp på oversiden av karosseriplaten gjennom det store hullet på midten av plata.

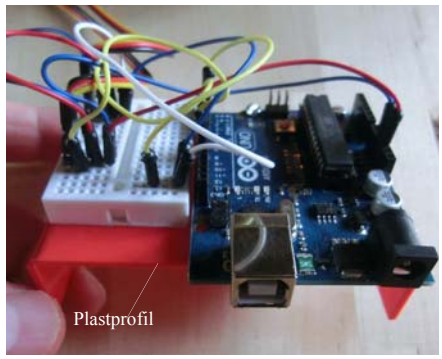
Montering av batteriholder og Arduino

Den medfølgende batteriholderen har plass til tre AAA batterier, dvs. 4,5 V. Dette er akkurat tilstrekkelig til å drive en Arduino, dog ikke via batteripluggen, men må kobles direkte til +5 V kontakten på kortet (–0 V til Gnd).

Et alternativ er å montere et 9 V batteri enten som 6 AAA batterier eller et standard 9 V batteri (6LF22).

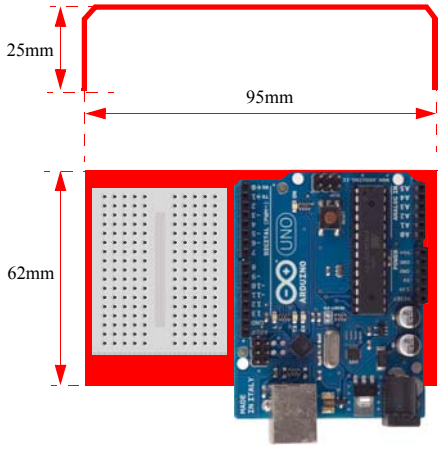
Vi har valgt å montere 6 AA batterier i en holder som vist på figuren under. Batteripakken kobles til Arduino ved hjelp av en standard batteriplugg.





Et rektangel av plast monteres i front ved hjelp av to maskinskruer med mutter. En borrelås er festet foran på plastrektangelet slik at en avstandssensoren kan festes til borelåsen.

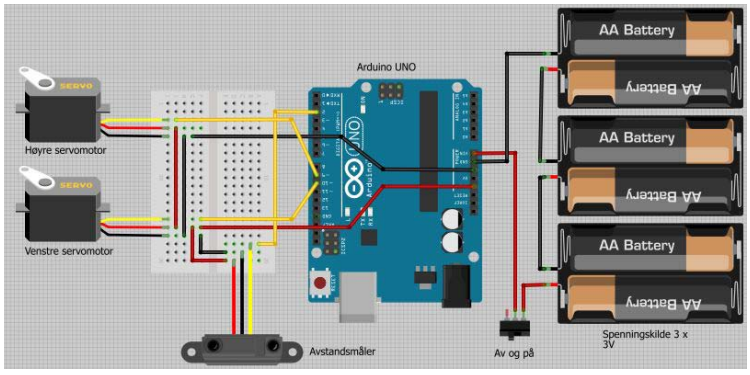
En U-formet plastprofil bøyes til ved hjelp av en plastknekker. Denne er ment for montasje av et lite koblingsbrett (mini) og en Arduino UNO, figuren til høyre.

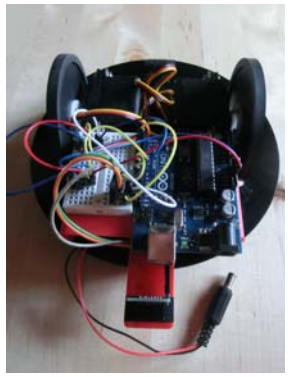


Oppkoblingen

Koblingsbrettet benyttes primært som mellomstasjon for ledningene fra servomotorene og Arduino-en. Koblingsskjema er vist på figuren under.

Oppkoblingen er vist til venstre på figuren under. Til høyre ser vi den ferdige oppkoblingen. Av og på bryteren som er vist i oppkoblingen er ikke montert på Abot-en vist på bildet til høyre.



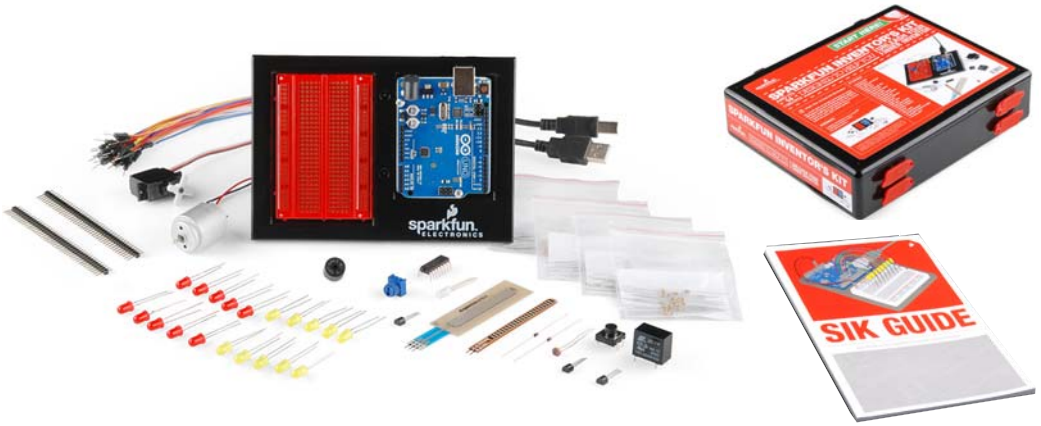


5 Programmer og programmering

I dette kapittelet skal vi først installere programeditoren og vise hvordan den brukes. Dernest skal vi ganske kort foreslå en måte å strukturere CanSat-programmet på, for deretter å gi en oversikt over noen viktige og anvendelige kommandoer.

5.1 "Sparkfun's Inventors kit"

Sparkfun's Inventors kit er i tillegg til en samling elektroniske komponenter (se avsnitt 3.1 på side 21), en rekke eksempelprogrammer som egner seg som en første introduksjon til programmering i C. Eksempelprogrammene kan lastes ned fra https://www.sparkfun.com/products/11227 (SIK examples code).



Med settet følger et rikt illustrert hefte Sparkfun Instruction Guide (SIK) som beskriver oppbyggingen av kretsene og introduserer de ulike eksempelprogrammene. Vi har i vårt introduksjonsopplegg valgt å bygge på opplegget i dette heftet. I alt inneholder heftet beskrivelse og programvare til 14 ulike byggeprosjekter, fra de helt enkle til de mer omfattende:

1. **"Blinkende LED"**
   *Oppgaven egner seg godt som første introduksjon og kan i stor grad differensieres. Her læres å definere enkle variabler, en port som in- eller output (**pinMode()**) og skriving til en digital port (**digitalWrite()**)*

2. ***"Potensiometer"*** *- Styr blinkhastigheten med et potensiometer*
   Oppgaven egner seg til å lære ***analog avlesning*** *(****analogRead()****) av en spenning samt variering av* ***forsinkelse*** *(****delay()****).*

3. ***"RGB LED"*** *- Styr fargene i en Rød-Grønn-Blå LED (RGB LED)*
   Oppgaven egner seg for å lære fargeblanding og ***pulsbreddemodulasjon*** *for styring av intensiteten til en lysdiode (****analog.Write()****). Dessuten omtales kontrollfunksjonene* ***for loop*** *og* ***if setning****.*

4. ***"Mange LED"*** *- Styr en rekke av 8 LED*
   Oppgaven egner seg for a lære bruk av ***for loop*** *og å sette opp* ***array[].***

5. ***"Trykk knapp"*** *- Bruk en trykknapp til å slå en LED av og på*
   *Oppgaven egner seg for oppkobling av knapper med bruk av motstand, samt å avlese en digital inngang (****digitalRead()****), og bruk av* ***if setninger.***

6. ***"Foto-motstand"*** *- Styr lyset i en LED ved hjelp av lyset i rommet*
   *Oppgaven introduserer spenningsdeleren, dvs. hvordan en varierende motstand kan omdannes til en varierende spenning. Dessuten anvendes funksjoner for å konvertere ett tallområde til et annet (****map()****) og en funksjon for avgrense et tallområde (****constrain()****).*

7. ***"Temperatursensor"*** *- Mål temperaturen og skriv ut verdien ut på PC-skjermen.*
   *Oppgaven egner seg for å lære hvordan lese av en analog spenning (****analogRead()****), hvordan sette opp en seriell kommunikasjonslinje tilbake til PC-en (****Serial.begin()****) og skrive resultatet til PC-skjermen (****Serial.print()*** *og* ***Serial.println()****).*

8. ***"En enkel servo"*** *- Styr vinkeldreiningen til en servo.*
   *Oppgaven egner seg for å lære hvordan inkludere biblioteker av funksjoner (****#include()****), definere en serverobjekt (****Servo*** *<servonavn>), knytte servoen til en spesiell utgang (<servonavn>****.attach()****) og styre servoen (<servonavn>****.write()****).*

9. ***"Bøyesensor"*** *- Bruk en bøyesensor (strekklapp) til å måle bøyningen og syre en servo.*
   *Oppgaven egner seg til å lese av en analog verdi (****analog.Read()****), avgrense tallområdet (****map()****) og styre en servo (<servonavn>****.write()****).*

10. ***"Trykkfølsomt potensiometer"*** *- Bruk et trykkfølsomt potensiometer til å styre RGB LED.*
    *Oppgaven egner seg til å lære å lese av en analog inngang (****analog.Read()****), konvertere et tallområde (****constrain()****) og styre nivået til lysdioder ved hjelp av pulsbreddemodulasjon (PWM)* ***(analog.Write())****.*

11. ***"Buzzer"*** *- Lag lyd og toner med en piezoelektrisk lydkilde*
    *Oppgaven egner seg godt for å lære å programmere lyd (****tone()****), dessuten læres å definere array som holder note- og toneverdier (<arraynavn>****[]****).*

12. ***"Roterende motor"*** *- Styr rotasjonshastigheten til en motor*
    *Oppgaven egner seg for å lære å koble opp og styre en motor ved help av kommandoer fra PC-konsollet. Dette krever at Arduino-en leser fra USB-inngangen (****Serial.available()*** *og* ***Serial.parseInt()****). Dessuten omtales bruk av* ***while setningen****.*

13. ***"Rele"*** *- Lære å bruke et rele for å styre større strømmer og spenninger*
    Oppgaven egner seg for å lære å koble opp og bruke et rele slik at det er mulig å styre utstyr

som krever større strømmer (motorer, pumper, varmelementer o.l.)

14. ***"Skift register"**. Lære å styre et skiftregister og en rekke av lysdioder.*
    *Oppgaven egner seg godt til å lære hvordan en skal legge inn data i en integrert krets bruk av SPI-portene (SerialPeripheral Interface) og klokke- og datainngang (**shiftOut()**) og overføring av array av bit (**bitWrite()**).*

Før vi går i gang med disse skal vi installere programeditoren.

5.2 Installasjon av Arduino programeditor

5.2.1 Arduino programeditor

Nedlasting av programvare

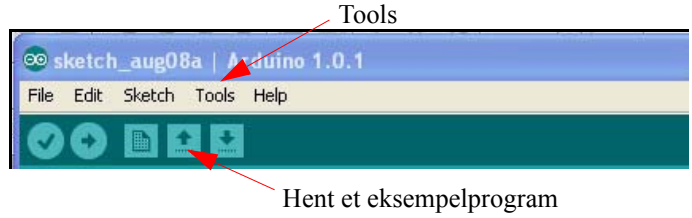
Arduino-editoren og kompilatoren kort sagt det som trengs hentes fra:

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

Aktuell versjon av Arduino er Arduino 1.0.5. Filen som har navnet *arduino-1.0.5-windows* er pakket som en zip-fil og er på ca. 52 Mbyte. En tilsvarende fil er tilgjengelig for Mac fra samme nettsted.

Installasjon av programvaren:

1. Klikk på fila ***Windows Installer, Windows (ZIP file)***
2. *For å pakke ut fila trenger du programmet WinRAR som kan hentes fra: http://www.rarlab.com/download.htm eller Winzip som kan hentes fra: www.winzip.com/downwz.htm*
3. *Velg Extract to fra menylinjen øverst og velg f.eks. C:/Programfiler og trykk OK. Programfilene legges da i en egen underkatalog (arduino-1.0.5) i katalogen Programfiler.*
4. *Programmet startes ved å klikke på programikonet:*  .
5. *Koble til USB-kabelen til ønsket port.*
6. *Klikk på Tool på menylinjen og velg Board. Her velges hvilken variant i Arduino familien du skal jobbe med. I dette tilfellet velg: Arduion UNO...*
7. *Klikk på Tool på menylinjen og velg Serial Port. Sjekk at riktig port (Com?) er valgt.*



Programmet skal nå være klart til bruk og du kan skrive inn programmet. Alternativt kan du hente opp et ferdig program som et første forsøk. Det gjør du ved å velge ikonet  fra menylinjen.

Velg ***01. Basic*** og ***Blink*** fra nedtrekksmenyene.

Programmet lastes inn i editoren og programlinjene framkommer i editeringsfeltet. Før programmet kan overføres til mikrokontrolleren på Arduino-kortet må det *kompileres*, dvs. overføres til en kode (binær kode) som mikrokontrolleren forstår. Dette kan gjøres ved å velge ikonet  . Dersom programmet inneholder ulovlige kommandoer eller skrivefeil, så vil kompilatoren varsle om det og vise på hvilken linje feilen er avslørt. Det er ikke nødvendigvis alltid der feilen er gjort.

Til slutt overføres programmet til mikrokontrollerens minne (Arduino-kortet). Dette gjøres ved å trykke på knappen  (som både, sjekker koden, kompilerer og overfører programmet til Arduino-kortet).

Manglende kontakt med kortet

Det hender at en ikke oppnår umiddelbar kontakt med Arduino-kortet når en forsøker å overføre 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 ikke tilkoblet eller ødelagt kabel
- Feil port er valgt av programeditoren
  Endres ved å velge: *Tool* og *Serial Port* fra menylinjen i editoren
- Feil type kontroller-kort valgt
  Endres ved å velge: *Tool* og *Board* fra menylinjen for så å velge rett kort, i vårt tilfelle *Arduino UNO*.
- Rx/Tx linjene har en tilleggsfunksjon som ikke er frakoblet under programmeringen. F.eks. ved at datainnsamlingsenheten ikke er frakoblet, eller at strappene J2 og J3 ikke er fjernet under programmeringen.
- Manglende driver
  Driveren er et program som gjør at PC-en og Arduino-kortet kan kommunisere med hverandre.

Ved installasjon av drivere gjøres følgende (Windows 7 - tilsvarende kan gjøres for XP):

- Åpne kontrollpanelet
  - Velg: *System og sikkerhet*
  - Velg: *System*
  - Velg: *Enhetsbehandling* (venstre meny)
  - Velg: *Andre enheter* eller *Porter* og finn den porten som Arduino er tilkoblet
  - Høyreklikk på porten og velg *Oppdater driverprogramvare* fra menyen
  - Velg: *Søk på datamaskin etter driverprogramvare*
  - Bla gjennom og finn katalogen *Arduino 1.0.5/Drivere* - pek på den og start installasjon

Om du er heldig installeres driveren og du får kontakt med Arduino-kortet.

5.2.2 Grunnleggende bruk av programeditoren

Kort oversikt over Arduino-editoren


Undersøker at koden er riktig ved å kompilere programmet

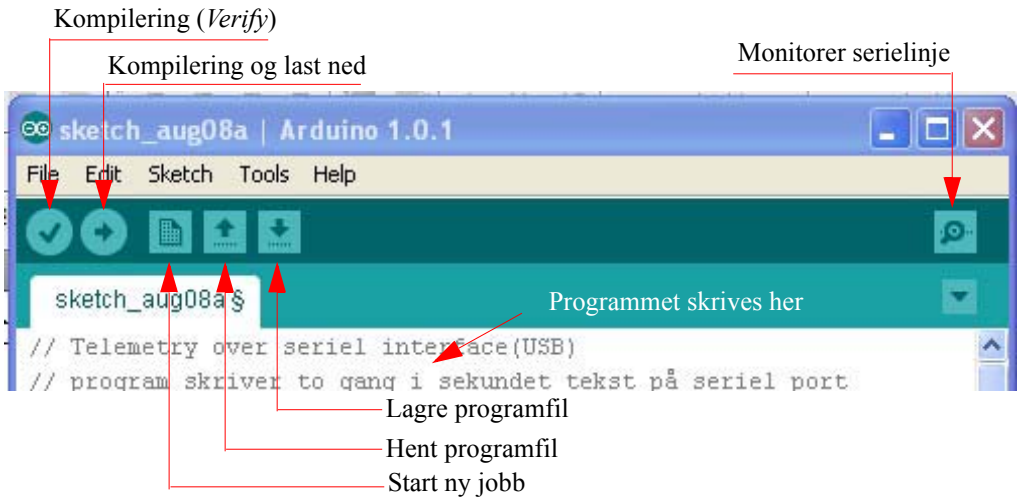

Kompiler og last ned programmet til kontrollenheten


Hent nytt "arbeidsark"


Hent en eksisterende programfil


Lagre programfil


Monitorer data sendt tilbake på serielinjen



Hjelp

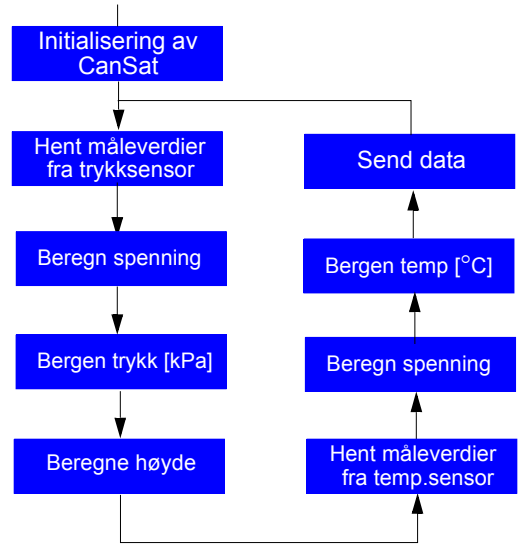
På den øverste menylinjen finnes menyen Help. Her finner du mange nyttige tips. Her er et lite utvalg:

- ***"Getting started"***
  Som gir deg hint om oppkobling av Arduino-kortet, installasjon av drivere og bruk av programeditoren med mer.
- ***"Environment"***
  Som viser deg hvordan du bør sette opp editoren på en best mulig måte, bruke programeditoren på en god måte, henting og lagring av filer med mer.
- ***"Reference"***
  Her finner du referansemanualen til de mest brukte funksjonene. Dersom du f.eks. lurer på hvordan du lager et *delay* på 1 sek., så kan du slå opp her.

- ***"Find in reference"***
  Dersom du merker en kommando, f.eks. *delay* og trykker <Ctrl shift f> kommer du rett inn i referansemanualen på funksjonen *delay*.

5.3 Programstruktur

Databehandlinsenheten (Arduino UNO) styrer hele prosessen i CanSat'en. Den henter inn data fra datainnsamlingsenheten (overlay kortet), omregner fra tallverdi til trykk, temperatur og høyde, tilrettelegger og sørger for å sende data til bakkestasjonen via kommunikasjonsenheten, eller til datalagringsenheten (Openlog). Dette gjøres i en gjentakende sløyfe som vist på figuren til høyre. En slik oversikt over programmet kalles et *flytdiagram*.



5.4 Viktige kommandoer

Referansemanualen til C for bruk ved programmering av Arduino-prosessorer finnes på følgende nettadresse:

http://arduino.cc/en/Reference/HomePage

Referanse manualen ligger også under *help* på menylinjen i programeditoren som vist på figuren til høyre.

5.4.1 Generelle kommandoer

Programstruktur

Programmet består av en rekke mindre rutiner omsluttet av klammeparenteser. I void setup() rutinenen initieres mikrokontrolleren, mens selve programmet legges under void loop() rutinen

```
void setup()
{
                                    <initiering>
}
void loop()
```

```
{
                                        <programkode>
}
```

Alle kommandoer må avsluttes med ; (semikolon)

Initiering av dataoverføring til PC

Under uttestingen kan det være praktisk at data leses tilbake til terminalen. Datahastigheten settes opp i setup-rutinen med kommandoen: Serial.begin(9600); her satt til 9 600 baud:

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

Kommentarer:

Kommentarer kan skrives hvor som helst og begynner med

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

Disse blir fjernet under kompilering og overføres ikke til mikrokontrolleren.

Deklarasjon av variable:

I C må alle variable deklareres før de kan brukes og gjerne i starten av programmet. Deklarasjonene må inneholde *type* og *navn* på variabelen:

Deklarering kan også gjøres innenfor hver subrutine. Slike variable gjelder da bare innenfor den rutinen de er definert:

```
void loop()
{
                                        Int a;// deklarasjon av 16 bit heltall (word)
                                        char b;// deklarasjon av 8 bit karakter (byte)
                                        char c, d;// deklarasjon av to 8 bits karakterer (byte)
                                        float e;// deklarasjon av variabelen e som et desimaltall f.eks.
1,65 (32 bit, dobbel word)
                                        unsigned long f;// deklarasjon av 4 byts heltallsvariabel f (32
bit) uten fortegn
                                        boolean g;// deklarasjon av en boolsk variabel g som kan ha
verdiene 0 og 1
                                        <programkode>
}
```

Skriv tilbake til PC skjerm:

Følgende kommandoer skriver en variabel eller en tekst tilbake på terminalvinduet i programeditoren.

```
Serial.print(a);                    // Skriver variabelen a til en linje på skjermen,
                                    // neste skrivekommando skriver på samme linje
Seriel.println(a);                  // Skriver variabelen a til en linje på skjermen,
                                    // neste skrivekommando skriver på ny linje
Seriel.println("Hallo");            // Skriver teksten Hallo til en linje på skjermen,
                                    // neste skrivekommando skriver på ny linje
```

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

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

Definer digitale porter som inngang eller utgang:

Kontrolleren ATmega 328 har en rekke porter, digitale og analoge. De digitale portene må defineres som inn- eller utgang. Dette gjøres i setup-rutinen:

```
void setup() {
                        pinMode(8,OUTPUT);// Definerer pinne 8 som utgang, dette gjøres under
setup
                        pinMode(7,INPUT);// Definerer pinne 7 som inngang, dette gjøres under setup
}
```

Lese og skriv til en digital port:

Digitale porter kan enten settes til høy eller lav spenning. Dette gjøres med følgende kommandoer:

```
  boolean bolsk;          // Definerer den boolske variabelen bolsk kan ha verdien 0 eller 1
void loop()
{
digitalWrite(8, HIGH);              //Setter port 8 høy (5 V)
digitalWrite(8, LOW);               //Setter port 8 lav (0 V)
bolsk = digitalRead(7);             // Leser den digitale verdien på port 7 og setter i variabelen
bolsk
}
```

Vent-kommando:

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

```
delay(1000);                        //Stopper programmet i 1000 msek (1 sek)
```

Aritmetiske 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;                    //Kvotienten av a / b settes i variabelen kvo
```

5.4.2 Avlesning av sensorer

Syntaksen for lesing fra en AD-konverter inngang kan skrives som:

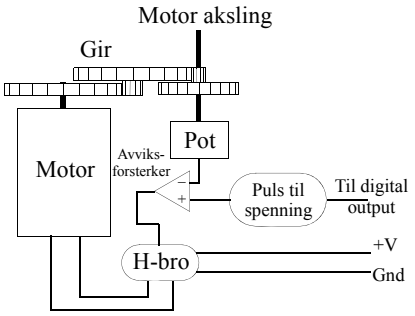
```
<variabel> = analogRead(<kanal>);    //Kanal kan ha verdiene 0 til 5
```

Eksempel 1:

```
Int VERDI;
VERDI = analogRead(0);               //Digitale verdien fra AD-kanal 0 leses inn i verdi
```

5.5 Programmering av servomotorer

En servo er en motor hvor en har full kontroll på vinkelen akslingen skal dreie. Normalt fra 0° til 180°. Dette skjer ved en intern tilbakekobling som vist i figuren til høyre. Servoen styres av en puls på styreinngangen. En *kontinuerlig roterende servo* kan dreie 360° og vil kunne fungere som en motor. Arduino har egne bibliotek som gir oss et sett av funksjoner som gjør det lett å programmere servomotorer. Vi får tilgang til biblioteket ved å inkludere "headeren" i starten av programmet:



Bibliotek

```
#include <Servo.h>
```

Det første man gjør er at deklarere de servoene man trenger, på samme måte som man deklarerer en variabel.

Deklarasjon

Servo <*servonavn*>*;*

<*servonavn*> er det valgte navnet på servoen. I alt kan en definere 12 ulike servoer for en Arduino UNO.

Tilkobling (tilordning)

Dernest må en fortelle Arduino'en om hvilken digital utgang <*pin*> som skal kobles til styreinngangen på servoen.

<*servonavn*>.attached(<*pin*>);

Frakobling

Denne kommandoen kobler fra den angitte servoen.

<*servonavn*>.detach();

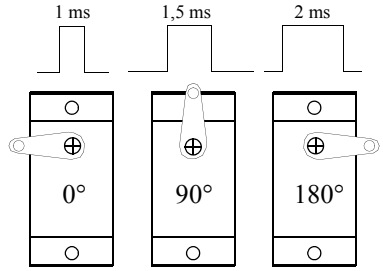
Den krever intet argument (pin), denne er gitt ved servoens navn.

Styrekommando - "write"

Servoen styres med kommandoen:

<*servonavn*>.write(<*vinkel*>);

Hos en *vanlig servo* vil vinkel angi dreiningsvinkelen til servoen i grader (0° – 180°). Hos en *kontinuerlig roterende servo* vil <*vinkel*> angi motorens hastighet og retning. Her vil 0° angi full hastighet i den ene retningen, 90° være i ro, mens 180° angir full hastighet i motsatt retning.



Egentlig styres servoen av pulser som antydet på figuren over.

Styrekommandoen – "writeMicroseconds"

Denne kommandoen fungerer omtrent som "write", men angir direkte lengden av pulsen i mikrosekunder.

<*servonavn*>.writeMicroseconds(<*µs*>);

Følgende er vanlige verdier for *en standard servo*:

1000µs dreies til posisjon 0°
1500µs dreies til posisjon 90°
2000µs dreies til posisjon 180°

Følgende er vanlige verdier for *en kontinuerlig roterende servo*:

1000µs full fart *mot* urviseren
1500µs i ro
2000µs full fart *med* urviseren

Servomotorene som benyttes av **Abot** har følgende spenn:

1300µs full fart *mot* urviseren
1500µs i ro
1700µs full fart *med* urviseren

Det er viktig å unngå at servoen står å stanger mot et ytterpunkt, da dette trekker mye strøm, samtidig som det kan skade komponenten.

Monitoreringskommandoen – "read"

Kommandoen "read" returnerer siste "write"-verdi sendt til servoen.

<*vinkel*> = <*servonavn*>.read();

Eksempel 1

Eksempelet viser hvordan to kontinuerlig roterende servoer programmers for at roboten skal øke for så å sakne farten:

```
// Sweep
// by BARRAGAN <http://barraganstudio.com> This example code is in the public domain.

#include <Servo.h>

Servo myleftservo;                      // create servo object to control a servo
                                        // a maximum of eight servo objects can be created
Servo myrightservo;                     // create servo object to control a servo
                                        // a maximum of eight servo objects can be created
int pos = 0;                            // variable to store the servo position

void setup()

{
myleftservo.attach(9);                  // attaches the left servo on pin 9 to the servo object
myrightservo.attach(10);                // attaches the right servo on pin 10 to the servo object

}

void loop()

{
 for (pos = 0; pos < 180; pos += 1)     // goes from 0 degrees to 180 degrees
 {                                      // in steps of 1 degree
  myleftservo.write(pos);               // tell servo to go to position in variable 'pos'
  myrightservo.write(pos);              // tell servo to go to position in variable 'pos'
  delay(15);                            // waits 15ms for the servo to reach the position
 }
 for (pos = 180; pos>=1; pos-=1)        // goes from 180 degrees to 0 degrees
 {
  myleftservo.write(pos);               // tell servo to go to position in variable 'pos'
  myrightservo.write(pos);              // tell servo to go to position in variable 'pos'
  delay(15);                            // waits 15ms for the servo to reach the position
 }
}
```

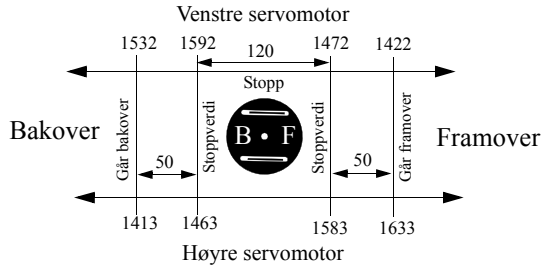
Måling av avvik fra rettlinjet bevegelse

Det er foretatt måling av terskelverdien for når servomotorene stopper og når de går rett fram. I følge beskrivelsen skal de være i ro ved en styrepuls på 1500 µsek.

Tabellen under viser at vinduet der servoene er i ro er begge på 120µsek., men noe skjevt i forhold til 1500 µsek. Dessuten er stoppområdene forskjøvet 9µsek. i forhold til hverandre. Vi legger også merke til at når verdien til venstre servo senkes fra 1472 til 1460 µsek. (dvs. 12 µsek.) og verdien til høyre servo økes fra 1583 til 1596 µsek. (dvs. 13 µsek.), så går roboten så og si rett fram.

Måling	Venstre	Diff.	Høyre	Diff	Middel
Går sakte rett fram	1460	-	1596	-	1528
Øvre terskel for stopp	1592	120	1583	120	
Nedre terskel for stopp	1472		1463		

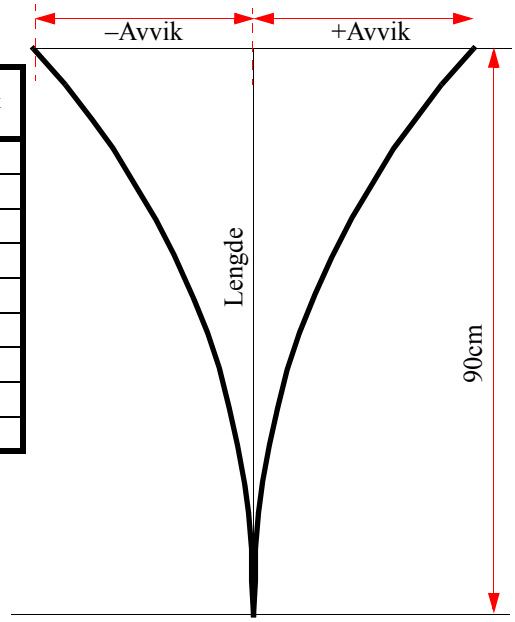
I figuren har vi illustrert stoppverdiene og vist hvilke verdier som gir bevegelse fram-over og bevegelse bakover. 50 er kun et eksempel for at roboten vil bevege seg fram eller tilbake.



Måling av avvik fra rettlinjet bevegelse

Det er utført målinger for avvik på en Abot. Målingene ble foretatt på et 90 cm bredt bord. Roboten startet med bakenden til roboten kant i kant med bordplata, og endte på den andre siden med forenden kant i kant med bordplata. Avviket ble målt som vist på figuren under.

#	Venstre pulslengde [µsek]	Høyre pulslengde [µsek]	Avvik
1	1422	1633	1,5 cm
2	1422	1633	1,0 cm
3	1422	1633	1,0 cm
4	1422	1633	0,5 cm
5	1422	1633	1,7 cm
6	1422	1633	1,0 cm
7	1422	1633	0,5 cm
8	1422	1633	0,5 cm
9	1422	1633	0,5 cm



Den venstre servoen er satt til 1422 hvilket er 50 “hakk” under stoppverdien. Tilsvarende er den høyre servoen satt til 1633 som er 50 “hakk” over stoppverdien. Dermed skal de to servomotorene bevege seg med like stor hastighet, men i motsatt retning som er en forutsetning for at roboten skal bevege seg rett fram og i samme retning. Som vi ser av måleresultatene i tabellen over så er avvikene for den aktuelle distansen minimale. Roboten skulle derfor ha gode forutsetninger for å kunne brukes i en konkurranse hvor det gjelder å treffe et mål.

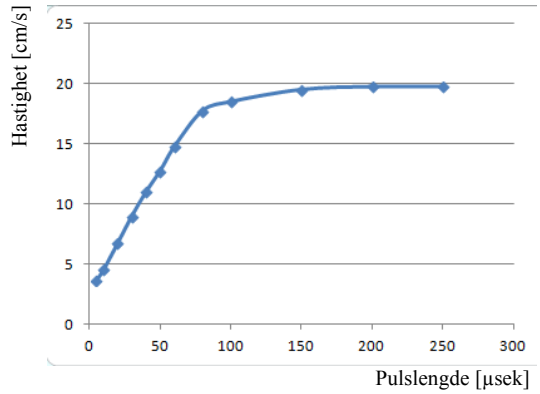
Funksjon – Rettlinjet bevegelse

```
#include <Servo.h>

Servo myleftservo;         // Definer venstre servomotor
Servo myrightservo;        // Definer høyre servomotor
int fart = -50;            // positiv hastighet framover, negativ hastighet bakover

void setup()
{
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);
  myleftservo.attach(10);        // Forbind venstre servomotor til pinne 9
  myrightservo.attach(9);        // Forbind venstre servomotor til pinne 10
}

void loop()
{
  rettFart(fart);
}
}

void rettFart(int lokalFart)
{
     if (lokalFart > 0)
    {
     myrightservo.writeMicroseconds(1583 + lokalFart);          // Høyre servo forover
     myleftservo.writeMicroseconds(1472 - lokalFart);           // Venstre servo bakover
     delay(100);
    }
     else if (lokalFart < 0)
    {
     myrightservo.writeMicroseconds(1463 + lokalFart);          // Høyre servo bakover
     myleftservo.writeMicroseconds(1592 - lokalFart);           // Venstre servo forover
     delay(100);
    }
}
```

Måling av hastighet og kjørelengde

Vi skal nå måle hastigheten som funksjon av pulsbreddeverdien. Dette gjøres ved at vi setter kjøretiden fast til, f.eks. 2000 msek. Så måler vi kjørelengden, for så å beregne hastigheten

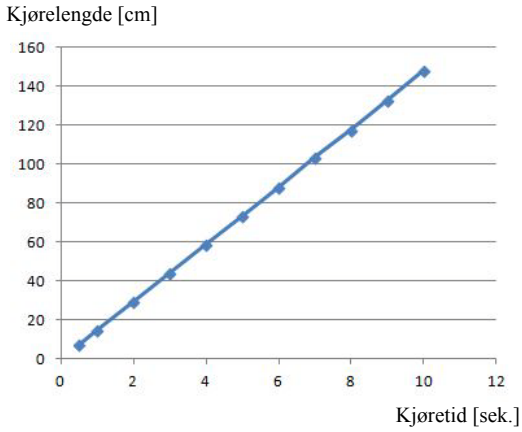
#	Pulslengde [µsek]	Kjørelengde [cm]	Hastighet [m/s]
1	5	7,3 cm (h+)	3,65 cm/s
2	10	9,3 cm (h+)	4,55 cm/s
3	20	13,5 cm (h)	6,75 cm/s
4	30	18 cm (h)	9 cm/s
5	40	22 cm (h)	11 cm/s
6	50	25,5cm	12,75 cm/s
7	60	29,5 cm	14,75 cm/s
8	80	35,5 cm	17,75 cm/s
9	100	37 cm	18,5 cm/s
10	150	39 cm	19,5 cm/s
11	200	39,5 cm	19,75 cm/s
12	250	39,5 cm	19,75 cm/s



Figur 5.1 Figuren viser hastighet som funksjon av puls-lengden som er omtrent linear fra 5 til ca. 80 µsek. Det synes derfor hensiktsmessig a velge en hastighet i området 50–70 µsek for sa a variere kjøretiden for å endre kjørelengden.

Dersom tiden det tar å nå målet ikke er av betydning, vil det være mest hensiktsmessig å benytte fast hastighet (f.eks. 50–70 µsek.), og heller angi kjørelengden som funksjon av tiden. I den neste målingen har vi derfor valgt å holde farten konstant (60 µsek) og finne sammenhengen mellom kjørelengden og den tiden servomotorene går.

#	Kjøretid [sek]	Kjørelengde [cm]
1	0,5	7,4
2	1	15,0
3	2	29,5
4	3	44,2
5	4	58,9
6	5	73,3
7	6	88,0
8	7	103,5
9	8	117,6
10	9	132,6
11	10	147,9



Figur 5.2 Figuren viser kjørelengde som funksjon av kjøretid. Pulslengden er satt fast til 60 µsek.

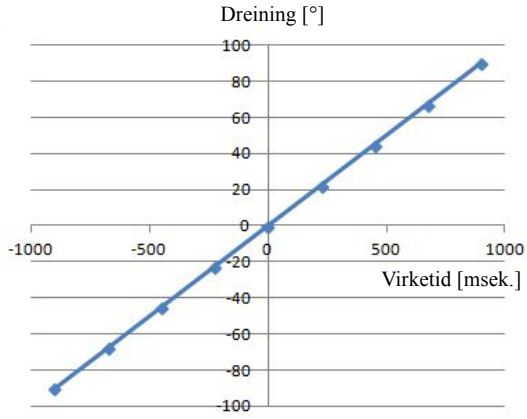
Av figur 5.2 ser vi at det som ventet er en lineær sammenheng mellom kjøretid og kjørelengde. Vi ser videre at forholdet mellom kjøretid i msek og kjørelengde i cm er 10000/147,9 = 67,61. Vi kan dermed angi ønsket kjørelengde i cm for å så multiplisere med 67,61 og dermed få kjøretiden.

Måling av svingvinkel

Vi skal nå undersøke robotens evne til å svinge til høyre og venstre. Dette gjøres ved at høyre og venstre servomotor roterer i motsatt retning (egentlig i samme retning siden de er plassert speilvendt i forhold til hverandre). Svingvinkelen bestemmes ved å endre hvor lenge vi lar dreiningen virke. Vi definerer at en positiv (+) vinkelverdi dreier roboten mot klokka, mens en negativ (–) vinkelverdi dreier med klokka.

Vi velger å la 900 msek tilsvare 90° (mot klokka) og tilsvarende – 900 msek tilsvare – 90° (med klokka). Dernest justerer vi motorenes hastighet slik at roboten dreier 90° i løpet av det aktuelle tidsrommet. Dermed har vi fått en enkel sammenheng mellom svingvinkel og tiden vi lar dreiningen virke.

#	Virketiden [msek]	Dreining [°]
1	900	90°
2	675	67,5°
3	450	45°
4	225	22,5°
5	0	0°
6	– 225	– 22,5°
7	– 450	– 45°
8	– 675	– 67,5°
9	– 900	– 90°



Funksjon – Svingbevegelse

```
// Styring av høyre og venstre servomotor for svingbevegelse.
// Nils Kr. Rossing, Skolelaboratoriet ved NTNU

#include <Servo.h>

Servo myleftservo;   // Definer venstre servomotor
Servo myrightservo;  // Definer høyre servomotor

int run = 0;
int drei = 900;      // Angi vinkel i 1/10-dels grad
                     // Positiv verdi dreier mot klokke
                     // Negativ verdi dreier med klokka

void setup()
{
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);
  myleftservo.attach(10);  // Tilknytt venstre servo til pinne 10
```

```
  myrightservo.attach(9);  // Tilknytt venstre servo til pinne 9
}

void loop()
{
  sving(drei);
}

void sving(int lokalDrei)
{
  int dreieFart = 35;
     if (lokalDrei > 0)
    {
     myrightservo.writeMicroseconds(1583 + dreieFart);   // Sett høyre servo til å gå forover
     myleftservo.writeMicroseconds(1592 + dreieFart);   // Sett venstre servo til å gå bakover
      delay(lokalDrei);
      myrightservo.writeMicroseconds(1500);               // Stopp høyre servo
      myleftservo.writeMicroseconds(1500);                // Stopp venstre servo
    }
    else if (lokalDrei < 0)
    {
     myrightservo.writeMicroseconds(1463 - dreieFart);   // Sett høyre servo til å gå bakover
     myleftservo.writeMicroseconds(1472 - dreieFart);   // Sett venstre servo til å gå forover
      delay(abs(lokalDrei));
      myrightservo.writeMicroseconds(1500);               // Stopp høyre servo
      myleftservo.writeMicroseconds(1500);                // Stopp venstre servo
    }
}
```

6 Hjelpeprogrammer

I dette kapittelet skal vi beskrive noen programmer som kan være nyttige som støtte når vi arbeider med bygging av elektroniske kretser.

6.1 Fritzing

6.1.1 Bruksområde

Fritzing er et tegneprogram for tegning av oppkoblinger på koblingsbrett. Programmet har et fyldig bibliotek med komponenter og kretskort fra bl.a. Arduino og PIC. Programmet egner seg derfor glimrende for dokumentasjon.

Når man har montert komponentene på koblingsbrettet, kan man konvertere tegningen til et koblingsskjema som viser komponentsymbolene. Derfra kan

6.1.2 Installasjon

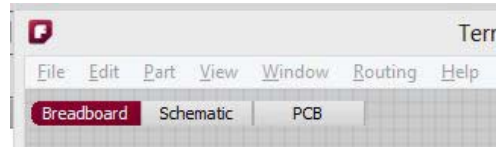
Programmet finnes i skrivende stund som en Beta-versjon som fritt kan lastes ned fra nettstedet: www.fritzing.org

6.1.3 Introduksjon til bruk

Når man starter Fritzing kommer følgende introduksjonsvindu opp.

Et standard koblingsbrett ligger klart til bruk i arbeidsfeltet til venstre. Til høyre finner vi *komponentbiblioteket*. Flere *spesialbiblioteker* kan hentes inn ved å trykke på ikonene til venstre for komponentbiblioteket. Ved hjelp av glidebryteren nederst kan en zoome inn og ut i arbeidsfeltet (hjulet på musa kan brukes på samme måte).

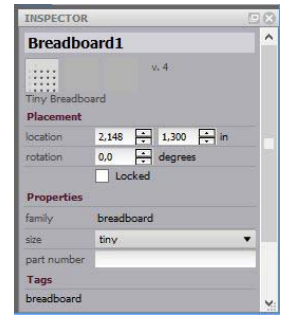
På menylinjen oppe til venstre finnes tre faner:



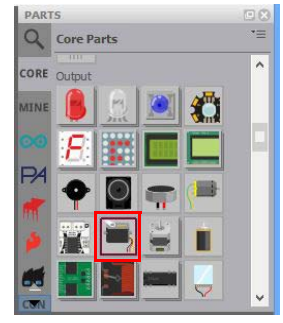
- ***Breadboard*** – Denne fanen gir deg mulighet til å konstruere kretser på koblingsbrett. Dette er et særdeles nyttig hjelpemiddel når du skal dokumentere dine konstruksjoner.
- ***Schematic*** – Ved å velge denne fanen overføres konstruksjon din til kretsskjema, Riktignok er programmet ganske primitivt slik at du selv må plassere og dreie komponentene slik at skjemaet ser greit ut.
- ***PCB*** – Ved å velge denne fanene overføres kretsskjemaet til "layout" delen av programmet. Her kan du få lagt skjemaet ut som et kretskort og automatisk få lagt ut kortet med kobberbaner. Enten på ensidig eller tosidig. Så kan en generere ulike filer som kan sendes inn for produksjon av kretskortet. Også her må en hjelpe programmet med komponentplasseringen.

6.1.4 Oppbygging på koblingsbrett (Breadboard)

Vi skal her beskrive oppbygging av en liten krets på koblingsbrettet trinn for trinn. Som eksempel velger vi å bygge opp en roboten Abot som styres ved hjelp av en Arduino UNO. Sammenkoblingen av servoer og avstandssensoren bygges opp på et lite (tiny) koblingsbrett.

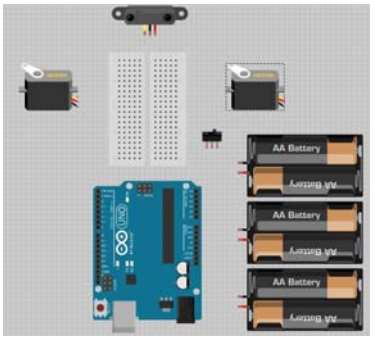


1. **Fjern *First time help* fra arbeidsfeltet**
   Velg *Help* fra øverste menylinje og ta bort hake på *First time help,* eller trykk på den vesle krysset øverst i venstre hjørne.
2. **Velg størrelse på koblingsbrettet**
   Klikk på koblingsbrettet som ligger i arbeidsfeltet slik at koblingsbrett menyen kommer opp i komponentfeltet til høyre. Velg *tiny* for størrelse (*size*). Grip brettet med cursoren og legge det mitt på arbeidsfeltet.
3. **Drei koblingsbrettet**
   Komponenter kan dreies 90° ved å høyreklikke på komponenten for så å velge *Rotate* og ønsket dreining og retning. Koblingsbrettet kan bare roteres et helt antall 90°.
4. **Hent inn komponentene**
   Til vårt formål trengs to 360° servoer. En slik finnes i komponentmenyen, øverst til høyre, under kategorien *output*. Servoer har gjerne tre ledninger (en *gul*, en *rød* og en *sort*). Rød er +5V, sort er GND (0V), og gul er styreinngangen som bestemmer dreiehastig-



heten. Videre hentes Arduinoen inn og legges i arbeidsfeltet. En avstandssensor (IR proximity sensor) og en bryter (toggel switch) hentes også inn i arbeidsfeltet slik at resultatet blir omtrent som på figuren under til høyre.

5. **Batteri**
   Vi har her valgt å benytte et 9 V batteri. Siden Arduinoen og komponentene normalt benytter 5V, så har vi valgt å benyttet 5V regulatoren på Arduino-kortet til å senke spenningen fra 9V til 5V.



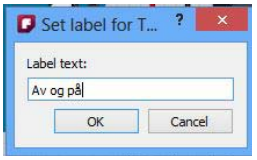
6. **Fest komponentene**
   Dersom en plages med ufrivillig flytting av komponenter. Høyreklikker man på komponenten og velger *Lock Part* i nedtrekksmenyen.

7. **Trekk forbindelseslinjer**
   Dernest trekkes forbindelser mellom terminalene til komponentene på følgende måte: Finn underkategorien *Breadboard view* i komponentmenyen og velg ikonet  . Forbind så to terminaler med hverandre ved å klikke på den ene og dra musa til den andre, mens venstreknappen holdes nede. Endepunktene skal feste seg til terminalene.

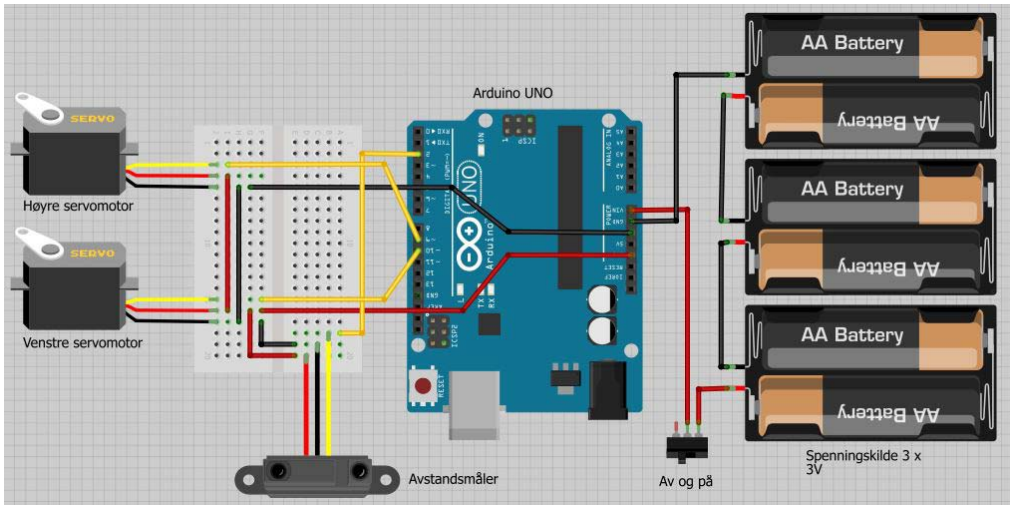
8. **Lag og fjern knekkpunkter på forbindelse**
   Dersom en ønsker en knekk på ledningen, klikker man på ledningen omtrent der en ønsker en knekk. Dra så i knekkpunktet til ønsket posisjon. Et knekkpunkt kan fjernes ved høyreklikke på punktet, for så å velge *Remove bendpoint* i nedtrekksmenyen.

9. **Endre farge på forbindelseslinjene**
   Det kan være lurt å merke forbindelse med ulike farger. F.eks. rødt for + og sort for –. Dernest kan en velge fritt slik en finner det hensiktsmessig. Farge velges ved å høyreklikke på forbindelsen og så velge *Wire color* i nedtrekksmenyen.

10. **Legge til merkelapper til komponenter**
    Dersom man ønsker å sette en merkelapp på en eller flere av komponentene, høyreklikker man på komponentene og velger *Show part label*, dermed kommer det opp en default merkelapp. Ved å dobbelklikke på denne, kommer det opp en innboks (se figur til høyre) hvor man kan erstatte innholdet med det man måtte ønske.

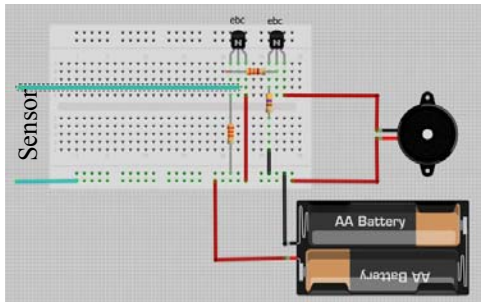


11. **Ferdig oppkobling**
Figuren under viser den ferdige oppkoblingen som enten kan brukes for dokumentasjon, eller hjelp for elevene å koble opp etter oppskrift om det er ønskelig.

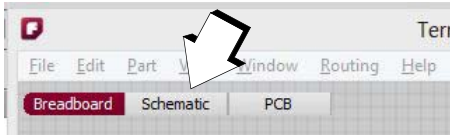


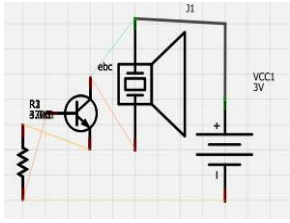
6.1.5 Overføring til koblingsskjema

Etter å koblet opp kretsen på koblingsbrettet og testet at den fungerer som ønsket, kan kretsen overføres til et standard koblingsskjema. Som et eksempel har vi valgt kretsen vist på figuren til høyre.



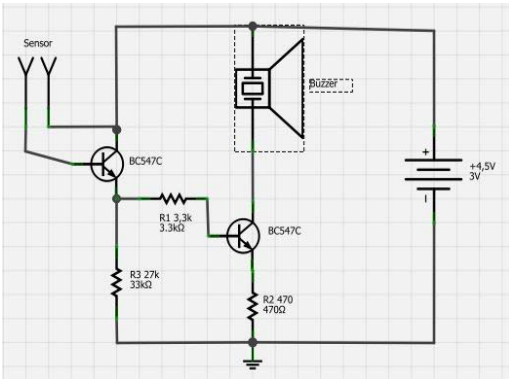
1. **Konvertering til koblingsskjema**
Ved hjelp av menyen øverst i venstre hjørne kan oppkoblingen på koblingsbrettet konverteres til et koblingsskjema med symboler ved å velge *Schematic*.





2. **Justering av skjema layout**
Den automatiske konverteringen gir ingen god layout som vist nederst på figuren til venstre. *De to transistorene og de tre motstandene er plassert oppå hverandre.* Derfor er antallet komponenter for lav. Skjemaet må derfor justeres for hånd. Dette gjøres ved å flytte komponentene slik de normalt vil være plassert i et koblingsskjema, med signalflyten fra venstre mot høyre. Flyttingen skjer ved å "gripe" komponenten ved å venstreklikke på komponenten, for så å dra den dit den skal.

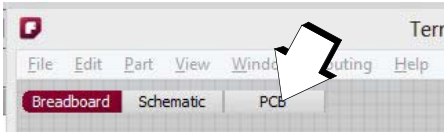
3. **Trekke opp forbindelsene på nytt**
   Etter at komponentene er plassert der de skal være, trekkes forbindelsene opp på nytt. Komponentene er forbundet med tynne streker som indikerer sammenkoblingene. Om noen forbindelser mangler, skyldes det at det er brudd i forbindelsen på koblingsbrettet. Om dette er tilfelle går en tilbake og retter opp forbindelsen. I det en trekker opp forbindelsene på nytt, linjene bli markert med en tykkere strek. Figuren til høyre viser resultatet.

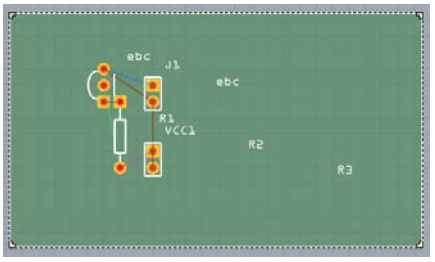


4. **Tekst**
   Teksten kan endres ved å dobbelklikke på tekstrubrikken. Skriv ny tekst inn i innboksen og trykk OK. Tekstfeltet kan så flyttes til ønsket posisjon.

6.1.6 Trykte kretskort

Fra skjema til PCB (Printed Circuit Board)

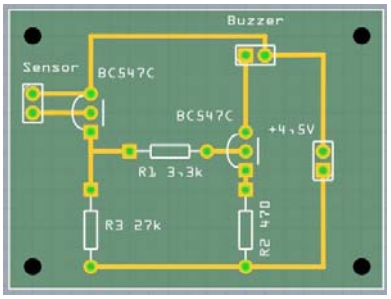




Når skjemaet er klart, kan dette rutes til et kretskort. Ved å trykke på PCB på menyen (øverst på figuren til høyre) konverteres skjemaet til en layout. Denne er svært uferdig og forutsetter at en manuelt plasserer komponentene slik at PCB-layout-en blir tilfredsstillende (nederst på figuren til høyre).

Vi legger merke til at begge transistorene legges på samme sted, tilsvarende med motstandene.

Layout

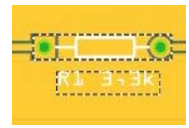


I utgangspunktet er forbindelseslinjene tynne rette linjer som knytter sammen beina til komponentene. Når komponentene er plassert kan en begynne å trekke de endelige ledningsforbindelsene. Idet man venstreklikker på en av linjeforbindelsene vil denne konverteres til en kobberbane med tilhørende loddeland. Knekkpunkter langs kobberbanen legges inn ved å klikke på banen. Ved å høyreklikke på et knekkpunkt kan en fra nedtrekksmenyen velge å fjerne punktet.

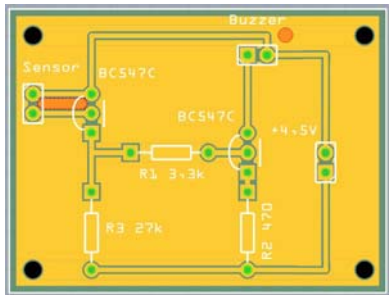
På et tosidig kort ligger de gule linjer på loddesiden av kretskortet, og de oransje linjene ligger på komponentsiden. Dersom en ønsker at alle linjene skal ligge på loddesiden høyreklikker man på kobberbanen og velger *Move to top layer*.

Tekst:

Ved å venstreklikke på komponenten markeres både komponent og tilhørende tekst. En nå skrive inn ny tekst i tekstrammen samtidig som man kan gripe ramma og flytte den dit man ønsker.



Jordplan:



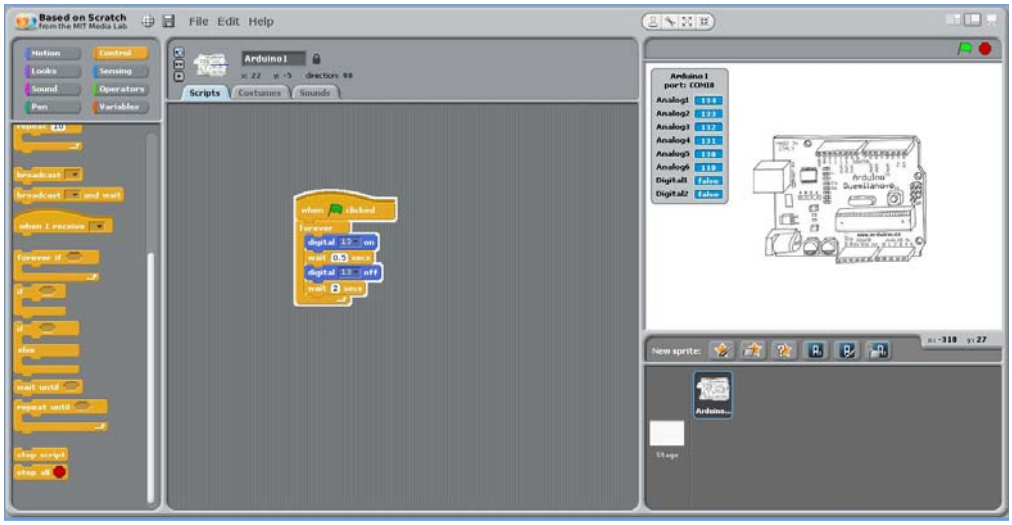
Det er ikke uvanlig at alle mellomrom mellom kobberbanene fylles med en kobberflate. Ved å forbinde denne til jord, vil man skape et robust jordplan som reduserer spredning av elektrisk støy. Et slikt jordplan legges inn i mellomrommene ved å dra ikonet  fra komponentkategorien *PCB view.*

6.2 Scratch for Arduino (S4A)

Scratch er et allsidig program utviklet av MIT som både kan brukes til å programmere animasjoner og til å programmere mikrokontrollere ala Arduino. Scratch benytter flow-programmering som er et høynivå grafisk programmeringsspråk, hvor programelementene framstilles som byggeklosser som kan settes sammen som LEGO klosser til et flytdiagram.

I denne sammenheng skal vi se på *Scrath for Arduino* (S4A) som er en variant av Scratch utviklet av *Citilab,* og som gjør det mulig å programere Arduino ved hjelp av grafiske kommandoer.

Figuren under viser brukergrensesnittet mot programmet S4A:



Ulempe

- Programmer laget og overført til mikrokontrolleren ved hjelp av S4A, krever kontinuerlig forbindelse mellom S4A programmet på PC-en og Arduino mikronotrollerkortet. Programmene kan derfor ikke kjøres med mindre kabelen er tilkoblet.

La oss først se hvordan vi skal installere programmet.

6.2.1 Installasjon

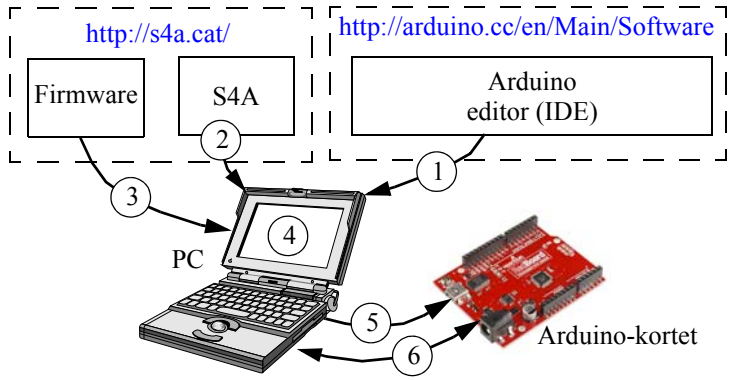
Følgende gir en beskrivelse av installasjon av Scratch for Arduino, *S4A*.

1. Det forutsettes at programeditor og kompilator for Arduino er installert på PC-en. Om dette ikke alt er gjort kan programmet hentes fra nettstedet: http://arduino.cc/en/Main/Software og installeres.
2. Gå til nettstedet http://s4a.cat/ for å laste ned Scratch for Arduino
3. Finn figuren vist under, på nettsiden:
   Rett under figuren finnes *S4A* for nedlasting og installasjon.

4. **Installasjon av programmet *S4A***
   Velg riktig plattform. Last ned og installer programmet. Følgende icon dukker på på skrivebordet:

5. **Firmware.**
   For at *S4A* skal fungere må det installeres en programvare på Arduino-kortet som mottar og administrerer prorammene fra *S4A*. Denne programvaren går under navnet "firmware".
6. **Nedlasting av "Firmware"**
   "Firmware" er en Arduino programfil skrevet i Arduino C (*S4AFirmware15.ino*). Denne lastes ned og legges på et sted hvor du lett kan finne igjen den.
7. **Installasjon av "Firmware"**
   Åpne editoren for Arduino og åpne programmet (*S4AFirmware15.ino)* med editoren. Sørg for at du har kontakt med kortet ved å velge riktig kort og riktig serieport.
8. Overfør filen til Arduino-kortet på vanlig måte ved hjelp av "Upload" knappen og kortet skal være klart for å motta programmer utviklet i *S4A*. Husk at det alltid må være forbindelse mellom *S4A* og Arduino-kortet for at programmene skal fungere.
9. Når du nå starter *S4A* så skal programmet automatisk finne porten der Arduino-kortet er tilkoblet og opprette forbindelse. Du er nå klar til å begynne å lage programmer ved hjelp av flow-diagrammer og kjøre på Arduino-kortet.

La oss se litt på hvordan dette fungerer:



1. Først hentes og installeres IDE på PC-en, programeditoren fra Arduino.
2. Dernest hentes og installeres Scratch for Arduino fra s4a.cat/.
3. Så hentes firmwaren fra s4a.cat/ til PC-en.
4. Deretter åpnes IDE og man henter firmwaren inn i editoren som et vanlig C-program.
5. Firmwaren kompileres og lastes ned til Arduinokortet på vanlig måte.
6. S4A startes og skal nå automatisk koble seg opp mot Arduino-kortet og starte programmet.

Normalt vil programmer som overføres til Arduino-kortene kunne kjøres på kortet uavhengig av PC-en og vil, når kortet får spenning, kunne kobles fra PC-en. Slik er det *ikke* med Scratch for Arduino, her kjøres programmet i PC-en, som så sender kommandoer over til firmwaren som utfører kommandoene og setter inn- og utganger høyt og lavt, leser av analoge innganger osv.

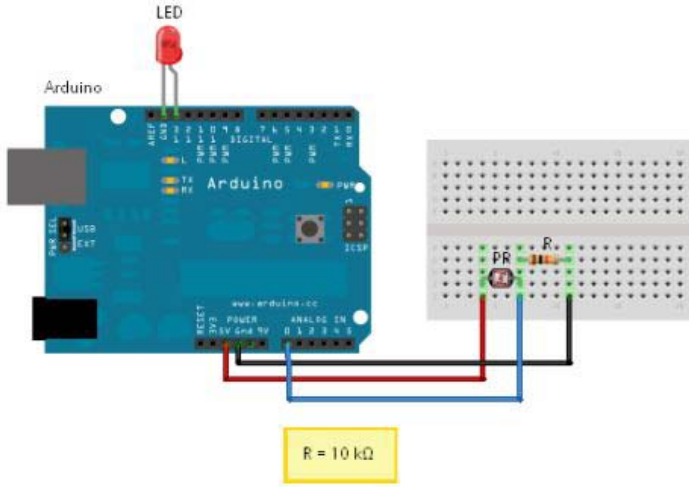
Ord og begreper[5]:

- **IDE** - *Integrated Development Environment* - utvilingsverktøy for programvare og maskinvare for Arduino-familien. Omfatter *editor, kompilator, debugger*, *nedlaster* m.m. Verktøyet bruker en variant av C, som er standard innenfor industri og utdanning.
- **Scratch** - Et visuelt programmeringsspråk rettet mot utdanning innen realfag. Ikke spesielt dedikert Arduino, men også Arduino (og LEGO MINDSTORMS o.l.). Utviklet ved MIT av *The Lifelong Kindergarten group*.
- **S4A** - Scratch for Arduino utviklet av *Citilab*
- **FW** - Firmware (norsk: “Fastvare”) - innebygget teknisk programvare som modifiserers av produsenten, men ikke av sluttbruker. I denne sammenhengen er det kun FW som legges over på Arduino-kortet, selve programmet kjøres i PC-en, men FW fungerer som en utfører av kommandoene fra PC-en.

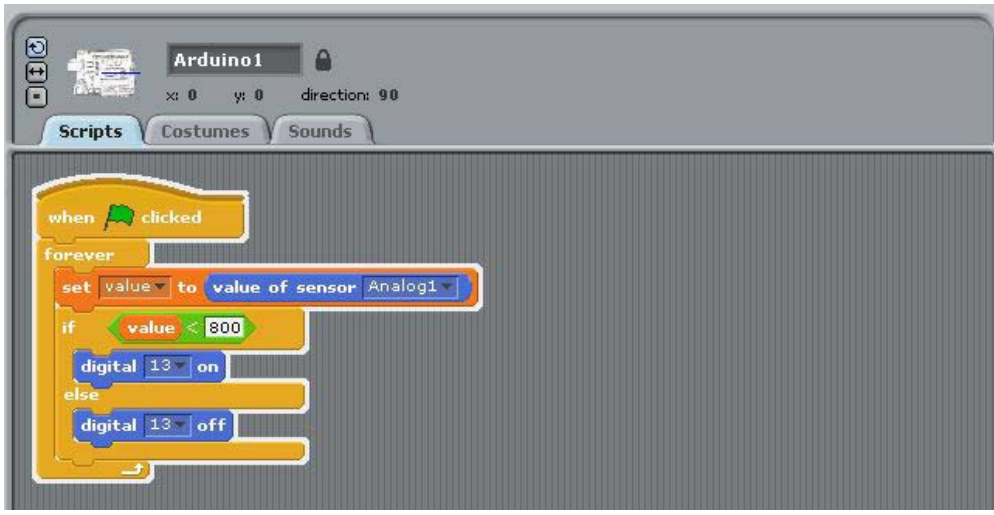
5. Skrevet av Åge Eriksen.

6.2.2 Mitt første program i *S4A* (tutorial)

Det aller enkleste er kanskje å laste ned noen ferdige programeksempler som ligger på http://s4a.cat/. Figuren under viser ett av disse programmene, med tilhørende oppkobling illustrert ved hjelp av Fritzing:

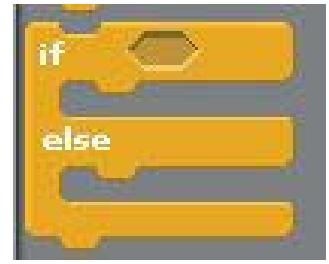


Figuren under viser programmet slik det framstår i *S4A*:



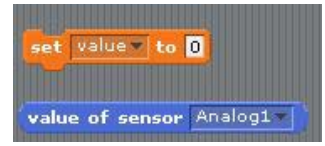
- **"When click on flag"**
  Ved å starte programmet med dette flow-symbolet vil programmet starte når man trykker på det grønne flaget, enten på ikonet eller på det grønne flagget øverst lengst til høyre i skjermbildet.

- **"if ... else ..."**
  Dernest følger flow-symbolet for **"if...else..."** hvor en kan legge inn kommandoer i gapene etter *if* og *else*. Disse gapene vil utvide og tilpasse seg til de nye kommandoene som legges inn.



- **Les en sensorverdi inn i variabel.**
  Når man skal lese av en analog eller en digital inngang og legge verdien inn i en variabel, velger man først ikonet:

  hvor man definerer nye variabler, med ønsket navn. Dernest henter man fram symbolene: ***"set value to"*** og ***"value of sensor Analog 1"***. Ved å flytte det blå flow-symbolet over verdien i det oransje flow-symbolet, får man en kommando som leser av en port (i dette tilfellet *Analog 1*) og tilordner den til verdien *"value"*. Den resulterende kommandoen blir seende ut som vist i figuren under.



- **Operatorer**
  "If...else..." flow-symbolet krever en operator, hvor den avleste verdien sammenlignes med en fast verdi. Fra **"operator"**-menyen (grønn) og **"variable"**- menyen (oransje) henter man de to flow-symbolene vist i figuren til høyre. Ved å "drag and drop" **"value"** flow-symbolet til input-boksen til venstre i operator-symbolet, og skrive inn en verdi (her 800) i inputboksen til høyre, får man den ferdige operatoren vist til nederst til venstre på figuren til høyre.

- **Styr digital utgang**
  Til sist ønsker vi å slå en digital utgang på eller av. Dette gjøres ved hjelp av **"digital on"** og **"digital off"** flow-symbolene som finnes under ikonet **"Motion"**. Man kan imidlertid bare velge mellom portene 10, 11 og 13 som digitale utganger.

6.2.3 Bruk av porter ved bruk av *S4A*

En Arduino UNO har 14 digitale og 6 analoge porter. Normalt kan alle de digitale defineres som både inn- og utganger. *S4A* legger imidlertid begrensninger på disse mulighetene. Følgende begrensninger gjelder:

- 6 analoge innganger: *Analog 1 til 6 (A0 – A5)*

- 2 digitale innganger: *Digitale pinner 2 og 3*
- 3 analoge utganger (pwm): *Digitale pinner 5, 6 og 9*
- 4 digitale utganger for servomotorer (kontinuelig rotasjon): *Digitale pinner 4,7, 8 og 12*

S4A tillater at man styrer flere Arduino-kort samtidig. Maksimalt antall kort er gitt av antall USB-porter på PC-en.

6.3 Ardublocks

Ardublocks er et grafisk programeringsverktøy som ytre sett ligner på Scratch ved at programmet bygges opp ved hjelp av grafiske byggeblokker. Ardublocks har noen klare fordeler framfor Scratch for Arduino:

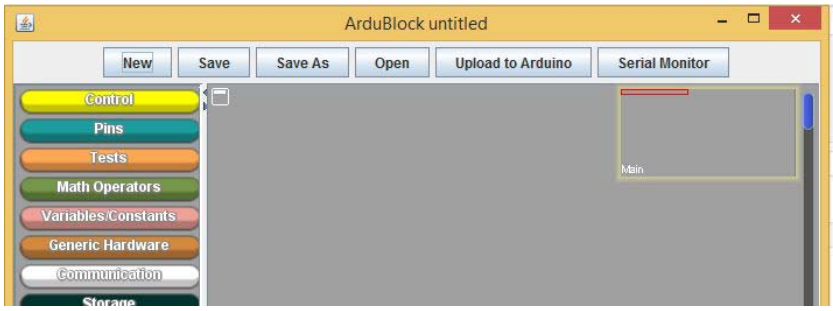
- Ardublocks er integrert i Arduino's editor (IDE) og kan velges inn fra Tools-menyen
- Det grafisk skrevne programmet oversettes til Arduino C-kode og vises i IDE etter at det er kompilert. På denne måten kan en lett se sammenhengen mellom Ardublocks-kommandoer og de grafiske programmeringselementene
- Det kompilerte programmet legges i sin helhet over på Arduino kortet og krever derfor ingen forbindelse med PC'en etter nedlasting til forskjell fra Scratch.

På den annen side så har Ardublocks ingen muligheter til å lage animasjoner som en introduksjon til programmering slik Skratch har.

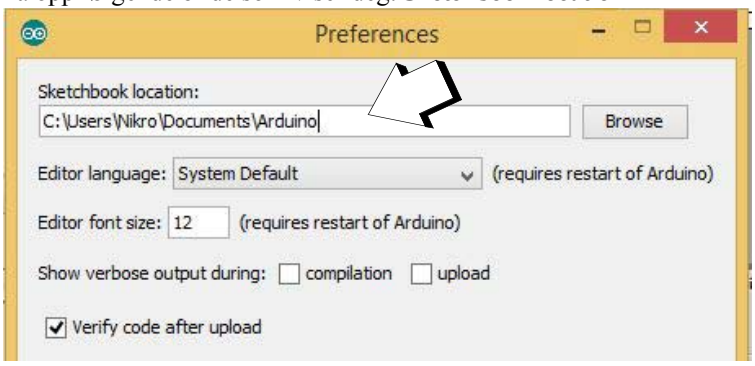
6.3.1 Installasjon

Gå til: http://sourceforge.net/projects/ardublock/files/ . Her finner du en oversikt over de siste versjonene av Ardublocks. Som du vil se så er det mange beta-versjoner, dvs. de er ennå ikke testet ut skikkelig og kan inneholde feil. Vi anbefaler likevel å laste ned den mest komplette versjonen (pr. 05.12.2014)

1. Velg: Download ardublock-beta-20140702.jar (9.6 MB)
2. Etter noen få sekunder starter nedlastingen som går fort.
3. I windows 8.1 legges filen i katalogen: ***Nedlastinger***
   Gå til katalogen og dobbelklikk på filen: ardublock-beta-20140702. Du vil da starte en Java-applikasjon som ser slik ut:

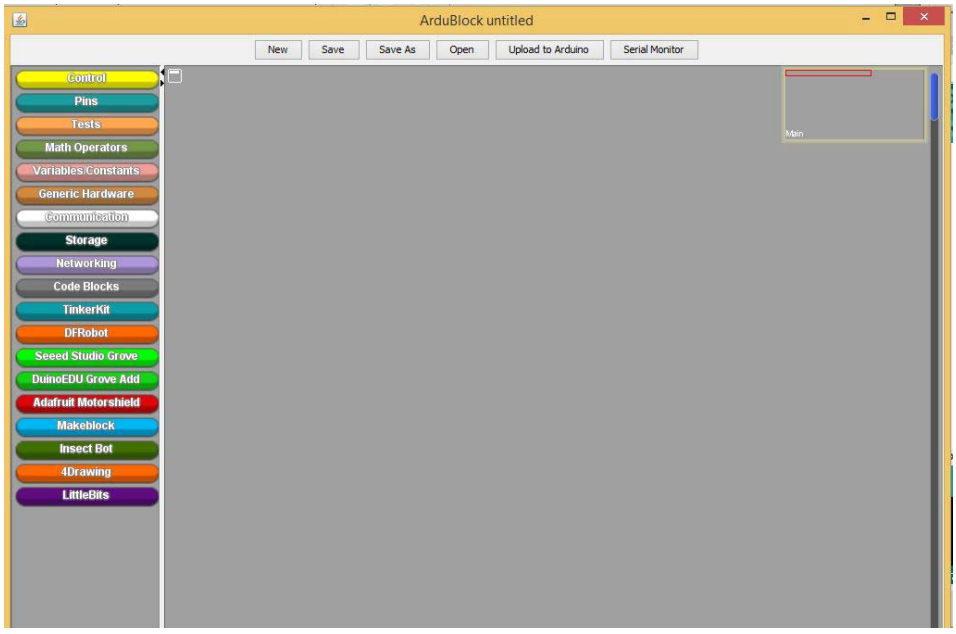


4. Vi skal nå integrere denne i den tradisjonelle Arduino editoren (IDE)
   Åpne Arduino editoren og velg: ***File***
   Deretter velg: ***Preferences***
   Du vil da få opp følgende bilde som viser deg: ***Sketchbook location***



   I mitt tilfelle: C:\Users\Nikro\Documents\Arduino
   Velg eller lag underkatalogen: tools
   Opprett følgende underkataloger under tools\ArduBlockTool\tool\
   Legg den nedlastede filen: ***ardublock-beta-20140702*** i denne katalogen.

5. Start Arduino editoren og velg menyen ***Tools.***
   Under rullegardinmenyen ***Tools*** skal du nå finne et valg som heter: ***ArduBlocks***
   Ved å klikke på denne åpnes ***ArduBlocks*** inne i Arduino-editoren



6.3.2 Bruk av Ardublocks

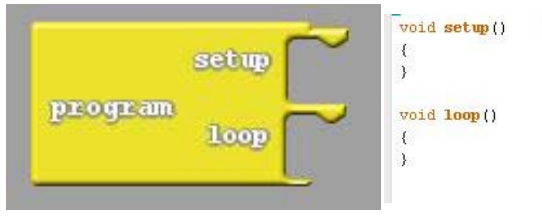
Vi skal nå ganske kort vise hvordan man kan bygge opp et programved hjelp av Ardublocks. I menyen langs venstre side er de ulike blokkene organisert i grupper. Disse er de viktisgste gruppene:

- ***Control* (gul)**
  Her finner du programstrukturen med Setup og Loop samt if (), for (), while (),delay, subrutine() osv.
- ***Pins* (blå)**
  Her kan du sette digitale innganger til høy eller lav, eller analoge utganger til en gitt verdi. Du kan slå på intern "pull-lup" motstand og slå av og på toner. I Ardublocks trenger du ikke eksplisitt å definere digitale porter som inn- eller utganger. Programmet definerer selv inn- og utganger i henhold til bruken.
- ***Tests* (beige)**
  Her finner du blokker som kan utføre tester, dvs. du kan sammenligne og utføre logiske operasjoner på variabler.
- ***Math operators* (oliven)**
  Her finner du de vanligste matematiske operasjonene på tall, og funksjonene map() og Constrain().
- ***Variables/Constants* (lilla)**
  Her gis variabler navn, type og verdi. Du kan velge mellom int, long, desimal og char. I tillgg kan du definere arrays. Selv om nenyen inneholder også noen konstanter (f.eks. 3.14) så kan den ennåikke definere konstanter slik overskriften synes å love.
- ***Communikasjon* (hvit)**
  Her finner man alle kommandoer som kommuniaserer med monitoren (seriell datalinje) og I^2C porten.
- ***Code blocks* (grå)**
  Her finnerdu separate setup(), loop() og header() blokker.

De øvrige blokkene er knyttet til spesielle komponenter fra ulike leverandører.

Programstruktur

Programstrukturen i Arduino er enkel og består normalt av en ***header()*** som henter inn biblioteker, definerer globale variable og konstanter. Dernest følger ***setup()*** funksjonen som utfører de kommandoene som bare skal kjøres en gang, og til slutt ***loop()*** funksjonen som inneholder alle de kommandoene



som skal kjøres mange ganger i loop. I tillegg kan man definere egne funksjoner (subrutiner) som kan kalles etter behov. På figuren over til høyre ser vi den vanligste strukturblokken sammen med resultatet når denne oversettes til C.

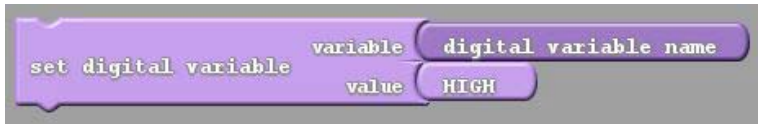
If() og while() er tilsvarende blokker som kontrollerer flyten i et program. En buet passform øverst til høyre på blokken indikerer at her skal det settes inn en betingelse, mens passformen under tilbyr innfylling av kommandoer.

Variabler og konstanter

Blokk-kategorien "Variables/Constants" inneholder blokker hvor man kan tilordne variabler navn, type og verdi. Blokken på figuren til høyre definerer og tilordner en verdi til en integer variabel. Under bildet av blokken er den tilsvarende C-koden. "int_var_1" er variabelens navn som defineres i subblokken øverst til høyre.


```
int int_var_1 = 0;
```

Setter man inn et enkelt veraibelnavn vil man imidlertid oppdage at dette utvides i oversettelsen til C.



Tilsvarende kan man definere og tilordne *digitale variabler* (boolan), *long integer*, *char* og *desimal* (float) som vist på figuren til høyre.

Videre kan en definere et array av variabler.

Vi legger spesielt merke til at hver type variabel har forskjellig passform og kan dermed bare brukes i forbindelse med blokker som tilbyr tilknytning av denne passformen.

Tester og betingelser

Blokk-kategorien "Tests" inneholder betingelser som f.eks. kan tilknyttes en if() blokk. I figuren til høyre ser vi eksempler på test-blokker.

Vi legger merke til at variablene som kan settes inn i test-blokkene har forskjellig passform og dermed er tilpasset ulike typer variabler. Figuren under viser eksempler på bruk av digitale variable, integer og character variable.

Disse betingelsene kan f.eks. brukes i forbindelse med en if()-blokk.

Egendefinerte funksjoner (subrutiner)

Fra blokk-kategorien "Control" henter man blokken "subrutine". Denne gir mulighet til å definere et innhold som kan utvides med de kommandoene man ønsker. I dette eksempelet er funksjonen kalt: "Funksjonens_navn".

Denne kan så kalles med en egen blokk som gis funksjonenens navn. Funksjonskallet kan godt komme før funksjonen defineres.

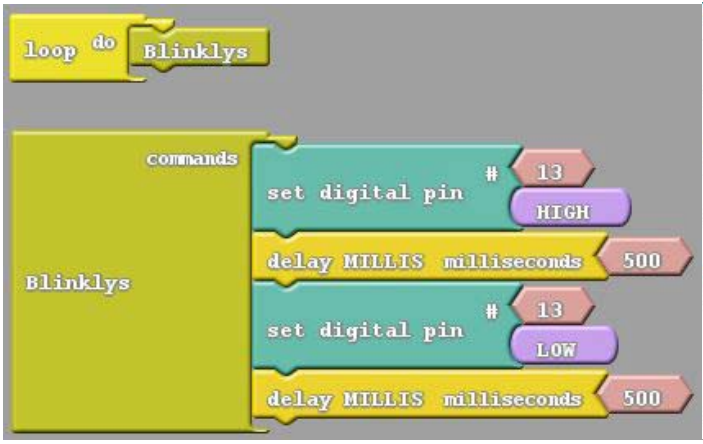
Eksempel 1: Blinklys

Figuren viser hvordan man ganske lett kan lage et program som får lysdioden på digital port nr. 13 til å blinke. C-koden som Ardublocks genererer er vist til høyre på figuren. Vi legger merke til at selv uten at vi har inkludert setup() i Ardublocks programmet, så genereres denne automatisk og inkluderer pinMode() som definerer at pin 13 er en utgang.


```
void setup()
{
  pinMode( 13 , OUTPUT);
}

void loop()
{
  digitalWrite( 13 , HIGH );
  delay( 1000 );
  digitalWrite( 13 , LOW );
  delay( 1000 );
}
```

Eksempel 2: Blinklys med funksjon

I dette eksempelet har vi valgt å flytte hele blinkfunksjoen til en egen funksjon (subrutine) som vi har kalt blinklys. Så langt har vi ikke sett at det er mulig å bringe en variabel over i kallet av funksjonen.




```
void setup()
{
  pinMode( 13 , OUTPUT);
}

void loop()
{
  Blinklys();
}

void Blinklys()
{
  digitalWrite( 13 , HIGH );
  delay( 500 );
  digitalWrite( 13 , LOW );
  delay( 500 );
}
```

7 Oppgavesamlinger

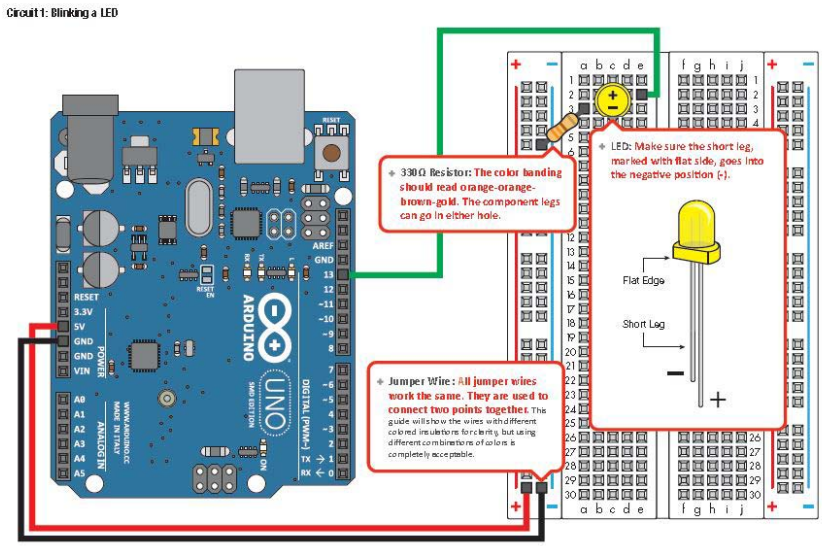
7.1 Introduksjonsoppgaver

Dette er oppgaver for den som er helt ny med hensyn til programmering og vil egne seg som introduksjon for både lærere og elever.

7.1.1 Blinkende lysdiode - Øving 1 A

Det skal bygges opp og programmeres en blinkende lysdiode.

Bygg følgende krets:



Skriv inn følgende programkode:

```
void setup()
{
   pinMode(13, OUTPUT);}


void loop()
{
   digitalWrite(13, HIGH);   // Turn on the LED
   delay(1000);              // Wait for one second
```

```
    digitalWrite(13, LOW);    // Turn off the LED
    delay(1000);              // Wait for one second
}
```

Last opp kretsen og test at den fungerer.

7.1.2 Morsesignalering - Øving 1 B, C, D og E

Du skal nå utvide funksjonen til øving 1A ved å lage en automatisk morsesender som sender SOS.

Spesifikasjon B: SOS

SOS som morse er: * * * - - - * * *

Prikklengde skal være: 200 msek

Streklengde skal være: 600 msek

Mellomrom mellom streker og prikker: 200 msek

Avstanden mellom bokstaver skal være: 600 msek

Avstanden mellom ord skal være: 1800 msek

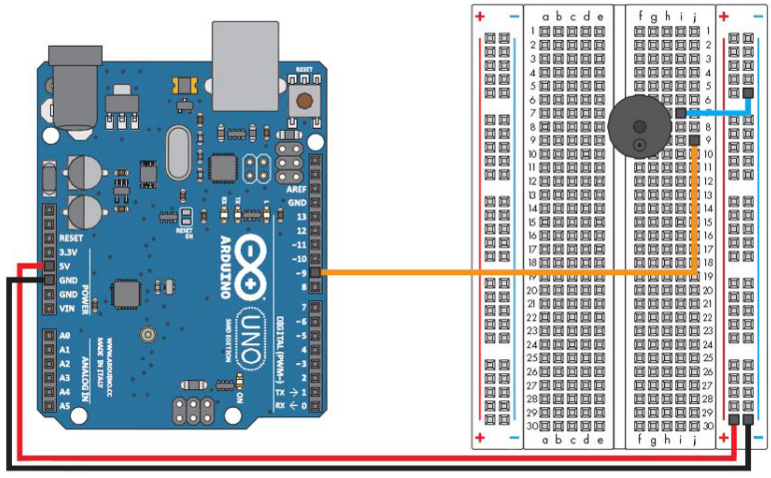
Spesifikasjon C: SOS med variabel hastighet

Det skal lages en SOS morsesender hvor det er lett å variere hastigheten

Spesifikasjon D: SOS med lys og lyd

Det skal lages en SOS morsesender som sender både lys og lyd.

Koble opp buzzer-en i tillegg til lysdioden



Spesifikasjon E: SOS med lys og lyd og bruk av funksjoner

Det skal lages en SOS morsesender som sender både lys og lyd, men koden skal forenkles ved at det lages en funksjon for bokstaven s og bokstaven o. Disse to funksjonene kalles av i loopen.

Løsningsforslagene finnes i vedlegg B.

7.1.3 To blinkende lysdioder

Koble opp to lysdioder en rød og en gul. Koble dem til hver sin utgang.

Spesifikasjon A:

Lag et program som er slik at lysdiodene blinker annen hver gang. Når den gule lyser er den røde slukket og omvendt.

Spesifikasjon B: Test øyets reaksjonsevne[6]

Halvåpen tilnærming:

Hvor raske diodeblink kan øyet ditt oppfatte?

For hvilke tidsverdier (grenseverdier) går lyset over til å se ut som sammenhengende lys?

Ser det ut for deg som om ulike lysdioder, av samme farge eller av ulik farge, gir samme grenseverdier?

Sammenlign dine resultater med andres resultater. Er de like? Hva forteller resultatene deg?

Styrt framgangsmåte:

Bruk bare den røde lysdioden. La av- og på-tiden være like lange. Øk blinkhastigheten til det punktet da lysdioden synes å lyse kontinuerlig. Noter grenseverdien.

Gjenta forsøket med den gule lysdioden og undersøk om grenseverdien for å se at den blinker, er forskjellig fra når du brukte rødt lys. Noter grenseverdien.

Hva kan du gjøre for å redusere usikkerheten i målingen?

Undersøk om grenseverdien varierer med:

- Kjønn
- Alder
- Farge
- Blink
- Forholdet mellom av- og påtid

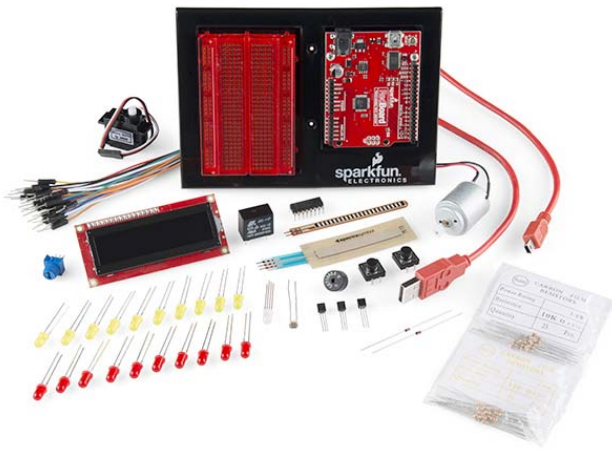
6. Etter en ide av Ingeborg Ranøyen

7.2 Nybegynner kit

I dette avsnittet skal vi beskrive to nybegynner kit.

7.2.1 "Sparkfun Inventors Kit V3"

Opplegget er basert på Sparkfun's nye *Inventors kit* V3 og anbefales brukt som fri eksperimentelle oppgaver som elevene kan prøve etter introduksjonen i avsnitt 7.1. Oppgavene egner seg også for å gi lærere grunnleggende erfaringer med bruk av ulike elektroniske komponenter som en forberedelse for å løse halvåpne oppgaver som omtalt i avsnitt 7.3. I dette avsnittet har vi valgt å henvise til guiden som følger med Inventors kit: SIK Guide, som er et rikt illustrert hefte som ved hjelp av eksempeloppgaver oppgave introduserer nye komponenter og tema.



I alt inneholder heftet viser byggebeskrive for 15 ulike byggeprosjekter fra de helt enkle til de mer omfattende. Programvaren til prosjektene kan lastes ned fra https://www.sparkfun.com/products/11576 ("SIK Code Library" - et stykke nes på sida).

Her følger en kortfattet liste over prosjektene:

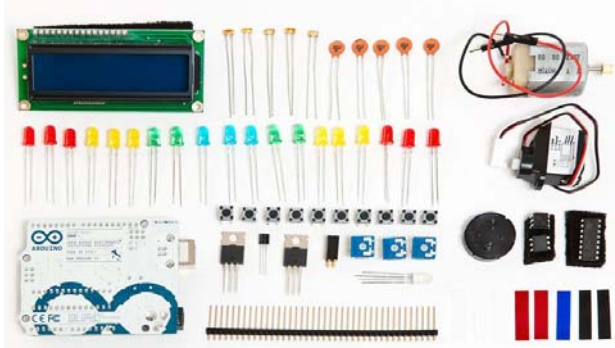
1. **"Blinkende LED"**
   *Oppgaven egner seg godt som første introduksjon og kan i stor grad differensieres. Her læres å definere enkle variabler, en port som in- eller output (**pinMode()**) og skriving til en digital port (**digitalWrite()**)*
2. ***"Potensiometer"** - Styr blinkhastigheten med et potensiometer*
   *Oppgaven egner seg til å lære **analog avlesning** (**analogRead()**) av en spenning samt variering av **forsinkelse** (**delay()**).*
3. ***"RGB LED"** - Styr fargene i en Rød-Grønn-Blå LED (RGB LED)*
   *Oppgaven egner seg for å lære fargeblanding og **pulsbreddemodulasjon** for styring av intensiteten til en lysdiode (**analog.Write()**). Dessuten omtales kontrollfunksjonene **for loop** og **if setning**.*

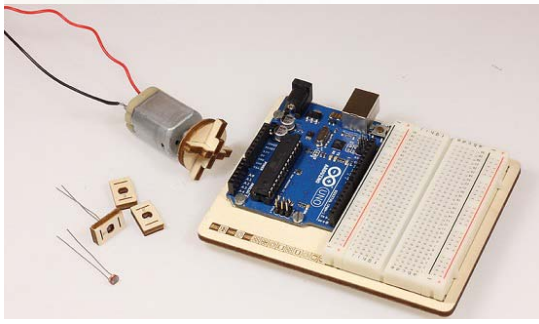
4. ***"Mange LED"*** *- Styr en rekke av 8 LED*
   Oppgaven egner seg for a lære bruk av ***for loop*** *og å sette opp* ***array[].***
5. ***"Trykk knapp"*** *- Bruk en trykknapp til å slå en LED av og på*
   *Oppgaven egner seg for oppkobling av knapper med bruk av motstand, samt å avlese en digital inngang (****digitalRead()****), og bruk av* ***if setninger.***
6. ***"Foto-motstand"*** *- Styr lyset i en LED ved hjelp av lyset i rommet*
   *Oppgaven introduserer spenningsdeleren, dvs. hvordan en varierende motstand kan omdannes til en varierende spenning. Dessuten anvendes funksjoner for å konvertere ett tallområde til et annet (****map()****) og en funksjon for avgrense et tallområde (****constrain()****).*
7. ***"Temperatursensor"*** *- Mål temperaturen og skriv ut verdien ut på PC-skjermen.*
   *Oppgaven egner seg for å lære hvordan lese av en analog spenning (****analogRead()****), hvordan sette opp en seriell kommunikasjonslinje tilbake til PC-en (****Serial.begin()****) og skrive resultatet til PC-skjermen (****Serial.print()*** *og* ***Serial.println()****).*
8. ***"En enkel servo"*** *- Styr vinkeldreiningen til en servo.*
   *Oppgaven egner seg for å lære hvordan inkludere biblioteker av funksjoner (****#include()****), definere en serverobjekt (****Servo*** *<servonavn>), knytte servoen til en spesiell utgang (<servonavn>****.attach()****) og styre servoen (<servonavn>****.write()****).*
9. ***"Bøyesensor"*** *- Bruk en bøyesensor (strekklapp) til å måle bøyningen og styre en servo.*
   *Oppgaven egner seg til å lese av en analog verdi (****analog.Read()****), avgrense tallområdet (****map()****) og styre en servo (<servonavn>****.write()****).*
10. ***"Trykkfølsomt potensiometer"*** *- Bruk et trykkfølsomt potensiometer til å styre RGB LED.*
    *Oppgaven egner seg til å lære å lese av en analog inngang (****analog.Read()****), konvertere et tallområde (****constrain()****) og styre nivået til lysdioder ved hjelp av pulsbreddemodulasjon (PWM)* ***(analog.Write())****.*
11. ***"Buzzer"*** *- Lag lyd og toner med en piezoelektrisk lydkilde*
    *Oppgaven egner seg godt for å lære å programmere lyd (****tone()****), dessuten læres å definere array som holder note- og toneverdier (<arraynavn>****[]****).*
12. ***"Roterende motor"*** *- Styr rotasjonshastigheten til en motor*
    *Oppgaven egner seg for å lære å koble opp og styre en motor ved help av kommandoer fra PC-konsollet. Dette krever at Arduino-en leser fra USB-inngangen (****Serial.available()*** *og* ***Serial.parseInt()****). Dessuten omtales bruk av* ***while setningen****.*
13. ***"Rele"*** *- Lære å bruke et rele for å styre større strømmer og spenninger*
    Oppgaven egner seg for å lære å koble opp og bruke et rele slik at det er mulig å styre utstyr som krever større strømmer (motorer, pumper, varmelementer o.l.)
14. ***"Skiftregister"*** *- Lære å styre et skiftregister og en rekke av lysdioder.*
    *Oppgaven egner seg godt til å lære hvordan en skal legge inn data i en integrert krets bruk av SPI-portene (SerialPeripheral Interface) og klokke- og datainngang (****shiftOut()****) og overføring av array av bit (****bitWrite()****).*

15. *"**LCD display"** - Lærer å skrive til et LCD display (16 x 2 karakterer)*
*Oppgaven egner seg godt dersom en ønsker å vise tallresultater i det produktet man ønsker å realisere. Her lærer man å inludere lcd biblioteket av funksjoner **(#include<LiquidCrystal.h>)** og å skrive tekst og variabler til displayet (**lcd.print()**).*

7.2.2 " The Arduino starter kit"

Også dette kitet beskriver 15 ulike prosjekter med tilhørende programvare. Til forskjell fra Sparkfun's kit så legger men har her valgt prosjekter som i seg selv burde ha større appell, samtidig som de trener bruk av elektroniske komponenter og programmeringskommandoer og funksjoner.





I tillegg til elektroniske komponenter inneholder settet også pappsjablonger som skal brukes på enkelte prosjekter. I stedet for plast benyttes treblater for å holde Arduino-kortet og koblingsbrettet sammen. Programvaren kan hentes ned fra følgende side: https://github.com/arduino/Arduino/tree/master/build/shared/examples/10.StarterKit, og boka som er meget god kan lastes ned fra:

Her følger en kortfattet liste over prosjektene:

1. **Bli kjentoppgave,** Tenn lysdiode med brytere
2. ***"Spaceship interface" –** Tenning og slukking av LED*
3. ***"Lovo-o-meter" –** Mål temperaturen i huden*
4. ***RGB-LED –** Fargeblanding med rødt, grønt og blått lys*
5. ***Servomotor –** Fraværsindikator, pekende pil*

6. ***Lys-teremin (LDR/Buzzer) –*** *Styrer lyd med lys*
7. ***"Keyboardinstrument" –*** *5 tone instrument*
8. ***Digitalt timeglass –*** *Lysdiodetimeglass*
9. ***Pinwheel (Motor) –*** *Styr hastigheten til en roterende skive*
10. ***Zoetrop (H-bridge) –*** *Styr motor retning og hastighet*
11. ***"Crystal ball" (LCD/tilt sensor) –*** *Rist og få en spådom*
12. ***Hemmelig lås (Mic/Servo) –*** *Bankerytme låser opp*
13. ***Touch-bryter –*** *Slå av og på lys ved berøring*
14. ***Kommunikasjon til PC –*** *Send melding fra Arduino til PC*
15. ***Optokobler –*** *Styr eksterne kretser*

Flere eksempler kan hentes fra: http://arduino.cc/en/Tutorial/HomePage

7.3 Halvåpne oppgaver

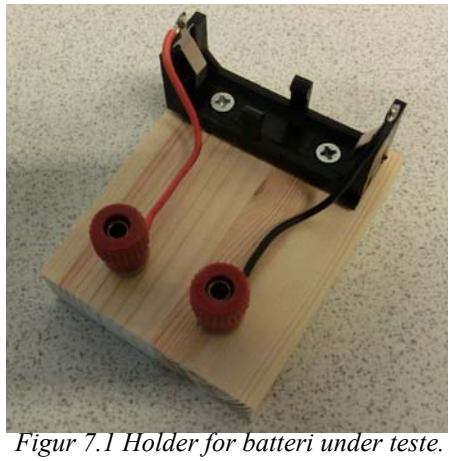
I dette kapittelet skal vi gi eksempler på ulike halvåpne oppgave, dvs. oppdraget er gitt som en funksjonspesifikasjon, slik at ordentlig oppdrag gjerne er gitt av en oppdragsgiver.

7.3.1 Halvåpen oppgave – Lag en batteritester

Spesifikasjon:

Det skal lages en batteritester for 1,5 V eller 2 x 1,5 V batterier.

- Studer databladet for batteriet og finn ut hvilke kriterier dere vil sette for et ubrukelig, et dårlig og et godt batteri.



Figur 7.1 Holder for batteri under teste.

Spesifikasjon 1 A

- Les av kalibrert spenning, og skriv den til monitoren i Arduino-monitoren

Spesifikasjon 1 B

- Skriv tekst til monitoren som forteller om batteriet er godt eller dårlig eller ubrukelig

Spesifikasjon 1 C

- Monter dioder på koblingsbrettet som viser om batteriet er godt (grønn), dårlig (gult) eller ubrukelig (rødt)

Spesifikasjon 1 D

- Tegn koblingsskjema i Fritzing

Utfordring

- Hvordan vil dere teste ut batteritesteren?

Utstyr:

- Batteriholder med batteri og tilkoblingsklemme
- Multimeter
- Sparkfun's Inventors kit med Arduino UNO

Løsningsforslag finnes i vedlegg C.1.

7.3.2 Halvåpen oppgave – Lag en automatisk blomstervanner

Spesifikasjon:

Det skal lages en automatisk blomstervanner.

Spesifikasjon 2 A - Vanning ved tørr jord:

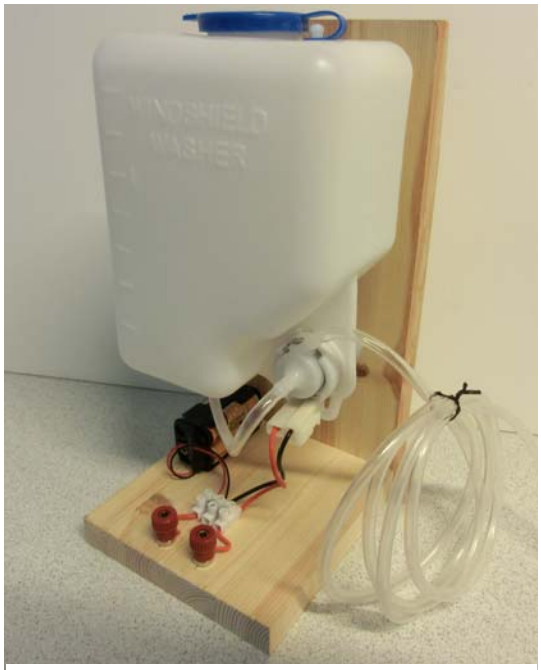
- Vanning skal skje kun når planten er tørr
- Vannmengden skal tilpasses plantens behov
- Hvordan vil du **teste** ut om denne fungerer som den skal?

Spesifikasjon 2 B – Sikkerhet mot vannsøl:

- Det skal legges inn ekstra **sikkerhet** mht. til søl. Dersom det kommer vann på bordet skal vanningen stoppe umiddelbart.
- Hvordan vil du **teste** ut om denne fungerer som den skal?

Spesifikasjon 2 C – Differensiert vanning

- Noen planter trenger lite vann ofte, andre mye vann sjelden.
- Vurdere fuktighetsgraden slik at litt fuktig gir 0,5 dl vann.
  Svært tørr gir 1 dl vann.



Figur 7.2 Vannbeholder med pumpe og tilkobling for styring av automatisk vanning.

- Hvordan vil du **teste** ut om denne fungerer som den skal?

Spesifikasjon 2 D – Tegn koblingsskjema

- Tegn koblingsskjema i Fritzing

Utstyr:

- Begerglass, vann
- Komplett sett med motstander (E12) (Clas Ohlson)
- Koblingsledning (ELFA)
- Vannbeholder med pumpe, batteri og slanger (Biltema)
- Multimeter (Clas Ohlson/ELFA)
- Sparkfun's Inventors kit med Arduino UNO (Sparkfun)

Løsningsforslag finnes i vedlegg C.2.

7.3.3 Halvåpen oppgave – Lag en automatisk vannvarmer

Vannet i et begerglass skal varmes opp til 30°C ved hjelp av et lite varmeelement.

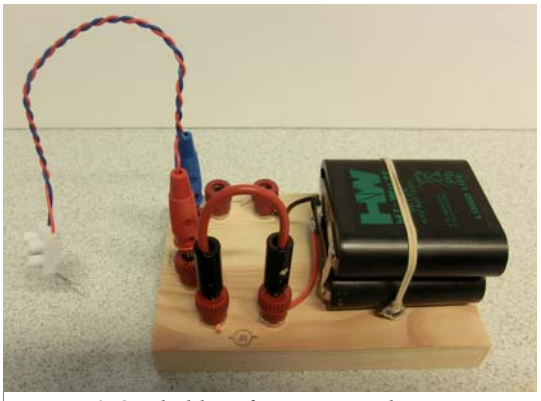
Spesifikasjon:

Spesifikasjon 3 A

- Lag en innretning som varmer opp og holder temperaturen konstant på 30°C.

Spesifikasjon 3 B

- … som gir fra seg en høy tone hver gang den går fra 29°C til 30°C.
- … som gir fra seg en dyp tone hver gang den går fra 30°C til 29°C.



Figur 7.3 Oppkobling for automatisk vannvarmer. Temperaturmåleren (NTC) er ikke med.

Spesifikasjon 3 C – Tegn koblingsskjema

- Tegn koblingsskjema i Fritzing

Utstyr:

- Begerglass, termometer, vann,
- Varmtvannskran/Vannkoker
- NTC motstand (50 kOhm)
- Varmeelement, batteri m/tilkobling

- Multimeter
- Sparkfun's Inventors kit med Arduino UNO

Løsningsforslag finnes i vedlegg C.3.

7.3.4 Halvåpen oppgave – Lag en elektronisk terning

Det skal lages en elektronisk terning ved hjelp av 7 lysdioder.

Spesifikasjon:

Spesifikasjon 4 A

- Med ett trykk på en bryter skal terningen gi en verdi fra 1 – 6.
- Verdiene skal angis på samme måte som øynene på en vanlig terning
- Hva er det minste antall I/O utganger som er nødvendig for å lage terningen?
- Hvordan vil dere teste kvaliteten til terningen?

Spesifikasjon 4 B

- Dere skal nå legge inn en bryter til som gir 50% større sjanse for å få en 6'er enn den "rettferdige" terningen.
- Lag mekanismen for å velge mellom de to terningene slik at den som kjenner trikset lett kan velge den "urettferdige" terningen uten at andre merker det.
- Undersøk om et av de andre lagene klarer å avsløre jukset?
- Hvordan vil dere teste at den oppfyller kravet?

Spesifikasjon 4 C – Tegn koblingsskjema

- Tegn koblingsskjema i Fritzing

Utstyr:

- Multimeter
- Sparkfun's Inventors kit med Arduino UNO

Løsningsforslag finnes i vedlegg C.4.

7.3.5 Halvåpen oppgave – Lag et trafikklys

Det skal lages et trafikklys etter følgende spesifikasjon:

Spesifikasjon

Spesifikasjon 5 A – Programmer et trafikklys:

- *Bestem rekkefølgen på lysene i et trafikklys når lyset skifter fra rød til grønt og fra grønt til rødt.*
- *Programmer trafikklyset slik at: Det er rødt i 5 sek, gult i 1 sek og grønt i 5 sek.*
- *Koble opp trafikklyset med:*
  Rød, Gul og Grønn lysdiode og skriv programmet

Spesifikasjon 5 B – Grønt lys på forespørsel:

- *Endre sekvensen i oppgave 1 slik at trafikklyset blir stående på rødt lys, helt til noen gir fotgjengerknappen ett kort trykk.*
- *Etter 5 sek. skal lyset skifte til grønt på riktig måte, være grønt i 5 sek., før det igjen skifter tilbake til rødt på riktig måte, hvor det blir stående til det kommer et nytt trykk på fotgjenger knappen.*
- ***Nyttige opplysninger:***
  - Bruk trykknappen som følger med settet

Spesifikasjon 5 C – Grønt lys på forespørsel med blinkende avslutning:

- *Endre programmet fra oppgave 5 B slik at det blir grønt lys i 8 sek hvorav de siste 5 skal være blinkende.*
  Hvert blink skal være 500 ms på og 500 ms av.
  Trafikklyset skal ellers oppføre seg som i oppgave 5 B.

Spesifikasjon 5 D – Blinkende grønt lys på forespørsel med lyd:

- *Hvert grønt lysblink skal etterfølges av et kort pip på 100 ms.*
- *Lyden skal lages i en egen funksjon som genererer lydsignalet med en lengde som bestemmes idet funksjonen kalles.*

Spesifikasjon 5 E – Blinkende gult lys etter mørkets frambrudd:

- *Når det blir mørkt skal lyset gå over i blinkende gult. Det skal da være 0,5 sek. på og 0,5 sek. av helt til det blir lyst igjen. Det skal være nok å legge hånda over den lysfølsomme motstanden for at det skal begynne å blinke.*
- ***Nyttige opplysninger:***
  - Koble opp den vedlagte fotomotstanden.
  - Bruk spenningsdeler.

Spesifikasjon 5 F – To lys som fungerer sammen:

- *Tenk dere en fotgjengerovergang. Når en trykker på den ene siden av gaten så skal begge lysene virke samtidig. Dette skal fungere uansett fra hvilken side bestillingen kommer fra.*

- *Kan dere løse denne oppgaven ved å gå sammen to og to grupper?*
- *Tenk dere at dere skal lage et lyskryss hvor det ene lyset er for fotgjengerovergangen og det andre for bilene.*
- *Hvordan vil dere nå løse oppgaven ved hjelp av to sett?*

Utstyr:

- Multimeter
- Arduino, pluss grønn lysdiode

7.3.6 Halvåpen oppgave - Lag et kolorimeter

Det skal lages et kolorimeter for å måle transmisjon av lys gjennom en oppløsning I et kolorimeter måles absorbsjonen av lys gjennom oppløsningen. Jo mindre lys som slipper gjennom, jo mer vannet er forurenset, jo høyere *absobans* (absobsjon av lyset gjennom oppløsningen). Som test kan benyttes vann tilsatt dråper med melk, selv om lyset i dette tilfellet spres mer enn det absorberes.

Som test skal instrumentet kunne bestemme rekkefølgen til 8 plastkrus med 1,5 dl vann som er "forurenset" med ulike antall dråper melk.

Spesifikasjon

Spesifikasjon 6.1A – Lag et kalorimeter:

Et kolorimeter består av en lyskilde og en lyssensor, montert i et mørkt kammer som hindrer lys fra utsiden å slippe inn. Kammeret må være så stort at det kan romme et plastkrus med plass til 1,5 – 2 dl vann.

- Konstruer kolorimeteret og tegn koblingskjema. I tillegg til komponenter fra Sparkfun Inventors kit, tillates brukt en lysdiode som leverer hvitt lys om ønskelig.

Spesifikasjon 6.1B – Program for å avlese lysstyrken

- Lag et program som leser av lysstyrken til lyset som slipper gjennom oppløsningen og skriv resultatet til Arduino-monitoren. Start gjerne med å tegn et flytdiagram for programmet.
- Bestem følsomheten til instrumentet, dvs. endring i måleverdi som funksjon av antall dråper melk i oppløsningen.

Spesifikasjon 6.1C – Les av verdien på LCD-displayet (bare for SIK V3)

- Koble opp kit'ets LCD-display og skriv den avleste verdien til displayet.

Spesifikasjon 6.1D – Kalibrer instrumentet og bestem antall dråper melk

- Finn en metode for å kalibrere instrumentet og lag en algoritme som gjør det mulig å skrive ut antall dråper melk i vannprøven.

Spesifikasjon 6.1E – Test kvaliteten til instrumentet

- Bruk instrumentet til å bestemme rekkefølgen til 8 prøver på 1,5 dl vann med ulik mengde melk blandet i. Prøvene skal stilles opp fra venstre mot høyre med økende mengde melk tilsatt.
- Bruk instrumentet til å anslå antall dråper tilsatt i hver prøve.
- Drøft feilkilder og avvik.

Spesifikasjon 6.1F – Bestem antall dråper i en ukjent prøve

- Anslå antall dråper i en ukjent oppløsning
- Drøft feilkilder og avvik

Et par tips:

Vurder å:

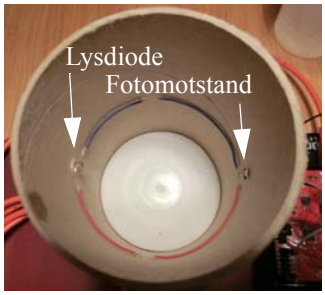
- ... midle måleresultatene over mange målinger.
  Hva vil fordelen med en slik metode være?
- ... slå av lyskilden mellom hver måling.
  Hva vil du kunne oppnå med en slik løsning?

En alternativ, mer åpen oppgave kan formuleres:

Spesifikasjon 6.2A – Lag et instrument for å bestemme forurensning

- Lag et instrument for å bestemme forurensningen i 8 vannprøver med 1,5 dl. Hver prøve er tilsatt et ukjent antall dråper melk.
- Hvordan vil du gå fram dersom du i tillegg skulle bestemme antallet dråper som er tilsatt?

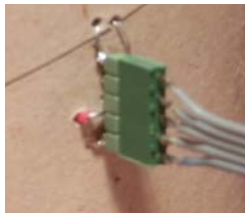
Kolorimeter for å bestemme forurensning (løsningsforslag)

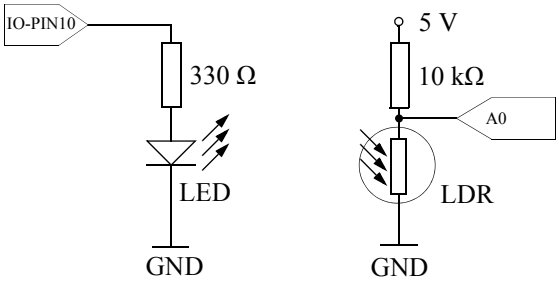


Analysekammeret kan lett lages av et papprør. I dette eksempelet er benyttet et embalasjerør med indre diameter 80 mm. Som "kyvette" er benyttet et plastkrus med høyde 104 mm og ytre diameter 78 mm ved øvre kant. Kruset er gjennomsiktig, og kan kjøpes på REMA 1000.

En hvit lysdiode og en fotomotstand (LDR) er plassert ca. 70 mm fra bunnen, diametralt over for hverandre i papprøret. Ledningene som sees på innsiden av kammeret, fører strøm til lysdioden fra en kontakt som er montert ved fotomotostanden (se bilde under til høyre).

Kontakten er en 5 pin header som er tilkoblet en stiftlist festet i pappen. Ledningene er loddet fast på understiden av stiftlista nærmest pappveggen i røret. Kammeret og monteringen av lysdioden og fotomotstanden kan gjøres enklere, men denne metoden er praktisk dersom man ønsker å benytte oppsettet gjentatte ganger i klasserommet.



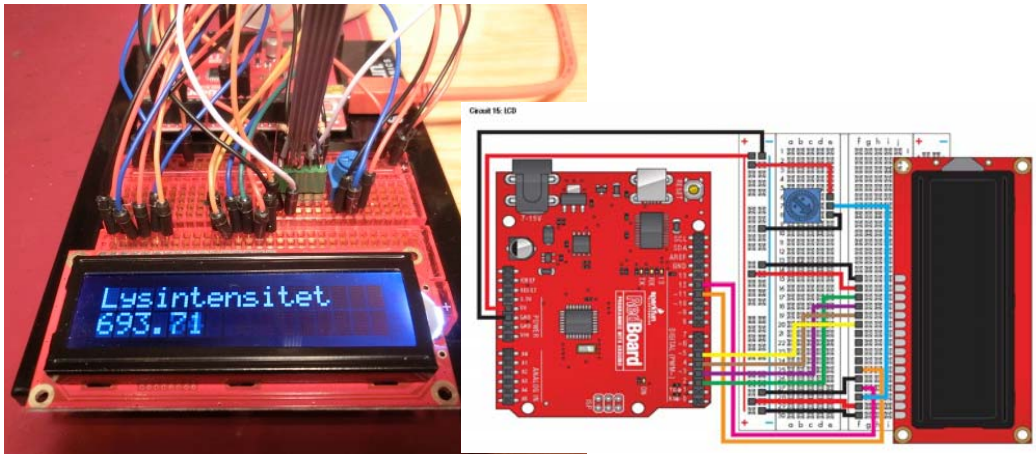


Koblingsskjema for LED'en og fotomotstanden er vist på figuren til venstre.

LED'en styres fra IO-pin 10 og LDR'en avleses av analog inngang A0.

I tillegg benyttes oppgave 15 i SIK V3 som beskriver oppkobling og bruk av LCD-display, som er praktisk når vi ønsker å vise verdien.

Bildet under viser oppkoblingen av LCD-displayet slik det er vist i SIK V3 (til høyre) og bilde av den virkelige oppkoblingen med de to seriemotstandene og header-kontakten.

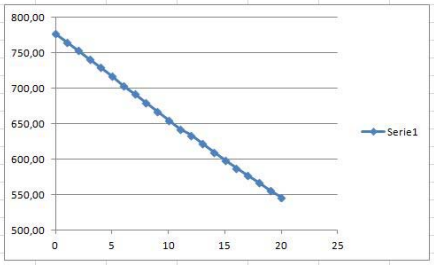


Målinger

Testmålingene er gjort under følgende betingelser:

- 1,7 dl vann uten bobler i plastkrus som rommer 2 dl
- 0 – 20 dråper melk tilført med en 9 ml pipette
- Plastkruset står på toppen av et standard 10 mm tykt plastlokk for bruk på embalasjerør
- Kammeret er tildekket med et lystett lokk (ikke hvitt plastlokk som er standard)
- Målinene er med pulset LED som tennes rett for målingene starter og slutter rett etter avsluttet måleserie

# Dråper	Lysintensitet
0 x_1	776,70 y_1
1	765,00
2	753,10
3	740,90
4	729,20
5	717,00
6	703,40
7	692,00
8	679,70
9	667,50
10	655,20
11	642,80
12	634,00
13	622,00
14	610,10
15	598,70
16	587,60
17	577,00
18	566,80
19	555,70
20 x_2	546,10 y_2



- Det gjøres 500 fortløpende målinger som midles

Som det framgår av figuren er sammenhengen nær lineær. Dog kan det synes som om det skjer noe mellom måling 11 og 12 der forskjellen i "lysintensitet" kun er ca. 8 og ikke ca. 12 som ellers."Lysintensiteten" i dette tilfellet er den AD-konverterte spenningen på analog inngangen A0, en verdi mellom 0 – 1023.

Matematisk modellering

Vi antar at det er en nær lineær sammenheng mellom antall dråper melk tilsatt vannet og den målte "lysintesiteten". Et uttrykk for en slik sammenheng kan lett settes opp ut fra topunktsformelen for en lineær funksjon:

Topunktsformelen for lineære ligninger kan også skrives slik:

$$y = (y_2 - y_1)/(x_2 - x_1) * (x - x_1) + y_1 \qquad (7.1)$$

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

$$y = k\,(x - x_1) + y_1 \qquad (7.2)$$

Setter vi verdier fra tabellen på figuren over inn i formelen, finner vi følgende sammenhenger:

$$y = (546{,}1 - 776{,}7)/(20 - 0)*(x - 0) + 776{,}7 \qquad (7.3)$$

$$\mathbf{y = -11{,}53\,x + 776{,}7} \qquad \mathbf{(7.4)}$$

Ligning angir forventet lysstyrke (y) som funksjon av antall (x) dråper melk. Stigningskoeffisienten –11,53 angir gjennomsnitlig endring i lysstyrke når vannet tilføres en ekstra dråpe med melk, og 776,7 angir lysstyrken ved rent vann (x = 0 dråper).

Programmet

Programlistingen finnes i sin helhet i vedlegg C.6, side 160.

Det er to ting å bemerke:

1. **Midling:**
   Erfaringer viser at enkeltmålinger kan bli svært ustabile. Dette skyldes sannsynligvis primært fluktuasjoner i væsken. Den enkleste måten å bøte på dette på, er å gjøre en lang rekke målinger for så å beregne middelverdien av måleserien. Vi har i vårt tilfelle midlet over 500 målinger. Selv ved så mange målinger er målingen unnagjort på godt under 1/10 sekund.
2. **Pulsing:**
   Tidligere varianter av lignende måleutstyr har vist at drift i LED'en, dvs. at lysstyrken endres over tid pga. temperaturendringer i halvlederen, kan være et problem. Det antas at dette problemet reduseres dersom LED'en er påslått kun i korte intervaller akkurat under målingen.

7.3.7 Turbidimeter

Til forskjell fra kolorimeteret måler *turbidimeteret* spredningen. Når f.eks. melk blandes med vann vil melka opptre i vannet som små partikler som ikke først og fremst absorberer lys, men som sprer det. Derfor vil det være interessant å måle hvor mye lys som spres ut til siden. Dette kan gjøres ved å montere en sensor f.eks. 90° på lysstrålen. En slik løsning gir muligheter til å sammenligne det transmitterte lyset (det som går rett gjennom) med det som spres og treffer sensoren som er montert 90° ut til siden. Det er viktig å finne en god måte å kalibrerer et slik instrument på.

7.3.8 Lag en stoppeklokke

Det skal lages en stoppeklokke, som kan brukes til å måle lange og korte tidsintervaller. En trykkbryter skal kunne brukes til å starte og stoppe klokka. Klokka kan også brukes som reaksjonstester. Utvikle prosjektet etter flere trinn.

Lag en klokke med følgende spesifikasjoner:

Spesifikasjon A - Lag en klokke som går

- Klokke skal starte på 0 sekunder hver gang programmet starter.
  Den skal vise sekunder og minutter med en nøyaktighet på +/- 0,5 sek
  Ta utgangspunkt i Eksempel øving 15 "LCD display"
- Lag et flytdiagram for programmet før det realiseres

Spesifikasjon B - Klokke med minutter, sekunder og 10-deler

- Lag klokka slik at den også viser 10-dels sekunder
  Ellers som i punkt A

Spesifikasjon C - Start klokka med en trykkbryter

- Bruk en trykkbryter og start klokka på 0 sekunder hver gang bryter trykkes inn og slippes. Ellers som i pkt B
- Tegn koblingsskjema for oppkoblingen av trykkbryteren
- Lag et flytdiagram for programmet før det realiseres

Spesifikasjon D - Stopp klokka ved andre gangs trykk på bryteren

- Bruk den samme bryteren til å stoppe klokka og holde siste verdi så lenge knappen holdes inne. Ved et nytt trykk skal klokka starte på nytt på null. Ellers som i C.
- Lag flytskjema for programmet før det realiseres

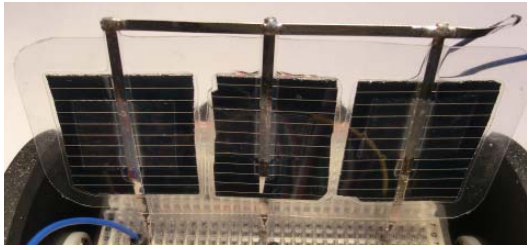
Spesifikasjon E - Bruk stoppklokka til å måle reaksjonstida

- Når startknappen trykkes går det en tilfeldig tid på noen sekunder en lysdiode tennes. Ellers som D.
- Idet dioden tennes skal knappen trykkes på nytt og vise reaksjonstiden.
- Drøft feilkilder for stoppeklokka

7.4 Oppgaver med Abot – en liten robot

Vi skal i dette avsnittet se på noen eksempler på oppgaver hvor vi benytter roboten Abot (se kapittel 4, hvor oppbyggingen av roboten er beskrevet). Videre anbefales avsnitt 5.5 som beskriver programmering av servomotorer.

7.4.1 Mr. Abot styrt med laserpeker



Ved hjelp av tre små solcellepaneler skal Mr. Abot styres gjennom en hinderløype. Ved å belyse det høyre panelet svinger Mr. Abot mot høyre, ved å belyse det venstre panelet svinger Mr. Abot mot venstre, og belyses det midterste panelet, skal den kjøre rett fram.

1. **Oppkobling:**
   Monter solcellepanlene på koblingsbrettet og koble dem til de analoge sensorinngangene *Analog_sensor_3*, *Analog_sensor_4*, *Analog_sensor_5*. Tegn koblingsskjema.

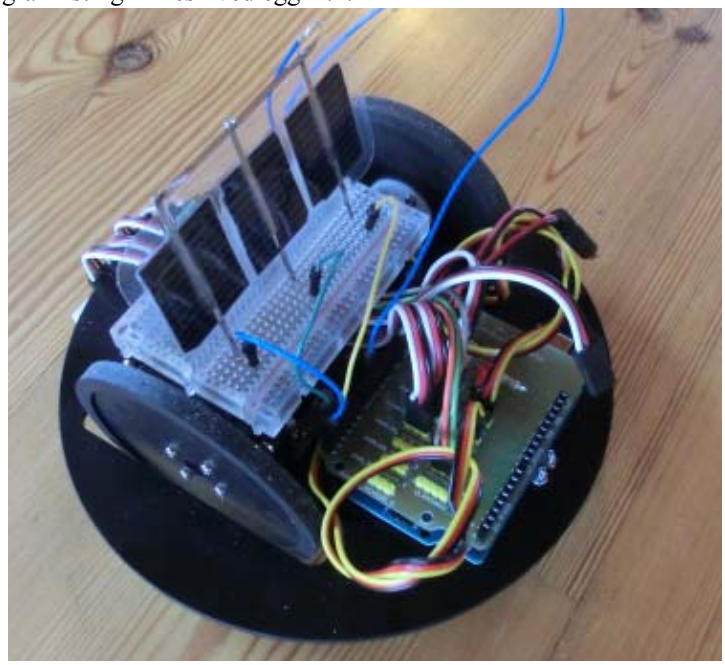
2. **Innledende test-program:**
   Skriv et program som leser av lysintensiteten på de tre panelene og skriver dem til monitoren. Det er tilstrekkelig med den relative lysintensiteten. Bestem den avleste verdien med og uten laserbelysning. Hva er marginen for med og uten laser?

3. **Kjøreprogram**
   Skriv koden slik at roboten ...
   ... stopper når den ikke belyses
   ... svinger til høyre når det høyre panelet belyses
   ... svinger til venstre når det venstre panelet belyses
   ... kjører rett fram når det midterste panelet belyses

4. **Automatisk terskeltilpasning**
   Bakgrunnsbelysning kan være et problem. Lag programmet slik at roboten måler bakgrunnsbelysningen idet programmet starter og tilpasser terskelnivået for hver av solcellene før oppstart.

I dette tilfellet er solcellepanelene laget ved å laminere tre nakne solceller på ca. 3 x 2,5 cm. Størrelsen er ukritisk,men bør være så store at det er lett å treffe med en laserpeker. For framstilling av solcellene se heftet: Praktisk solcelleteknologi for skolen. Heftet kan lastes ned fra: http://www.ntnu.no/skolelab/sl-bla-bokserie. Solceller kan også kjøpes fra KPT komet.

Forslag til programlisting finnes i vedlegg D.1.



7.4.2 Utforskende oppgaver

I disse oppgavene skal elevene karakterisere roboten. Slike oppgaver kan være en naturlig innledning for senere problemløsningsoppgaver.

Utforskende oppgave 1

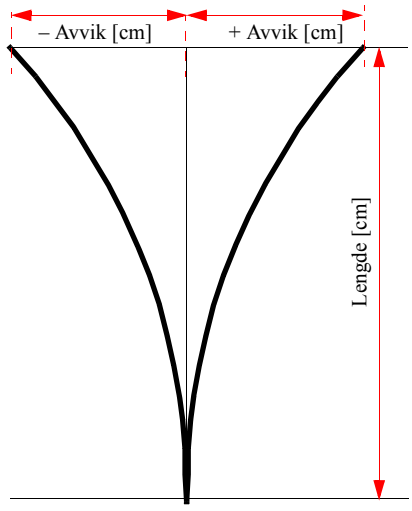
Rettlinjet bevegelse

*For at servomotorene skal bevege seg må den tilføres pulser. Databladet forteller at motoren står stille når pulslengden er 1500µsek., den går framover når pulslengden er større enn 1500µsek., og bakover når pulslengden er mindre enn 1500µsek. Dersom roboten skal bevege seg rettlinjet må den ene motoren settes til å gå framover og den andre til å gå bakover, siden motorene er plassert på hver sin side av roboten. Du vil ganske raskt også oppdage at motorene står stille for et intervall av pulslengder omkring 1500*µsek.

- *Bestem for hvilket pulsintervall hver av servoene står stille.*
- *Still inn farten for motorene slik at roboten beveger seg i rett linje.*
  Gjør 10 målinger og finn middelverdi og spredningen fra en rettlinjet bevegelse.
- ***Lag en funksjon** med hastighetsverdi som argument. Positiv hastighetsverdi er definert som framover og negativ hastighetsverdi som bakover.*

Venstre servomotors stoppområde:_____–_____[µsek] Høyre servomotors stoppområde:_____–_____[µsek]

#	Venstre pulslengde [µsek]	Høyre pulslengde [µsek]	Avvik [cm]
1			
2			
3			
4			
5			
6			
7			
8			
9			



Avvikets middelverdi _____[cm]

Utforskende oppgave 2

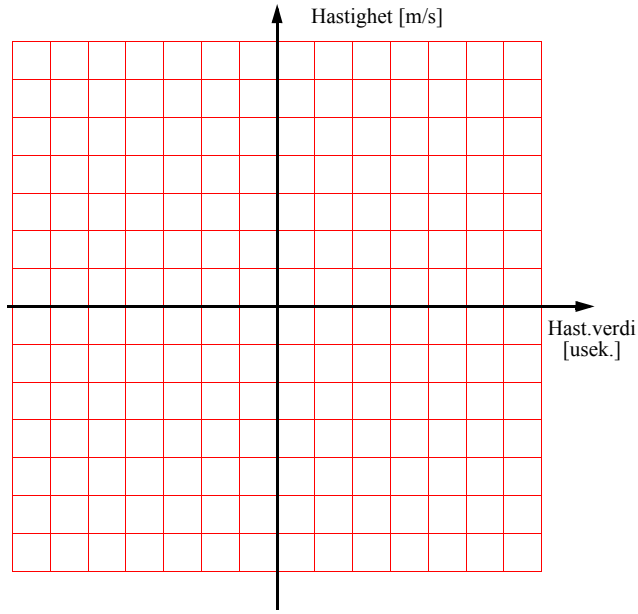
Hastighet

Fra oppgave 1 skal dere kjenne betingelsene for rettlinjet bevegelse, forover og bakover.

*Med **hastighetsverdi** menes pulslengden i µsek regnet fra kanten av stoppområdet. Definer verdien +1 som den hastighetsverdien (i µsek.) som akkurat får roboten til å bevege seg fram-over og –1 som den verdien (i µsek.) som akkurat får roboten til å bevege seg bakover.*

- *Finn en sammenheng mellom verdien i µsek. og hastigheten målt i m/s. Lag en tabell som viser sammenhengen mellom hastighetsverdien og hastigheten. Tegn deretter en graf som viser sammenhengen. Det skal måles for både positive og negative hastighetsverdier.*

#	Pulslengde [µsek]	Hastighet [m/s]
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		



- *Bestem funksjonsuttrykket for sammenhengen mellom hastighetsverdien og hastigheten i m/s.*
- *Endre funksjonen i oppgave 1 slik at argumentet til funksjonen kan oppgis i m/s.*

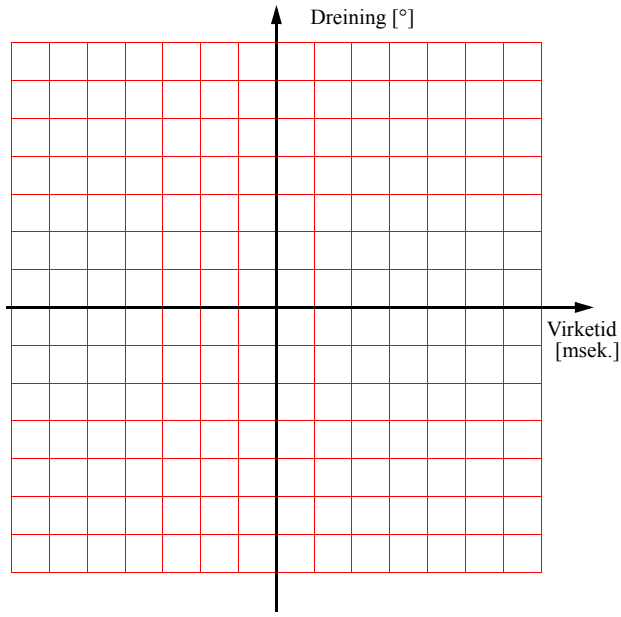
Utforskende oppgave 3

Dere har nå kontroll på robotens rettlinjede bevegelse og hastighet. Dere skal nå lage en funksjon som endrer robotens bevegelsesretning. Dette gjøres ved at høyre og venstre servomotor roterer i motsatt retning (egentlig i samme retning siden de er plassert speilvendt i forhold til hverandre). Vinkel bestemmes av hvor lenge vi lar dreiningen virke. Vi definerer at en positiv (+) vinkelverdi dreier roboten mot klokka, mens en negativ (–) vinkelverdi dreier med klokka.

- *Velg en passende fart for servomotorene og finn en sammenheng mellom vinkeldreiningen i grader og tiden dreiningen får lov til å virke.*

- *Lag en tabell over sammenhengen mellom vinkeldreiningen i grader og tiden dreiningen får virke.*

#	Virketiden [msek]	Dreining [°]
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		



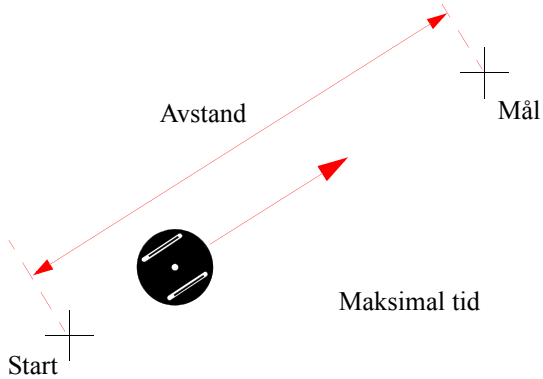
- *Lag en funksjon hvor argumentet er antall grader roboten skal dreie.*

7.4.3 Problemløsningsoppgaver

Abot drives fram av to servomotorer. Ved å endre hastigheten til motorene uavhengig av hverandre kan vi få roboten til å svinge.

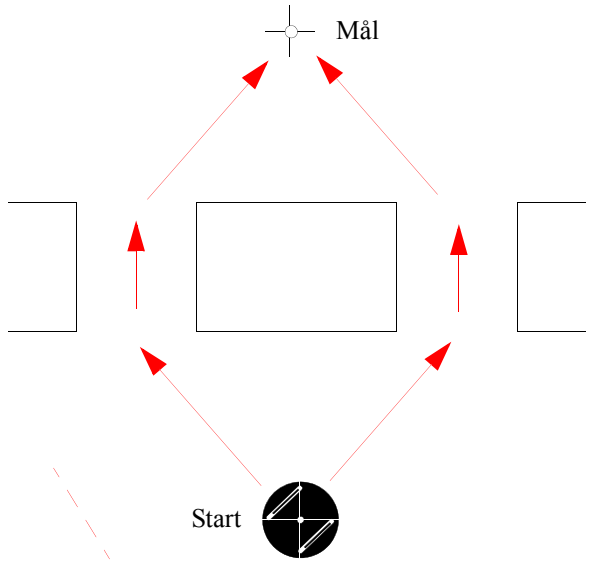
Problemløsningsoppgave 1:

Roboten skal bevege seg fra start til mål i løpet av en gitt tid. Det finnes ingen hindringer mellom start og mål. Ved målpunktet skal roboten stoppe. Avstanden mellom senterpunktet til roboten og målpunktet blir målt. Den har vunnet som kommer nærmest målpunktet innen tidsfristen. Avstanden mellom start og mål oppgis 10 min. for konkurransen starter. Bruk de neste 50 min. til å gjøre dere kjent med programmet og forberede dere til at den endelige avstanden blir oppgitt.



Måltallene (oppgitt etter 50 min.):

Avstand fra start til mål: 486 cm
Optimalt brukt tid: 20 sek.



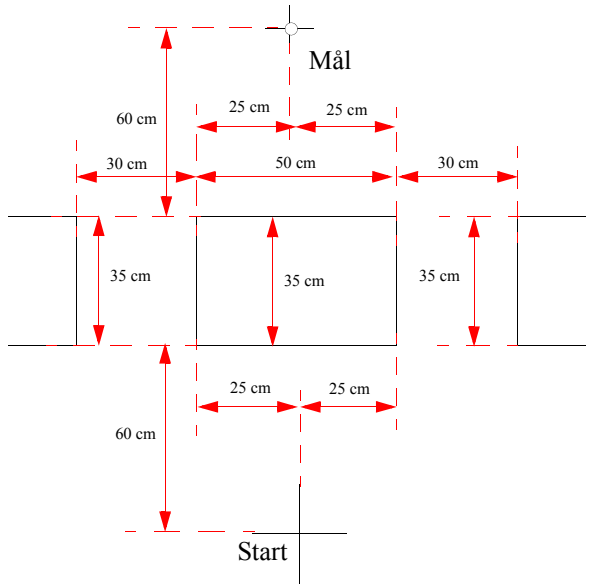
Problemløsningsoppgave 2

Forutsetningen for denne oppgaven er at roboten kun styres på tid uten sensorer. Dette stiller imidlertid store krav til robotens presisjon.

Abot skal bevege seg fra et startpunkt til et målpunkt. For å nå målet må robotene forsere en hindring. Formen på banen er vist i figuren til høyre, men ingen mål er foreløpig oppgitt. Dere har 50 min. på dere til å planlegge bevegelsen og gjør de nødvendige forberedelsen mht. programmeringen slik at dere i løpet av 10 min. kan ferdigstille programmet og være klar til å nå målet.

Måltallene (oppgitt etter 50 min.):

Figuren til høyre viser hvordan hindringene, startpunkt og målpunktet er posisjonert i forhold til hverandre. Dere har nå 10 min. til å programmere roboten slik at den kommer nærmest mulig målet.



Problemløsningsoppgave 3

Forutsetningen for denne oppgaven er at roboten styres ved hjelp av sensorer. Det forutsettes at roboten har en avstandssensor på høyre og venstre side. Det forutsettes at hindringene gir reflekser som er hensiktsmessige for styring.

Abot skal bevege seg fra et startpunkt til et målpunkt. For å nå målet må robotene forsere en hindring. Formen på banen er vist i figuren til høyre, men ingen mål er foreløpig oppgitt. Dere har 50 min. på dere til å planlegge bevegelsen og gjør de nødvendige forberedelsen mht. programmeringen slik at dere i løpet av 10 min. kan ferdigstille programmet og være klar til å nå målet.

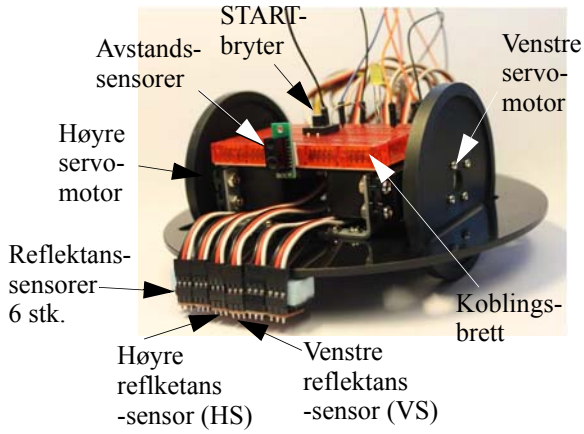
7.4.4 Mr. Abot følger en sort tape

I det neste oppdraget skal vi få Mr. Abot til å følge en sort tape klistret til et hvitt (eller lyst) bord eller plate. Til dette benyttes en eller flere *reflektans-sensorer* som måler reflektert lys fra underlaget.

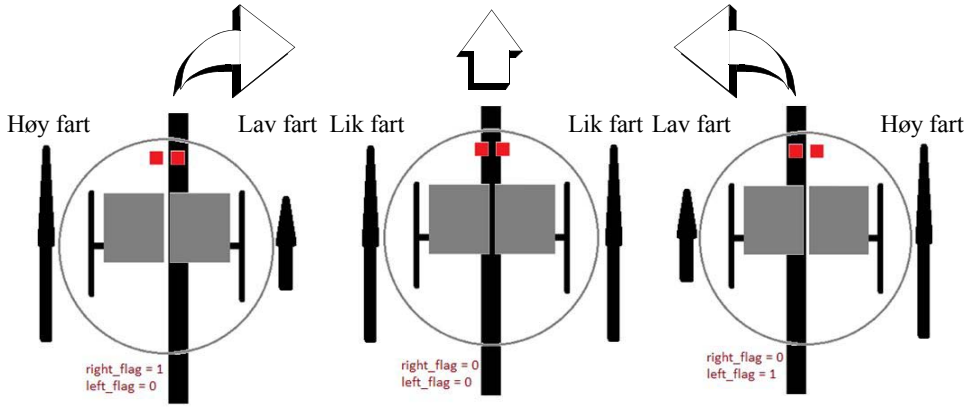
Oppbygging og styring av Mr. Abot

Figuren til høyre viser Mr. Abots viktigste deler:

Seks *reflektans-sensorer* er montert foran på roboten. Vi kan velge å bruke en, to eller fire av sensoren. I dette første eksempelet skal vi benytte de to i midten, høyre og venstre reflektanssensor (HS og VS). Roboten har også en *avstandssensor* for å oppdage hindringer. Programmet startes med et kort trykk på *START-bryteren s*om er plassert på koblingsbrettet. Programmet stoppes ved gi START-bryteren nok et kort trykk.



Når roboten skal kjøre rett fram går motorene samme vei[7] og like fort. Skal den svinge til venstre må høyre motor gå fortere enn venstre motor, og skal den svinge mot høyre så må venstre motor gå fortere enn høyre motor. Når vi skal kjøre bakover snur vi bare rotasjonsretningen til motorene.



Farten styres ved å sende pulser med forskjellig lengde til motorene. La oss studere strukturen i programmet før vi går videre med å styre motorene.

7. Egentlig går motorene motsatt vei når roboten kjører rett fram siden de er montert på høyre og venstre side av roboten.

Programstruktur

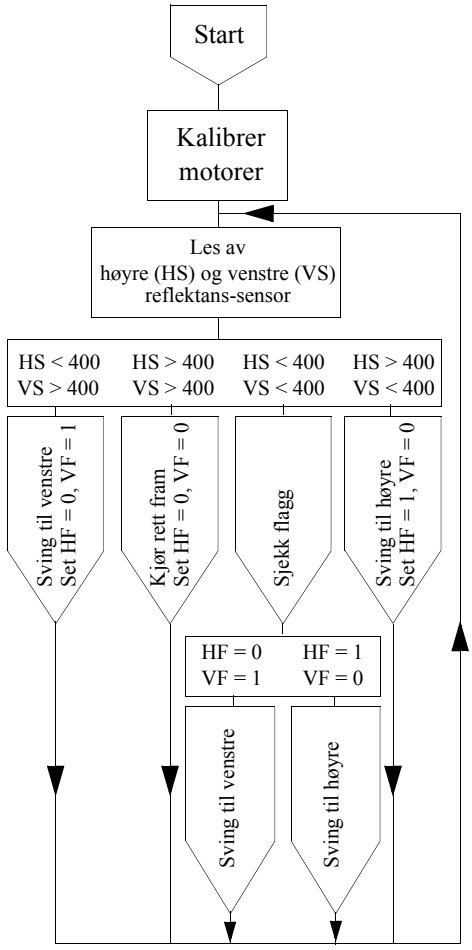
For å kunne følge den svarte tapen på bordet benytter vi i første omgang de to midterste reflektanssensorene. Disse gir en målt verdi fra 0 (helt lyst) til 1023 (helt mørkt). Programmet bruker de målte verdiene til å styre roboten.

Start: Programmet startes med å trykke på START-knappen som er montert på det hvite koblingsbrettet på roboten.

Kalibrering: Farten til motorene styres av tallverdier som legges inn i programmet. Disse er kalibret slik at de befinner seg mellom 0 til 100 (forover) og 0 til -100 (bakover). 0 verdien angir stopp, mens 100 angir full fart. Se egen *kalibreringsrutine*.

Les av reflektans-sensorer: Programmet leser av verdien fra høyre (HS) og venstre sensor (VS). Verdien er lav (< 400) når sensoren er over den lyse bordplata, og høy (> 400) når den er over tapen. To "flagg", høyre og venstre flagg, gis verdien 1 eller 0 avhengig om:

- Når begge sensorene "ser" tapen (HS>400, VS>400), kjører roboten rett fram. Sett venstre (VF) og høyre flagg (HF) til 0.
- Når høyre sensor "ser" bordet (HS<400) og venstre sensor ser tapen (HS>400), svinger roboten mot venstre. Sett HF til 0 og VF til 1.
- Når høyre sensor "ser" tapen (HS>400) og venstre sensor ser bordet (HS<400), svinger roboten mot høyre. Sett HF til 1 og VF til 0.
- Når begge sensorene "ser" bordet, har roboten enten havnet til høyre eller venstre for tapen. For å komme seg tilbake til tapen, må den vite hvilken sensor som sist "så" tapen. Til det bruker den flaggene:
  - Dersom den venstre sensoren sist "så" tapen (dvs. VF = 1), skal den svinge til venstre.
  - Dersom den høyre sensoren sist "så" tapen (dvs. HF = 1), skal den svinge til høyre.

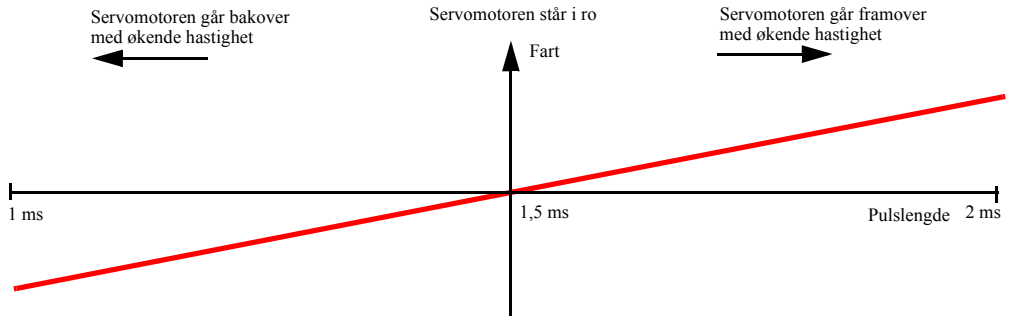


Beskrivelse av programmet

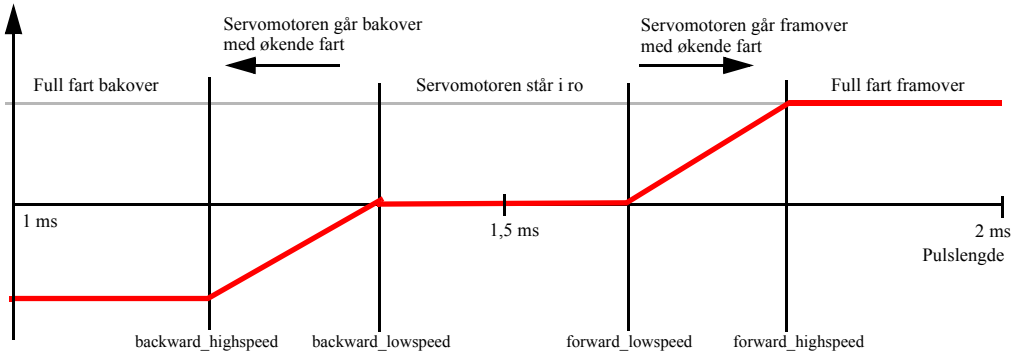
Vi skal i dette avsnittet beskrive i større detalj hvordan programmet fungerer. Vi tar da utgangspunkt i blokkdiagrammet på side 102. Vi antar at definisjon av variable og innhold av Setup-funksjonen er kjent.

Kalibrering av motorene

Hastigheten til en servomotor styres av et tog av pulser som sendes til motoren. Lengden av pulsene bestemmer farten. Pulslengden er fra ca. 1 ms (millisekund = 1000 µs) til 2 ms. I prinsippet skal motoren gå bakover med pulslengder i området 1 til 1,5 ms (1000 til 1500 µs), og forover i området 1,5 til 2 ms som vist på figuren under:

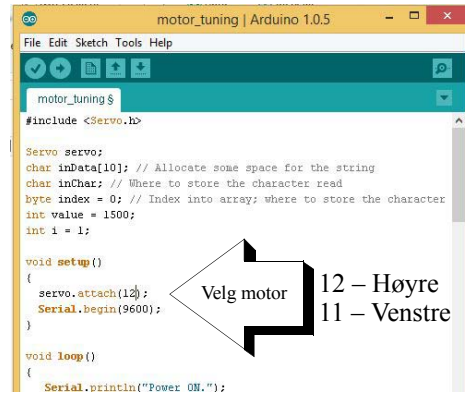


I virkeligheten er situasjonen en noe annen som vist på figuren under:



Bare i et snevert område kan farten styres nær lineært. Verdiene i *knekkpunktene* er heller ikke lik fra motor til motor. Vi må derfor finne disse fire grenseverdiene og bruke dem til å kalibrere programmet som styrer roboten:

backward_highspeed
backward_lowspeed
forward_lowspeed
forward_highspeed



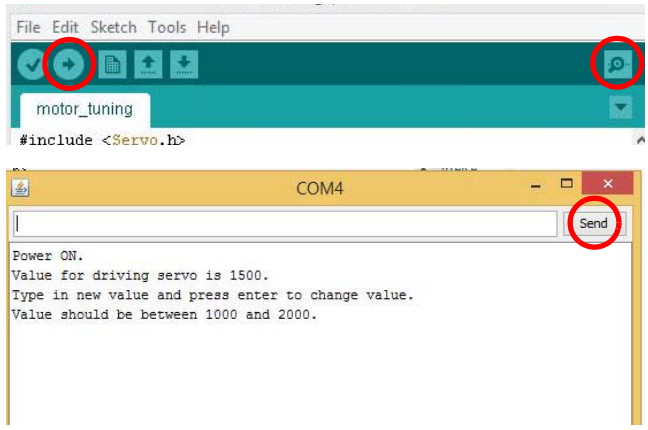
Slik bestemmes grenseverdiene:

1. Last opp programmet: *motor_tuning*
2. Velg motor (11 – Venstre, 12 – Høyre)

3. Overfør programmet til roboten

4. Åpne monitorvinduet

5. Skriv inn verdier fra 1000 til 2000 på kommandolinjen og velg **Send**.

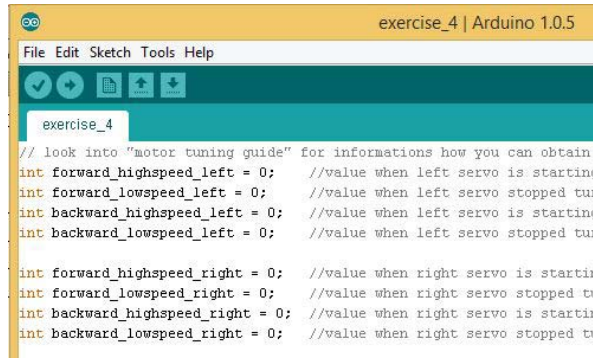


Punktet ***lowspeed*** er der motoren akkurat begynner å gå. Punktet ***highspeed*** er det punktet hvor maks. fart begynner å gå langsommere. For å finne omslagspunktet er det enklest å høre om lyden i motoren forandres fra toppfart og ned til dette punktet. Det er imidlertid ganske krevende å høre omslagspunktet.

For høyre motor (11):
Forward_lowspeed_right: _____
Forward_highspeed_right: _____
Backward_highspeed_right: _____
Backward_lowspeed_right: _____

For venstre motor (12):
Forward_lowspeed_left: _____
Forward_highspeed_left: _____
Backward_highspeed_left: _____
Backward_lowspeed_left: _____

Legg verdiene inn i programmet



6. Hent programmet exercise_4.: Skriv inn verdiene som er funnet på riktig sted (se figuren over til høyre).
7. Overfør programmet til roboten
8. Start programmet ved å trykke på: START-knappen på koblingsbrettet.
9. Sjekk at roboten følger den svarte streken på A3 arket.
10. Hva vil påvirke hastigheten til roboten langs streken?
11. Undersøk hva som skjer dersom den skal forsere:
    - En rett vinkel
    - En 270 graders skarp vinkel

12. Beskriv hvordan dere tror programmet fungerer.

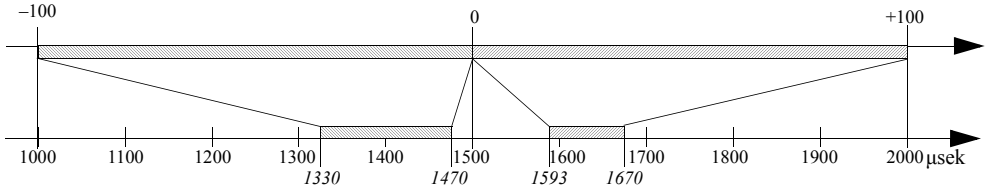
Normalisering

Vi har i forrige avsnitt bestemt pulslengdene for når de to motorene starter å gå forover og bakover, og vi har bestemt pulslengden der motorene når topphastighet, både forover og bakover. Da disse verdiene er forskjellige fra motor til motor, ønsker vi å standardisere disse slik at:

- Minimum forover for begge motorer er +1 og minimum bakover er –1
- Toppfart forover for begge motorer er +100 og toppfart bakover er –100

Programmet konverterer verdier i området 0 – 100 til det spesifikke området for vår motor. Vi har med andre ord foretatt en ***normalisering*** av verdiene. Normaliseringen gjøres i funksjonen: **void** set_speed_servo_left (**int speed**) som også sender pulsen som styrer den venstre servomotoren. En tilsvarende funksjon styrer den høyre servomotoren.

Figuren under illustrerer dette:



Uansett utgangspunkt vil vi kunne knytte en gitt verdi mellom 0 og 100 til en gitt hastighet, så fremt at topphastigheten for hver motor er den samme og vi er istand til å bestemme den, hvilket ikke alltid er så lett. For å utføre en slik konvertering benytter vi følgende map-funksjon:

```
var = map(speed, 1, 100, forward_lowspeed_left, forward_highspeed_left);
```

variablene `speed` inneholder ønsket fart i området 1 – 100, denne konverteres til en tilsvarende verdi i området `forward_lowspeed_left - forward_highspeed_left` som tillegges variabelen `var`. Det samme gjøres for negative hastigheter (bakover) og for høyre og venstre motor. Variabelen `var` settes så inn i kommandoen som sendes til servomotoren:

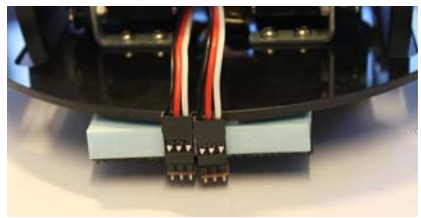
```
servo_left.writeMicroseconds(var);
```

Funksjonen som utfører denne operasjonen er i sin helhet vist i rammen under:

```
1   void set_speed_servo_left (int speed)
2   {
3       int var;
4       if (speed > 100) {speed = 100;}   // Sørg for at Speed alltid er mindre enn eller lik 100
5       if (speed < -100) {speed = -100;} // Sørg for at Speed alltid er større enn eller lik –100
6       if (speed < 0)       {var = map (speed, -1, -100, backward_lowspeed_left, backward_highspeed_left);}
7       else if (speed > 0) {var = map (speed, 1, 100, forward_lowspeed_left, forward_highspeed_left);}
8       servo_left.writeMicroseconds (var); // Sett servomotor til riktig fart
9       if (speed == 0) {servo_left.writeMicroseconds(1500);} // Dersom speed er lik 0, stopp servomotor
10  }
```

På linje 4 og 5 sørger programmet for at ***speed*** befinner seg i området fra – 100 til +100. Linje 6 og 7 utfører konverteringen fra området – 100 til +100 til det spesifikke området for den aktuelle motoren (se figuren over). I linje 8 sendes den konverterte pulslengden til servomotoren. Dersom speed = 0, stoppes motoren ved å sende pulslengden 1500 µsek.

Avlesning av reflektans-sensorene



To reflektans-sensorer er plassert under foran på roboten og måler hvor mye lys som reflekteres fra underlaget. Mørkt underlag reflekterer lite lys, lyst underlag reflekterer mye lys. Dette er grunnlaget får å kunne følge kanten av en svart stripe på et lyst underlag.

Selve avlesningen av sensoren utføres av koden:

```
1     analog_sensor_L_read = analogRead(analog_sensor_L);
2     analog_sensor_R_read = analogRead(analog_sensor_R);
```

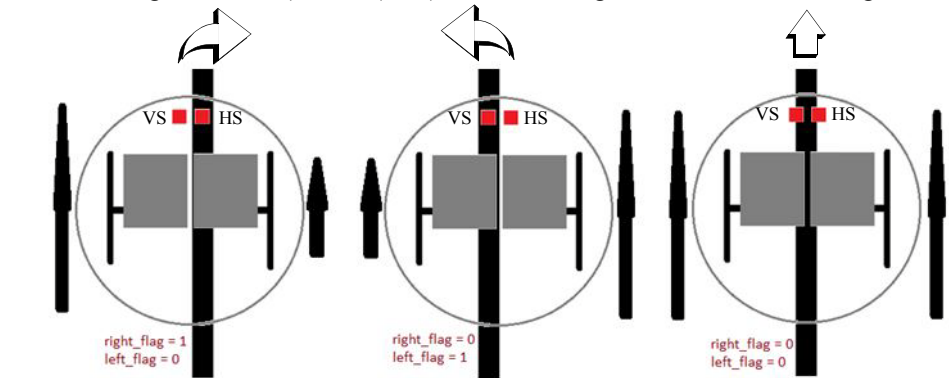
Funksjonen `analogRead(analog_sensor_L)` leser verdien fra den analoge inngangen `analog_sensor_L` eller også *A1* hvor den vestre reflektans-sensoren er tilkoblet. Den målte verdien tilordnes veraibelen `analog_sensor_L_read`. Tilsvarende for den høyre som er koblet til *A0*.

Følg den svarte stripen

Hvitt underlag gir verdier under 50, mens sort underlag gir verdier på over 600. Å sette terskelverdien til 400 kan være fornuftig. Ved å sammenligne de målte verdiene for de to sensorene kan vi bestemme om roboten befinner seg:

- ... langs venstre kant av stripen (VS< 400 og HS > 400)[8] –> sving til høyre
- ... langs høyre kant av stripen (VS> 400 og HS < 400) –> sving til venstre
- ... på stripen (VS > 400 og HS > 400) –> kjør rett fram
- ... utenfor stripen (VS < 400 og HS < 400)

Dette er vist i figuren under (de små (røde) kvadratene angir de to sensorene, VS og HS.



8. HS - Høyre Sensor, VS - Venstre Sensor

Vi ønsker at begge sensorene skal være inne på stripen. Dersom en av dem er utenfor stripen må vi sørge for at roboten svinger inn mot stripen.

Dersom venstre og høyre sensor begge viser verdier over 400, så er begge på streken og roboten skal bevege seg rett fram. Dvs. farten til begge motorene settes til 80.

```
1   if(analog_sensor_L_read > 400 && analog_sensor_R_read > 400)
2      {
3        set_speed_servo_left(80);
4        set_speed_servo_right(80);
5        left_flag = 0;
6        right_flag = 0;
7      }
```

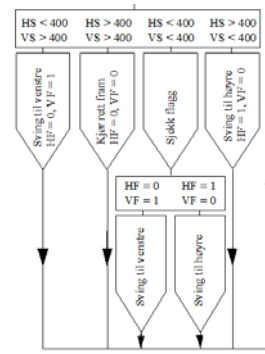
Dersom venstre sensor viser < 400, dvs. er utenfor streken og høyre sensor viser en verdi > 400, dvs. er på streken, så vil roboten befinne seg litt til venstre for streken og må svinge litt mot høyre. Dvs. venstre hjul (70) må gå litt fortere enn høyre hjul (30).

```
1   if(analog_sensor_L_read < 400 && analog_sensor_R_read > 400)
2      {
3        set_speed_servo_left(70);
4        set_speed_servo_right(30);
5        left_flag = 0;
6        right_flag = 1;
7      }
```

Dersom venstre sensor viser > 400, dvs. er innenfor streken og høyre sensor viser en verdi < 400, dvs. er på utsiden av streken, så vil roboten befinne seg litt til høyre for streken og må svinge litt mot venstre. Dvs. høyre hjul må gå litt fortere (70) enn venstre hjul (30).

```
1   if(analog_sensor_L_read > 400 && analog_sensor_R_read < 400)
2      {
3        set_speed_servo_left(30);
4        set_speed_servo_right(70);
5        left_flag = 1;
6        right_flag = 0;
7      }
```

Men hva skal vi gjøre dersom begge sensorene viser at de er utenfor den svarte stripen? I stå fall må vi vite på hvilken side av stripen roboten befinner seg. Dette kan vi vite dersom vi merker oss hvilke av de to sensorene som siste var i berøring med stripen.



Dersom den høyre sensoren sist var i berøring med stripen, settes `right_flag = 1;` og vi vet at roboten befinner seg til venstre for stripen. Dersom den forlater stripen helt, må den svinge til høyre for å gjenfinne stripen.

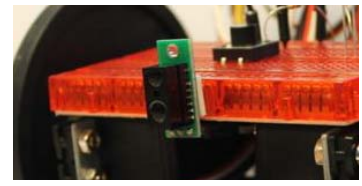
Dersom den venstre sensoren sist var i berøring med stripen settes `left_flag = 1;` og vi vet at roboten befinner seg til høyre for stripen. Dersom den forlater stripen helt, må den svinge til venstre for å gjenfinne stripen.

I programmet er dette uttrykt slik:

```
1  if(analog_sensor_L_read < 400 && analog_sensor_R_read < 400)
2      {
3      //if there is no line under sensors, use flags for moving
4        if(left_flag == 1)
5        {
6          set_speed_servo_left(-10);
7          set_speed_servo_right(70);
8        }
9        //if left flag is set, turn hard right
10       if(right_flag == 1)
11       {
12         set_speed_servo_left(70);
13         set_speed_servo_right(-10);
14       }
15       //if right flag is set, turn hard left
16     }
```

Vi legger merke til at for å gjøre svingen ekstra krapp, så lar man de to hjulene bevege seg motsatt vei (linje 6 og 7, og linje 12 og 13).

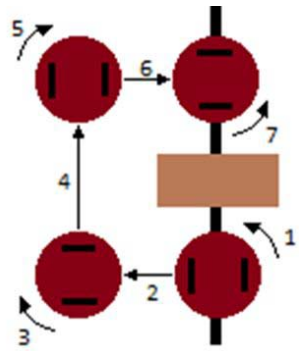
Unngå hindring



Vi skal nå inkludere en funksjon i programmet som gjør at roboten kan omgå en hindring. Til dette bruker vi avstandssensoren som er montert foran på roboten. Når en hindring kommer nærmere enn ca. 10 cm vil sensoren gi "beskjed".

Sensoren betegnes `digital_sensor` i programmet og gir lav spenning (en 0'er) på signalutgangen når en hindring detekteres. I utgangspunktet antar vi at hindringen har kjente dimensjoner slik at vi kan lage programmet slik at den kommer rundt og finner igjen den svarte stripen på baksiden.

Unnamanøveren er vist på figuren til høyre.



Når hindringen detekteres svinger roboten 90° til venstre, kjører tilstrekkelig langt mot venstre og tar en 90° sving til høyre, passerer hindringen svinger til høyre igjen for å komme tilbake til den svarte streken.

For at roboten skal kunne utføre en så krapp sving som mulig, kjøres motorene på full fart i motsatt retning. Da vil roboten svinge omkring midtpunktet sitt.

Programmet sjekker avstands-sensoren en gang for hver runde i loopen. Dette gjøres med kommandoen:

```
if (digitalRead(digital_sensor) == 0) avoid_obstacle();
```

Dersom avstandssensoren (`digital_sensor`) gir 0, kalles funksjonen `avoid_obstacle ();` og roboten kjører utenom hindringen.

Funksjonen `avoid_obstacle ();` for unnamanøvrering er vist i rammen under.

```
1  void avoid_obstacle(void)
2  {
3    set_speed_servo_left(-100);
4    set_speed_servo_right(100);
5    delay(500); // Sving til venstre
6    set_speed_servo_left(100);
7    set_speed_servo_right(100);
8    delay(1000); // Kjør rett fram
9    set_speed_servo_left(100);
10   set_speed_servo_right(-100);
11   delay(500); // Sving til høyre
12   set_speed_servo_left(100);
13   set_speed_servo_right(100);
14   delay(1000); // Kjør rett fram
15   set_speed_servo_left(100);
16   set_speed_servo_right(-100);
17   delay(500); //Sving til høyre
18   set_speed_servo_left(100);
19   set_speed_servo_right(100);
20   delay(1000); // Kjør rett fram
21   set_speed_servo_left(-100);
22   set_speed_servo_right(100);
23   delay(500); // Sving til venstre
24 }
```

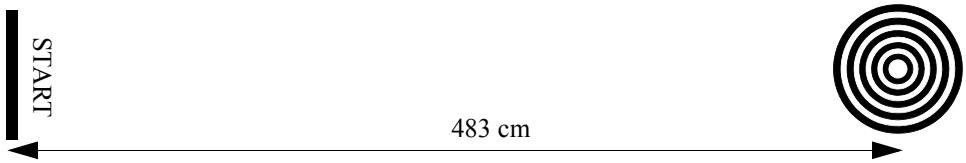
Størrelsen på `delay(<msek>;` bestemmer hvor mye den skal svinge og hvor langt roboten skal kjøre.

7.4.5 Banen

Banene lages på en A0 hvit papplate. Følgestripen lages med sort tape.

Oppgave 1A

Banen markeres på gulvet med en tape som start punkt. Målet angis med en blink laget på en pap-plate som tapes fast i gulvet rett før konkurransen.



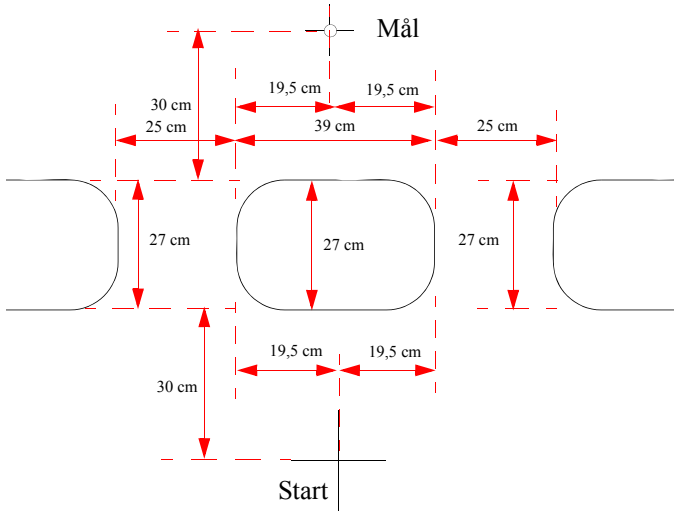
Oppgave1B

Oppgaven kan gjøres mer krevende ved at målet skal nås på en idealtid som oppgis. F.eks. 16 sek på nevnte avstand. Bedømmingen kan være slik at hvert avvik i sekunder gir et avvik i 3 cm. Dvs. at dersom roboten ender opp 8 cm fra målet etter 18 sekunder kjøring, så blir resultatet 8 cm + 2 x 3 cm = 14 cm.

Oppgave 2

Oppgaven kan utformes på ulike måter, men hovedpoenget er at deltagerne skal programmere roboten slik at den beveger seg fra et startpunkt til et mål skjult bak en hindring. Målene kan enten være oppgitt på forhånd eller ukjente helt til kort tid før konkurransen.

27 (l) x 39 (b) x 19 (h) cm



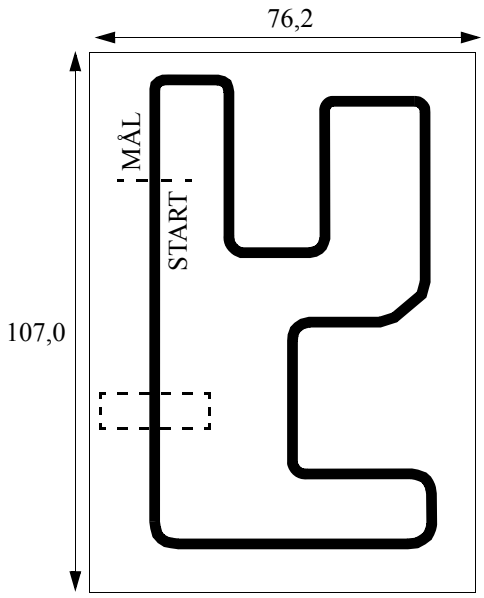
Vi velger å benytte plastbokser bra Biltema som hinder.

Oppgave 3A

Oppgaven går ut på å følge en svart stripe rundt en bane på kortest mulig tid. Figuren under viser banen som er tapet opp på en papplate av størrelse 76,2 x 107,0 cm.

Oppgave 3B

En hindring er satt opp i banen, denne skal forskeres. Konkurransen kan fortsatt gå på tid. Deltagerne vet ikke nøyaktig hvor hinderet befinner seg og må derfor detektere det når det oppdages.



8 Referanser

[1] Rossing, Grande, Gansmoe, Nielsen, *Miljø- og Romteknologi (CanSat) m/Sensorteknologi*, Rev. 3.10, Skolelaboratoriet ved NTNU, Mai 2013

[2] Datablad ATmega168A
http://www.atmel.com/dyn/resources/prod_documents/doc2545.pdf

[3] Atmel, ABOT chassis assambly guide, ABOT-0004, 02/13

Vedlegg A Organisering av undervisning

Dette avsnittet beskriver ganske kort hvordan programmering og bruk av Arduino kan introduseres i klassen, ev. som en første introduksjon på et lærerkurs.

A.1 Introduksjon for lærere

A.1.1 Omfang

To timer.

A.1.2 Målgruppe

Dette er et eksempel på hvordan en kan gi en kort introduksjon av Arduino for en gruppe lærere, for bruk i valgfaget Teknologi i Praksis i ungdomsskolen eller faget Teknologi og Forskningslære i videregående skole. Det kan også brukes av lærere som en introduksjon for sine kollegaer på egen skole.

A.1.3 Målsetning

Hensikten med opplegget er å gi lærerne en kort introduksjon og et inntrykk av hvordan undervisningsopplegget kan brukes i de aktuelle fagene:

- Peke på aktuelle kompetansemål
- Gi en kort introduksjon av hva mikrokontrollere er og hvordan de brukes (avsnitt 2.1, side 15)
- Gi "hands on" erfaring med å bygge og programmere en enkel krets, se avsnitt 7.1 på side 79.
- Vise eksempler på enkle prosjekter og hvordan disse kan oppfylle kompetansemålene i den aktuelle læreplanen, se f.eks. avsnitt 7.3 på side 85.

A.1.4 Utstyr

Følgende bør finnes tilgjengelig for gjennomføring for n antall lærere:

- 1/2 n stk. Sparkfun's Inventor's kit slik at to og to lærere kan arbeide sammen
- 1/2 n stk. PC-er med programvare og drivere installert
- 1 stk Multimeter for test
- 1/2 n stk Oppgaveark med løsninger
- n stk Ressurshefter
- 1 stk. PowerPoint presentasjon m/videokanon

A.1.5 Forslag til gjennomføring

Tabellen viser forslag til hvordan en kan legge opp en kort presentasjon av bruk av mikrokontrollere for en gruppe lærere.

Tabell 1: Forslag til gjennomføring av en kort introduksjon

Omfang	Innhold	Referanse
10 min	Introduksjon, hensikten med opplegget, eksempler på aktuelle kompetansemål.	Avsnitt A.1.3, side 114
20 min	Hva er mikrokontrollere og hva brukes de til? - Hva er en mikrokontroller og hvor finner vi dem? - Hva er Arduino? - Hva er et program?	Avsnitt 2.1, side 15
30 min	"Hands on" erfaring - introduksjon - Hva finnes av komponenter i settet? - Bygg opp den første kretsen (blinklys, trafikklys) - Bruk av programeditoren, skriv inn programmet - Test ut	Avsnitt 7.1, side 79 eller avsnitt 7.3.5, side 88
30 min	Løs en enkel oppgave - Lag et trafikklys som lyser i sekvens ev. - Lag en morsesender	
20 min	Muligheter - Vise muligheter mht. halvåpne oppgaver - Ev. invitere til lærerkurs.	Avsnitt 7.2.1, side 82, avsnitt 7.3, side 85, avsnitt 7.4, side 95.
10 min	Diskusjon	

A.2 Introduksjon av Mr. Abot for studenter ved NTNU

A.2.1 Omfang

Tre timer.

A.2.2 Målgruppe

Dette er et eksempel på hvordan en kan gi en kort aktivitet for studenter. Opplegget skal første gang holdes i begynnelsen oktober 2014 for ca. 3 x 30 studenter som deltar i Vektorprogrammet ved NTNU. Man kan ikke regne med at de kan programmering, selv om mange sikkert kan noe. De deles inn i 10 grupper av 3 elever.

A.2.3 Målsetning

Hensikten med opplegget er å gi studentene en positiv opplevelse:

- Gi en kort introduksjon av hva mikrokontrollere er og hvordan de kan programmere den til sitt formål. Det er ikke meningen at de skal bruke mye tid på å lære generell programmering, men likevel få litt innsyn i hva det går ut på
- Gi "hands on" erfaring med å programmere en robot til å løse en spesiell oppgave, .
- La studentene bruke mest mulig tid til utforsking og minst mulig til fellesundervisning.
- Ha felles konkurranser for å teste ut hva de har forstått

A.2.4 Utstyr

Følgende bør finnes tilgjengelig for gjennomføringen for n grupper:

- n stk. Mr. Abot m/kabel
- n stk. PC-er med programvare og drivere installert pr. gruppe
- n stk Multimeter for test
- n stk Oppgaveark (ev. med løsninger)
- n stk Ressurshefter
- n stk Testark for å følge en stripe
- 1 stk. PowerPoint presentasjon m/videokanon
- 3 stk Arena for uttesting (langs rett linje)
- 3 stk Arena for følging av strek

A.2.5 Forslag til gjennomføring

Tabellen viser forslag til hvordan en legge opp aktiviteten for studenter.

Tabell 2: Forslag til gjennomføring av aktiviteten for studenter

Omfang	Innhold	Referanse
(Tot: 180 min)	**Velkommen** Hensikten med opplegget, hva går dagens oppdrag ut på.	

Tabell 2: Forslag til gjennomføring av aktiviteten for studenter

Omfang	Innhold	Referanse
30 min (Akk: 30 min)	**Introduksjon til mikrokontrollere?** - Installasjon? - Arduino - Samspill mellom program og omverdenen, eksempler. - Styring ved hjelp av inn- og utganger? - IDE - programgrensnittet? - Hva er et program? - Typisk programstruktur?	
30 min (Akk: 60 min)	**Bli kjent med Mr. Abot** - Oppbygging av Mr. Abot? - Hvordan virker en servomotor? - Kommando for styring av servomotor? - Lager et enkelt program som styrer Mr. Abot - Studentene bygger opp programmet samtidig - Kalibrer motorene	
30 min (Akk: 90 min)	**Oppdrag 1A – Kom nærmest målet - uttesting** - De får et ferdig program for kjøring av Mr. Abot - De skal bli kjent med programmet - De skal løse oppdraget: Nærmest målet	
10 min (Akk: 100 min)	**Oppdrag 1B – Kom nærmest målet - Klargjøring** - Avstanden oppgis, deltagerne justerer parametrene	
20 min (Akk: 120 min)	**Oppdrag 1C – Kom nærmest målet - Konkurranse** - Det rigges til tre baner - Hver gruppe får to forsøk, beste resultatet regnes	
	Følg en strek	
30 min (Akk: 150 min)	**Introduksjon til refleksive sensorer** - Kort om sensoren - Om å forstå rutinen for å følge en svart strek - Studentene tester rutinen og justerer terskelen - Hva kan gjøres for at roboten skal gå raskere?	
20 min (Akk: 170 min)	**Oppdrag 2A - Raskest rundt i sløyfa - Uttesting** - Studentene tester på en minisløyfe	
20 min (Akk: 190 min)	**Oppdrag 2B - Raskest rundt i sløyfa - Konkurranse** - Hver gruppe kjører sin robot i sløyfa, tida blir notert	

Tabell 2: Forslag til gjennomføring av aktiviteten for studenter

Omfang	Innhold	Referanse
10 min (Akk: 200 min)	Avslutning og opprydding	

Oppgaver som må løses

- Tilpasse programmet slik at det kan brukes av studentene. Gjerne oppbygd av funksjoner.
- Lage et testark for testing av linjefølgefunksjonen
- Skrive tipsark for studentene

A.3 Elevaktivitet ved Vitensenteret

A.3.1 Omfang

3 timer inkludert lunsj.

A.3.2 Målgruppe

Elever i ungdomsskolen som har faget Teknologi i Praksis (TiP), men kan også være elever fra videregående skole som tar faget Teknologi og Forskningslære (ToF).

A.3.3 Målsetning

Målsetningen er å lære hvordan en robot kan programmeres og som trening i problemløsning. Det vil også være ønskelig at elevene skal bruke matematikk i forbindelse med løsning av oppgavene.

A.3.4 Utstyr

- 10 roboter av typen MrAbot
- 10 sett med div. sensorer
- 8 arbeidsstasjoner (PC)
- 1 eller arenaer

A.3.5 Forslag til oppgaver/problemstillinger

Under er ganske kort skissert en rekke forslag til oppgaver som burde kunne la seg realisere rent teknisk.

- **Treff nærmest målet**, uten hindringer, ukjent avstand til 5 min. før konkurranse da oppgis avstanden, og deltagerne må raskt stille inn sin robot. Deltagerne får 40 min. til å bli kjent med roboten før de får oppgitt avstanden.

- **Treff nærmest målet**, uten hindringer, ukjent avstand og ukjent idealtid inntil 5 min. før konkurransen
- **Treff nærmest målet**, med hindringer, ukjent geometri inntil 10 min. før konkurransen (ev. med idealtid)
- **Følg en sluttet bane** med slynger (følger svart strek), får utdelt lite testark med svart linje, konkurrerer på større bane, får ikke teste på virkelig bane før konkurranse, hvem er raskest
- **Følg en sluttet bane** med slynger og en hindring i banen (følger svart strek, men må avvike forbi hindring), får tildelt lite testark med svart linje, konkurrerer på større bane, får ikke teste på virkelig bane før konkurranse, hvem er raskest
- Finn veien ut av labyrinten: En robot er utstyrt med avstandssensor. Lage en strategi for komme ut av labyrinten på kortest mulig tid. Den enkleste type labyrint kan være et rom med en åpning (dør). deltagerne bruker enkle funksjoner for å kjøre rett fram og svinge. Vanskelighetsgraden kan økes. To roboter kan samtidig konkurrere i hver sin like labyrint.
- **Finn magnetsensorer:** I et bord med karmer er det skjult 12 magnetsensorer, roboten skal bevege seg over bordet innen en gitt tid og finne flest mulig av magnetsensorer. For hver sensor som finnes tennes en lysdiode langs kanten av bordet. Elevene programmerer roboten til å dekke et størst mulig areal innen tidsfristen. Elevene tar utgangspunkt i ferdig programvare.
- **Finn "hullet" og stopp:** Skal skanne over en bordplate på jakt etter et "hull" (svart merke). Deltagerne velger et startpunkt for roboten. Deretter kastes to terninger som angir koordinatene for hvor det svarte merket skal legges. Hvem finner "hullet" raskest. Maks to minutter.
- **Førstemann til "hullet".** Som over, men det er to roboter som konkurrerer om å finne "hullet" først.

A.4 Elevverksted ved Skolelaboratoriet

A.4.1 Omfang

5,5 timer inkludert lunsj.

A.4.2 Målgruppe

Opplegget er ment å være en komplett introduksjon til programmering og oppkobling av noen enkle kretser med Arduino for elever i ungdomsskole eller videregående skole. Det forutsettes ikke forkunnskaper, men det vil være en fordel om elevene har litt bakgrunn i elektrisitetslære. Gruppen bør ikke være større en 24 elever.

A.4.3 Målsetning

Hensikten med opplegget er å gi elevene en kort introduksjon og lyst til å utforske bruk av mikrokontrollere og programmering:

- Gi en kort introduksjon av hva mikrokontrollere er og hvordan de brukes (avsnitt 2.1, side 15)
- Gi "hands on" erfaring med å lage og programmere enkle kretser, se f.eks. avsnitt 7.1–7.2.1.
- Kunne løse noen enkle halvåpne oppgaver, se f.eks. avsnitt 7.3–7.4.

A.4.4 Utstyr

Følgende bør finnes tilgjengelig for gjennomføring med n antall elever:

- 1/2 n stk. Sparkfun's Inventor's kit slik at to og to elever kan arbeide sammen
- 1/2 n stk. PC-er med programvare og drivere installert
- 1 stk Multimeter for test
- 1/2 n stk Oppgaveark med løsninger
- n stk Ressurshefter
- 1 stk. PowerPoint presentasjon m/videokanon
- 1 stk. Suppleringsboks

A.4.5 Forslag til gjennomføring

Tabellen viser et forslag til hvordan en kan legge opp et elevverksted i programmering av mikrokontrollere for en gruppe elever.

Tabell 3: Forslag til gjennomføring av elevverksted

Omfang	Innhold	Referanse
5 min	Introduksjon, praktisk informasjon	
10 min	Hva er mikrokontrollere og hva brukes de til?	Avsnitt 2.1, side 15
15 min	Gjennomgang av komponenter og annet materiell	Avsnitt 3.2, side 23
30 min	Introduksjon til C-programmering	Avsnitt 5.3, side 52 og avsnitt 5.4, side 52
60 min	**Laboratorieøkt 1** - Koble opp en krets - Skrive inn og teste øving 1 - Felles gjennomgang av programmet og elektronikken	Avsnitt 7.1, side 79
60 min	**Laboratorieøkt 2** - Introduksjonsoppgaver knyttet til øving 1 - Felles gjennomgang	Avsnitt 7.1, side 79

Tabell 3: Forslag til gjennomføring av elevverksted

Omfang	Innhold	Referanse
30 min	Lunsj	
60 min	**Laboratorieøkt 3** - Individuelt arbeid med anbefalte øvinger - Felles gjennomgang	Avsnitt 7.2.1, side 82
	Laboratorieøkt 3 - Individuelt arbeid med egne valgte øvinger - Felles oppsummering av erfaring og svar på spørsmål	Avsnitt 7.2.1, side 82

A.4.6 Erfaringer

Oppsummering av erfaringer med ToF 1 klasse ved Byåsen videregående skole 5. juni 2013, med 24 elever (5 jenter og 19 gutter). Dagen etter verkstedet fikk elevene en rekke spørsmål som de besvarte:

1. *I hvilken grad forsto dere oppgavene dere løste?*

   Oppsummering: Elevene forsto oppgavene stort sett godt. Gjennomgangen på tavla fungerte greit. Det var også nyttig å ha skriftlig hjelpemateriell samt støtte i det som ble gjennomgått på tavla.

2. *Synes dere vi brukte for mye tid til styrte oppgaver, ville dere heller ha brukt mer tid til å eksperimentere på egen hånd?*

   Oppsummering: Fordelingen mellom styrte oppgaver og fri eksperimentering syntes ideell siden de aller fleste var ukjente med programmering av mikrokontrollere. Tempoet var dessuten ikke så høyt slik at noen fikk anledning til å gå litt ut over det som strengt tatt var oppgaven.

3. *Vi brukte mest tid på å forstå programmet. Burde vi ha brukt mer tid på å forstå de elektroniske komponentene?*

   Oppsummering: Samtlige synes det var riktig å fokusere på programmeringen framfor å bruke mer tid på komponentforståelse. De var dessuten stort sett kjent med de komponentene som ble brukt. Én nevner imidlertid at det kunne vært moro å visst mer om den fleksible motstanden og berøringspotensiometeret.

4. *Ser dere noen anvendelse av en slik programmerbar krets i faget ToF 1?*

   Oppsummering: Selv om stort sett alle er enig om at en har muligheter til å realisere prosjekter knyttet til ToF 1, så er det få som har konkrete forslag. Trafikklys nevnes som en mulighet, andre nevner at et slikt tema i ToF er viktig siden mikrokontrollere og programmering er så mye i bruk i alt vi omgir oss med.

5. *Kunne du tenkt deg en oversikt over hvilke muligheter en slik krets har, hvilke sensorer som finnes, og hva som kan styres?*

   Oppsummering: Mulig at spørsmålet var noe tvetydig. Behovet for en oversikt er derfor noe delt. Noen mener at fokuset på det vi holdt på med var tilstrekkelig, andre at de hadde en rimelig god oversikt over mulighetene, mens atter andre kunne tenke seg å teste ut muligheter, teste ut «grenser» som én sier. Man kunne på et tidspunkt vist dem noen konkrete eksempler på bruksområder, både hverdagseksempler, men også eksempler hentet fra introduksjonssettet som de brukte, eller andre tilsvarende byggesett.

6. *Dersom dere skulle prioritere. Hva ville dere foretrukket å jobbe med:*
   - Lys
   - Lyd, toner
   - Bevegelse
   - Bilder, film
   - Andre fysiske størrelser, temperatur, fuktighet, trykk, gasser

   Oppsummering: Både lys, lyd/toner og bevegelse kommer høyt oppe. En nevner at lyd er slitsomt i en større gruppe. Lyd er spennende fordi en kan lage melodier. Men også andre sensorer nevnes.

7. *Hvordan syntes dere forholdet mellom felles gjennomgang og egen selvstendig jobbing fungerte?*

   Oppsummering: Balansen mellom gjennomgang og egen jobbing syntes perfekt

8. *Dagen sett under ett, hvor fornøyd var dere med opplegget som ble gjennomført?*

   Oppsummering: Det synes som om deltagerne var svært fornøyd med dagen. Én nevner imidlertid at det kunne vært brukt mer tid til å gjennomgå programmene. En annen at det var moro å se hvordan mange var blitt fenget i løpet av dagen og hadde stor glede av å eksperimentere på egen hånd mot slutten av verkstedet.

Det synes som det valgte opplegget var perfekt for en klasse med litt bakgrunnskunnskaper i elektronikk, men omtrent ingen erfaring mht. programmering. Skulle en imidlertid laget en oppfølger, som en absolutt bør, måtte den gi dem noe mer utfordrende og kanskje noe mer åpne oppgaver som stilte krav til deres kunnskaper og kreativitet. Alene blir dette verkstedet stående som en positiv «happening» for elevene. Det ville også vært naturlig at de fikk ett knippe oppgaver å velge i. Forslag kan være: Trafikklys for den som ønsket å jobbe med lys. Melodier for de som ønsket å jobbe med lyd og en enkel robot for de som ønsket å jobbe med bevegelse.

Det bør også gjøres tilgjengelig en enkel manual hvor de på en forståelig måte kunne plukke ulike kommandoer som de har bruk for.

A.5 Lærerkurs over 2 dager

A.5.1 Omfang

14 timer inkludert lunsjpauser.

A.5.2 Målgruppe

Opplegget er ment å være en komplett introduksjon av programmering og oppkobling av noen enkle kretser med Arduino for lærere i ungdomsskole eller videregående skole. Det forutsettes ikke forkunnskaper, men det vil være en fordel om lærerne har grunnleggende kunnskaper om elektrisitetslære. Gruppen bør ikke være større enn 24 lærere.

A.5.3 Målsetning

Hensikten med opplegget er å gi lærerne en kort introduksjon og lyst til å utforske bruken av mikrokontrollere:

- Gi en kort introduksjon om av hva mikrokontrollere er og hvordan de brukes (avsnitt 2.1, side 15)
- Grunnleggende kunnskaper om noen elektroniske komponenter og hvordan de kan kobles opp i enkle kretser for å utføre en oppgave
- Gi "hands on" erfaring med å løse programmeringsoppgaver knyttet til enkle problemstillinger (avsnitt 7.1–7.2.1)
- Kunne løse noen enkle halvåpne oppgaver (avsnitt 7.3–7.4).
- Gi tips til hvordan introdusere temaet i egen klasse

A.5.4 Utstyr

Følgende bør finnes tilgjengelig for gjennomføring med n antall elever:

- 1/2 n stk. Sparkfun's Inventor's kit slik at to og to elever kan arbeide sammen
- 1/2 n stk. PC-er med programvare og drivere installert
- 1 stk Multimeter for test
- 1/2 n stk Oppgaveark med løsninger
- n stk Ressurshefte
- 1 stk. PowerPoint presentasjon m/videokanon
- 4 stk. Komplett utstyr for halvåpen oppgave: Batteritester
- 4 stk. Komplett utstyr for halvåpen oppgaver: Automatisk vannvarmer
- 4 stk. Komplett utstyr for halvåpen oppgave: Automatisk blomstervanner
- 1 stk. Suppleringsboks

A.5.5 Forslag til gjennomføring

Tabellen viser forslag til hvordan en kan legge opp et todagers grunnkurs i grunnleggende programmering av mikrokontrollere og litt elektronikk for en gruppe lærere.

Tabell 4: Forslag til gjennomføring av et todagers lærerkurs

Omfang	Innhold	Referanse
Dag I		
09:00–09:15	Velkommen. Praktisk informasjon	
09:15–10:00	Installasjon og montering, Gjennomgang av materiell	avsnitt 5.2 avsnitt 3.1–3.2
10:00–10:30	Introduksjon til C-programmering	avsnitt 5.3–5.4
10:30–12:00	**Laboratorieøkt 1** *Bruk av LED, potensiometer, trykknapp avbrutt av gjennomgang av programmene*	avsnitt 7.2.1
12:00–12:30	Lunsj	
12:30–15:00	**Laboratorieøkt 2** *Bruk av sensorer, fotomotstand, temperatursensor, piezoelektrisk lydgiver, motorer og rele*	avsnitt 7.2.1
15:00–15:15	**Hjemmearbeid til dag II:** *Lag en skisse til et undervisningsopplegg, bruk Arduino!*	
15:15–15:30	Oppsummering og refleksjon	
Dag II		
09:00–09:10	Praktisk informasjon	
09:10–09:40	Deltagerne presenterer sine forslag til undervisningsopplegg.	
09:40–10:30	**Laboratorieøkt 3** Arbeid med halv åpne oppgaver	avsnitt 7.3
10:30–11:00	Introduksjon til **Fritzing** for dokumentasjon av oppkobling.	avsnitt 6.1
	Laboratorieøkt 4 Fortsette arbeid med halv åpne oppgaver	avsnitt 7.3

Tabell 4: Forslag til gjennomføring av et todagers lærerkurs

Omfang	Innhold	Referanse
12:00–12:45	Lunsj	
12:45–13:05	CanSat, et mulig ToF-prosjekt med Arduino	
13:05–15:00	**Laboratorieøkt 4** Fortsette arbeid med halv åpne oppgaver	avsnitt 7.3
15:00–15:15	Oppsummering, utveksling av erfaringer	

A.5.6 Erfaringer

Etter gjennomført lærerkurs 5. og 6. mars 2013 med ca. 15 ToF-lærere fra videregående skole, de fleste med liten eller ingen erfaring med denne type programmering, høstet vi følgende erfaringer:

- **La deltagerne skrive inn det første programmet** for hånd istedet for å laste det ned fra eksempelfolderen. Dette gjelder for så vidt alltid der det er snakk om førstegangsbrukere. Dette gir erfaringer med innskriving og hvilke konsekvenser feilskrivinger har.
- **Ikke overvurder deltagernes grunnlag for å forstå elektronikk** tilknyttet oppgavene. Selv om dette er relativt enkle kretser, så er det få lærere som har erfaring med elektronikk.
- **Overgangen fra å forstå ferdigskrevne programmer til selv å komponere programmer er ofte større enn man skulle tro**. Det er derfor ofte ikke tilstrekkelig å gjennomgå eksempler, man må også gi dem grunnleggende oppskrifter for hvordan komme i gang med å konstruere programmer selv.

Ikke undervurder tiden det tar å løse en halvåpen oppgave. En skal ikke forvente at de i løpet av en dag klarer å løse mer enn én eller to oppgaver av typen presentert i avsnitt 7.3, side 85. For nybegynneren kan én slik oppgave være nok.

Vedlegg B Løsningsforslag på introduksjonsoppgaver

Oppgavene bygger på oppgave en i SIK guiden og gir stadig nye utfordringer til denne innledende oppgaven.

B.1 Opprinnelige oppgave

```
// Denne koden skriver elevene inn for hånd
void setup()
{
  pinMode(13, OUTPUT);
}

void loop()
{

  digitalWrite(13, HIGH);                    // Turn on the LED
  delay(1000);                               // Wait for one second
  digitalWrite(13, LOW);                     // Turn off the LED
  delay(1000);                               // Wait for one second
}
```

B.2 Innføring av variabler

```
// Innfører variablene pinDiode og blinkDelay

int pinDiode = 13;
int blinkDelay = 500;

void setup()
{
  pinMode(pinDiode, OUTPUT);
}

void loop()
{

  digitalWrite(pinDiode, HIGH);              // Turn on the LED
  delay(blinkDelay);                         // Wait for one second
  digitalWrite(pinDiode, LOW);               // Turn off the LED
```

```
  delay(blinkDelay);                    // Wait for one second
}
```

B.3 Kode som blinker SOS

```
// SOS

int pinDiode = 13;

int prikkLengde = 200;                  // Definerer lengen av en prikk, 200 msek
int strekLengde = 600;                  // Definerer lengen av en strek, 600 msek
int tegn_tegnLengde = 200;              // Definerer tidsrommet mellom to tegn, 200 msek
int bokstav_bokstavLengde = 600;        // Definerer tidsrommet mellom to bokstaver 600 msek
int ord_ordLengde = 1800;               // // Definerer tidsrommet mellom to ord 1800 msek

void setup()
{
  pinMode(pinDiode, OUTPUT);
}

void loop()
{

  // Send S som tre prikker ***
  digitalWrite(pinDiode, HIGH);
  delay(prikkLengde);
  digitalWrite(pinDiode, LOW);

  delay(tegn_tegnLengde);

  digitalWrite(pinDiode, HIGH);
  delay(prikkLengde);
  digitalWrite(pinDiode, LOW);

  delay(tegn_tegnLengde);

  digitalWrite(pinDiode, HIGH);
  delay(prikkLengde);
  digitalWrite(pinDiode, LOW);
```

```
delay(bokstav_bokstavLengde);

// Sende O som tre streker - - -
digitalWrite(pinDiode, HIGH);
delay(strekLengde);
digitalWrite(pinDiode, LOW);

delay(tegn_tegnLengde);

digitalWrite(pinDiode, HIGH);
delay(strekLengde);
digitalWrite(pinDiode, LOW);

delay(tegn_tegnLengde);

digitalWrite(pinDiode, HIGH);
delay(strekLengde);
digitalWrite(pinDiode, LOW);

delay(bokstav_bokstavLengde);

// Send S som tre prikker ***
digitalWrite(pinDiode, HIGH);
delay(prikkLengde);
digitalWrite(pinDiode, LOW);

delay(tegn_tegnLengde);

digitalWrite(pinDiode, HIGH);
delay(prikkLengde);
digitalWrite(pinDiode, LOW);

delay(tegn_tegnLengde);

digitalWrite(pinDiode, HIGH);
delay(prikkLengde);
digitalWrite(pinDiode, LOW);
```

```
  // Mellomrom ord - ord
  delay(ord_ordLengde);

}
```

B.4 SOS med variabel hastighet

```
// SOS med variabel hastighet

int pinDiode = 13;

int hastighet = 50;                          // Definerer hastigheten
int prikkLengde = 2*hastighet;               // Definerer prikklengde relativt til hastigheten
int strekLengde = 6*hastighet;               // Definerer streklengde relativt til hastigheten
int tegn_tegnLengde = 2*hastighet;
int bokstav_bokstavLengde = 6*hastighet;
int ord_ordLengde = 18*hastighet;

void setup()
{
  pinMode(pinDiode, OUTPUT);
}

void loop()
{

  // S ***
  digitalWrite(pinDiode, HIGH);
  delay(prikkLengde);
  digitalWrite(pinDiode, LOW);

  delay(tegn_tegnLengde);

  digitalWrite(pinDiode, HIGH);
  delay(prikkLengde);
  digitalWrite(pinDiode, LOW);

  delay(tegn_tegnLengde);

  digitalWrite(pinDiode, HIGH);
```

```
delay(prikkLengde);
digitalWrite(pinDiode, LOW);

delay(bokstav_bokstavLengde);

// O - - -
digitalWrite(pinDiode, HIGH);
delay(strekLengde);
digitalWrite(pinDiode, LOW);

delay(tegn_tegnLengde);

digitalWrite(pinDiode, HIGH);
delay(strekLengde);
digitalWrite(pinDiode, LOW);

delay(tegn_tegnLengde);

digitalWrite(pinDiode, HIGH);
delay(strekLengde);
digitalWrite(pinDiode, LOW);

delay(bokstav_bokstavLengde);

// S ***
digitalWrite(pinDiode, HIGH);
delay(prikkLengde);
digitalWrite(pinDiode, LOW);

delay(tegn_tegnLengde);

digitalWrite(pinDiode, HIGH);
delay(prikkLengde);
digitalWrite(pinDiode, LOW);

delay(tegn_tegnLengde);

digitalWrite(pinDiode, HIGH);
```

```
  delay(prikkLengde);
  digitalWrite(pinDiode, LOW);

  // Mellomrom ord - ord
  delay(ord_ordLengde);

}
```

B.5 SOS med lys og lyd

```
// SOS med lys og lyd

int pinDiode = 13;
int pinBuzzer = 9;

int hastighet = 50;
int prikkLengde = 2*hastighet;
int strekLengde = 6*hastighet;
int tegn_tegnLengde = 2*hastighet;
int bokstav_bokstavLengde = 6*hastighet;
int ord_ordLengde = 18*hastighet;

int frekvens = 880; // Hertz

void setup()
{
  pinMode(pinDiode, OUTPUT);
  pinMode (pinBuzzer, OUTPUT);
}

void loop()
{

  // S ***
  digitalWrite(pinDiode, HIGH);
  tone(pinBuzzer, frekvens);
  delay(prikkLengde);
  digitalWrite(pinDiode, LOW);
  noTone(pinBuzzer);
```

```
delay(tegn_tegnLengde);

digitalWrite(pinDiode, HIGH);
tone(pinBuzzer, frekvens);
delay(prikkLengde);
digitalWrite(pinDiode, LOW);
noTone(pinBuzzer);

delay(tegn_tegnLengde);

digitalWrite(pinDiode, HIGH);
tone(pinBuzzer, frekvens);
delay(prikkLengde);
digitalWrite(pinDiode, LOW);
noTone(pinBuzzer);

delay(bokstav_bokstavLengde);

// O - - -
digitalWrite(pinDiode, HIGH);
tone(pinBuzzer, frekvens);
delay(strekLengde);
digitalWrite(pinDiode, LOW);
noTone(pinBuzzer);

delay(tegn_tegnLengde);

digitalWrite(pinDiode, HIGH);
tone(pinBuzzer, frekvens);
delay(strekLengde);
digitalWrite(pinDiode, LOW);
noTone(pinBuzzer);

delay(tegn_tegnLengde);

digitalWrite(pinDiode, HIGH);
tone(pinBuzzer, frekvens);
delay(strekLengde);
```

```
  digitalWrite(pinDiode, LOW);
  noTone(pinBuzzer);

  delay(bokstav_bokstavLengde);

  // S ***
  digitalWrite(pinDiode, HIGH);
  tone(pinBuzzer, frekvens);
  delay(prikkLengde);
  digitalWrite(pinDiode, LOW);
  noTone(pinBuzzer);

  delay(tegn_tegnLengde);

  digitalWrite(pinDiode, HIGH);
  tone(pinBuzzer, frekvens);
  delay(prikkLengde);
  digitalWrite(pinDiode, LOW);
  noTone(pinBuzzer);

  delay(tegn_tegnLengde);

  digitalWrite(pinDiode, HIGH);
  tone(pinBuzzer, frekvens);
  delay(prikkLengde);
  digitalWrite(pinDiode, LOW);
  noTone(pinBuzzer);

  // Mellomrom ord - ord
  delay(ord_ordLengde);

}
```

B.6 SOS med bruk av funksjoner

```
// SOS med variabel hastighet og bruk av funksjoner

int pinDiode = 13;
int pinBuzzer = 9;
```

```
int hastighet = 50;
int prikkLengde = 2*hastighet;
int strekLengde = 6*hastighet;
int tegn_tegnLengde = 2*hastighet;
int bokstav_bokstavLengde = 6*hastighet;
int ord_ordLengde = 18*hastighet;

int frekvens = 440; // Hertz

void setup()
{
  pinMode(pinDiode, OUTPUT);
  pinMode (pinBuzzer, OUTPUT);
}

void loop()
{

 s();                                    // Kall funksjonen S

  delay(bokstav_bokstavLengde);

 o();                                    // Kall funksjonen O

  delay(bokstav_bokstavLengde);

  s();                                   // Kall funksjonen S

  // Mellomrom ord - ord
  delay(ord_ordLengde);

  while (true);

}

void s ()
{
   // S ***
```

```
  digitalWrite(pinDiode, HIGH);
  tone(pinBuzzer, frekvens);
  delay(prikkLengde);
  digitalWrite(pinDiode, LOW);
  noTone(pinBuzzer);

  delay(tegn_tegnLengde);

  digitalWrite(pinDiode, HIGH);
  tone(pinBuzzer, frekvens);
  delay(prikkLengde);
  digitalWrite(pinDiode, LOW);
  noTone(pinBuzzer);

  delay(tegn_tegnLengde);

  digitalWrite(pinDiode, HIGH);
  tone(pinBuzzer, frekvens);
  delay(prikkLengde);
  digitalWrite(pinDiode, LOW);
  noTone(pinBuzzer);
}

void o()
{
    // O - - -
  digitalWrite(pinDiode, HIGH);
  tone(pinBuzzer, frekvens);
  delay(strekLengde);
  digitalWrite(pinDiode, LOW);
  noTone(pinBuzzer);

  delay(tegn_tegnLengde);

  digitalWrite(pinDiode, HIGH);
  tone(pinBuzzer, frekvens);
  delay(strekLengde);
  digitalWrite(pinDiode, LOW);
```

```
  noTone(pinBuzzer);

  delay(tegn_tegnLengde);

  digitalWrite(pinDiode, HIGH);
  tone(pinBuzzer, frekvens);
  delay(strekLengde);
  digitalWrite(pinDiode, LOW);
  noTone(pinBuzzer);
}
```

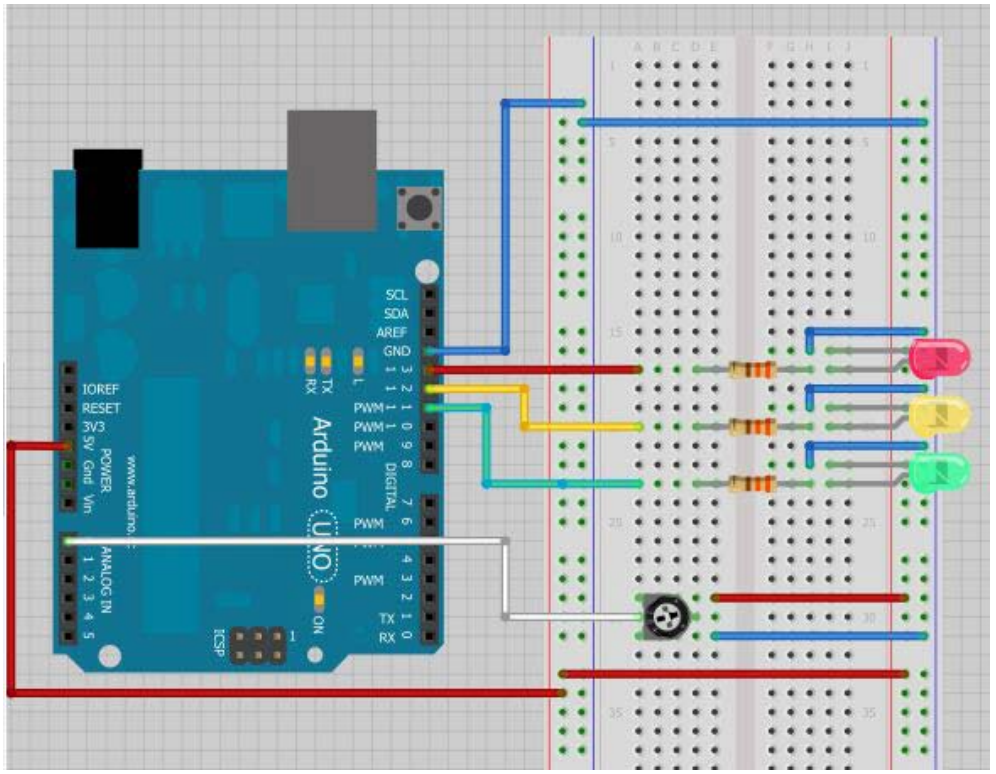
Vedlegg C Løsningsforslag på halvåpne oppgaver

C.1 Batteritester

Oppgaveteksten finnes på side 85.

C.1.1 Oppkobling

Oppkoblingen inkluderer et potensiometer som skal "simulere" et batteri hvor spenningen blir dårligere etter som batteriet svekkes. I stedet for at spenningen faller, så justeres spenningen ut fra potensiometeret ned. Ved tilfredsstillende "batterispenning lyser den grønne dioden. Ved lav, men tilstrekkelig "batterispenning" lyser den gule dioden, og ved for lav "batterispenning" lyser den røde dioden. Under målingen bør batteriet belastes med en passende belastningsmotstand.



Figur C.1 Oppkobling Batteritester.

C.1.2 Løsningsforslag oppgave A

```
/*
  Batteritester Oppgave 1A
```

```
  Lag en batteritester som måler spenningen på batteret,
  og skriver resultatet til monitoren
*/

int analogPin =  0;                          // Kontakt for måling av spenning
int verdiDigital = 0;                         // Avlest digital verdi
int kalibrertDigital = 655;                  // Avlest maks digital spenning
float kalibrertSpenning = 3.22;              // Maks verdi er 3,22 V
float verdiSpenning;                         // Målt kalibrert spenningsverdi

// Setup definerer hva som er utganger og hva som er innganger

void setup()
{
  Serial.begin(9600);                        // Sett opp kommunikasjon til monitor
  pinMode(analogPin,INPUT);                  //
}

// loop() kjører om og om igjen i det undelige,

void loop()
{
 verdiDigital = analogRead(analogPin);       // Les av spenningen
 delay(500);
 verdiSpenning = (1.0*verdiDigital/kalibrertDigital)*kalibrertSpenning;
 Serial.print("Digital batterispenning:" );
 Serial.print(verdiDigital);
 Serial.print(", Maalt batterispenning:" );
 Serial.println(verdiSpenning);

}
```

C.1.3 Løsningsforslag oppgave B

```
/*
  Batteritester Oppgave 1B
  Lag en batteritester som måler spenningen på batteret,
  og skriver en tekst om batteriet kan brukes eller ikke
*/
```

```
int analogPin =  0;                          // Kontakt for måling av spenning
int verdiDigital = 0;                        // Avlest digital verdi
int kalibrertDigital = 655;                  // Avlest maks digital spenning
float kalibrertSpenning = 3.22;              // Maks verdi er 3,22 V
float verdiSpenning;                         // Målt kalibrert spnningsverdi
float terskelGodtBatteri = 3.0;      /       / Nedre terskel for godt batteri
float terskelUbrukeligBatteri = 2.7;         // Øvre terskel for ubrukelig batteri

// Setup definerer hva som er utganger og hva som er innganger

void setup()
{
  Serial.begin(9600);                        // Sett opp kommunikasjon til monitor
  pinMode(analogPin,INPUT);                  //
}

// loop() kjører om og om igjen i det undelige,

void loop()
{
 verdiDigital = analogRead(analogPin);       // Les av spenningen
 delay(500);
 verdiSpenning = (1.0*verdiDigital/kalibrertDigital)*kalibrertSpenning;

 if (verdiSpenning > terskelGodtBatteri)
 {
   Serial.print("Batteriet er godt, spenning: ");
   Serial.print(verdiSpenning);
   Serial.println(" Volt");
 }
 else if ((verdiSpenning < terskelGodtBatteri) && (verdiSpenning > terskelUbrukeligBatteri))
 {
   Serial.print("Batteriet er daarlig men kan brukes, spenning: ");
   Serial.print(verdiSpenning);
   Serial.println(" Volt");
 }
 else
```

```
 {
   Serial.print("Batteriet er ubrukelig og boer kastes, spenning: ");
   Serial.print(verdiSpenning);
   Serial.println(" Volt");
 }
}
```

C.1.4 Løsningsforslag oppgave C

```
/*
  Batteritester Oppgave 1C
  Lag en batteritester som måler spenningen på batteret,
  og skriver en tekst om batteriet kan brukes eller ikke.
  Dessuten skal en grønn, gul eller rød lysdiode lyse for å vise kvaiteten til batteriet
*/

int analogPin =  0;                     // Kontakt for måling av spenning
int verdiDigital = 0;                   // Avlest digital verdi
int kalibrertDigital = 655;             // Avlest maks digital spenning
boolean God = false;
boolean Daarlig = false;
boolean Ubrukelig = false;
int pinGod = 11;
int pinDaarlig = 12;
int pinUbrukelig = 13;
float kalibrertSpenning = 3.22;         // Maks verdi er 3,22 V
float verdiSpenning;                    // Målt kalibrert spnningsverdi
float terskelGodtBatteri = 3.0;         // Nedre terskel for godt batteri
float terskelUbrukeligBatteri = 2.7;    // Øvre terskel for ubrukelig batteri

// Setup definerer hva som er utganger og hva som er innganger

void setup()
{
  Serial.begin(9600);                   // Sett opp kommunikasjon til monitor
  pinMode(analogPin,INPUT);             //
  pinMode(pinGod, OUTPUT);
  pinMode(pinDaarlig, OUTPUT);
  pinMode(pinUbrukelig, OUTPUT);
```

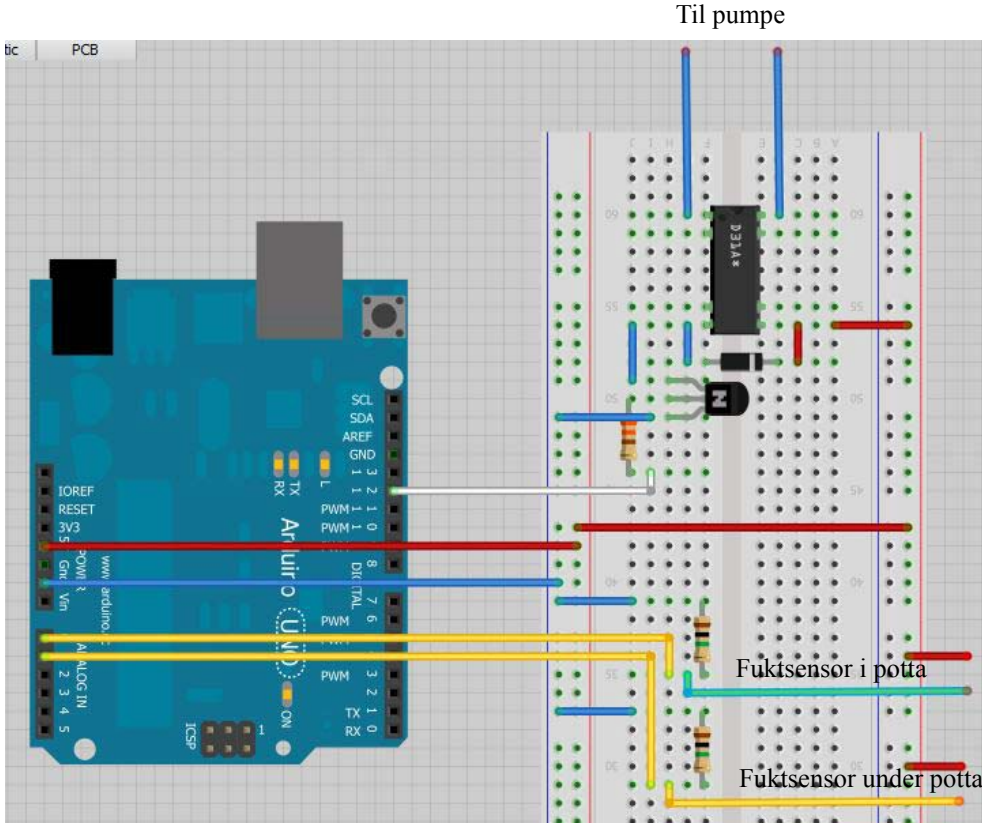
```
}

// loop() kjører om og om igjen i det undelige,

void loop()
{
 verdiDigital = analogRead(analogPin);              // Les av spenningen
 delay(500);
 verdiSpenning = (1.0*verdiDigital/kalibrertDigital)*kalibrertSpenning;

 if (verdiSpenning > terskelGodtBatteri)
 {
   Serial.print("Batteriet er godt, spenning: ");
   Serial.print(verdiSpenning);
   Serial.println(" Volt");
   God = true;
   Daarlig = false;
   Ubrukelig = false;

 }
 else if ((verdiSpenning < terskelGodtBatteri) && (verdiSpenning > terskelUbrukeligBatteri))
 {
   Serial.print("Batteriet er daarlig men kan brukes, spenning: ");
   Serial.print(verdiSpenning);
   Serial.println(" Volt");
   God = false;
   Daarlig = true;
   Ubrukelig = false;
 }
 else
 {
   Serial.print("Batteriet er ubrukelig og boer kastes, spenning: ");
   Serial.print(verdiSpenning);
   Serial.println(" Volt");
   God = false;
   Daarlig = false;
   Ubrukelig = true;
 }
```

```
  digitalWrite(pinGod, God);
  digitalWrite(pinDaarlig, Daarlig);
  digitalWrite(pinUbrukelig, Ubrukelig);
}
```

C.2 Automatisk blomstervanner

C.2.1 Oppkobling

Kretsen har to sensorer en som måler fuktigheten i jorda i potta (*fuktsensor i potte*), og en som måler fuktighet på skåla eller på gulvet rundt potta (*fuktsensor under potta*). Den siste skal være en sikkerhet som gjør at vanningen avbrytes umiddelbart. I tillegg benyttes et rele som kobler inn pumpa fra ekstern strømkilde. Det gjøres oppmerksom på at det releet som er avbildet på figuren under ehar en annen utforming enn releet som følger med Sparkfun Inventors kit.



Figur C.2 Oppkobling automatisk blomstervanner.

C.2.2 Løsningsforslag oppgave A

```
/*
  Blomstarvanner Oppgave 1A
  Det skal lages en krets som automatisk vanner blomster
  når blomsten er blitt tørr. Mengden vann skal kunne tilpasses
  plantens behov.
*/

int vanningPin = 12;
int torrPin = 0;
int torr = 0;                          // Avlest grad av fuktighet
int terskel_torr = 300;                // Terskelverid for når planten er tørr
int mengde_vann = 2000;                // Angir mengde vann
int min_tid_hver_vanning = 5000;       // Minimumstid mellom hver vanning

// Setup definerer hva som er utganger og hva som er innganger

void setup()
{
  pinMode(vanningPin,OUTPUT);
  pinMode(torrPin,INPUT);
  Serial.begin(9600);
}

// loop() kjører om og om igjen i det undelige,

void loop()
{
  torr = analogRead(torrPin);
  Serial.print("Torrverdi: ");
  Serial.println(torr);
  delay(1000);
  if (torr < terskel_torr)
  {
    digitalWrite(vanningPin,HIGH);
    delay(mengde_vann);
    digitalWrite(vanningPin,LOW);
    delay(min_tid_hver_vanning);
```

```
  }
}
```

C.2.3 Løsningsforslag oppgave B

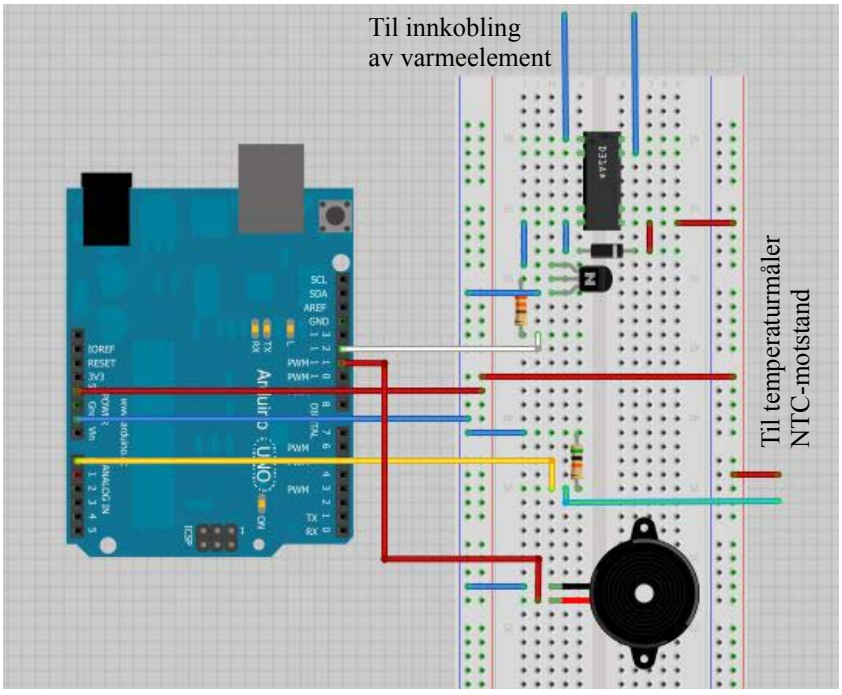
```
/*
  Blomstervanner Oppgave B
  Det skal lages en krets som automatisk vanner blomster
  når blomsten er blitt tørr. Mengden vann skal kunne tilpasses
  plantens behov. I tillegg skal det legges inn sikkerhet mot lekkasje.
*/

int vanningPin = 12;
int torrPin = 0;
int lekkasjePin = 1;
int torr = 0;                          // Avlest grad av fuktighet
int lekkasje = 0;
int terskel_torr = 300;                // Terskelverdi for når planten er tørr
int terskel_lekkasje = 300;            // Terskelverdi for deteksjon av lekkasje
int mengde_vann = 2000;                // Angir mengde vann
int min_tid_hver_vanning = 5000;       // Minimumstid mellom hver vanning

// Setup definerer hva som er utganger og hva som er innganger

void setup()
{
  pinMode(vanningPin,OUTPUT);
  pinMode(torrPin,INPUT);
  Serial.begin(9600);
}

// loop() kjører om og om igjen i det uendelige,

void loop()
{
  torr = analogRead(torrPin);
  lekkasje = analogRead(lekkasjePin);
  Serial.print("Torrverdi: ");
  Serial.println(torr);
```

```
  delay(1000);

  if ((torr < terskel_torr) && (lekkasje < terskel_lekkasje))
  {
    digitalWrite(vanningPin,HIGH);
    delay(mengde_vann);
    digitalWrite(vanningPin,LOW);
    delay(min_tid_hver_vanning);
  }
}
```

C.3 Vannvarmer med termostat

C.3.1 Oppkobling

Kretsen styrer et rele som kan slå av og på et lite varmeelement laget med en glødetråd. Ved hjelp av en NTC-motstand måles temperaturen. NTC-motstanden er karakterisert på forhånd, dvs. man kjenner sammenhengen mellom temperatur og motstand. Kretsen regulerer temperaturen i glasset rundt den ønskede temperaturen.Dernest legges det inn en lyd når temperaturen passerer 30C. En høy tone på vei opp og dyp tone på vei ned.



C.3.2 Løsningsforslag oppgave

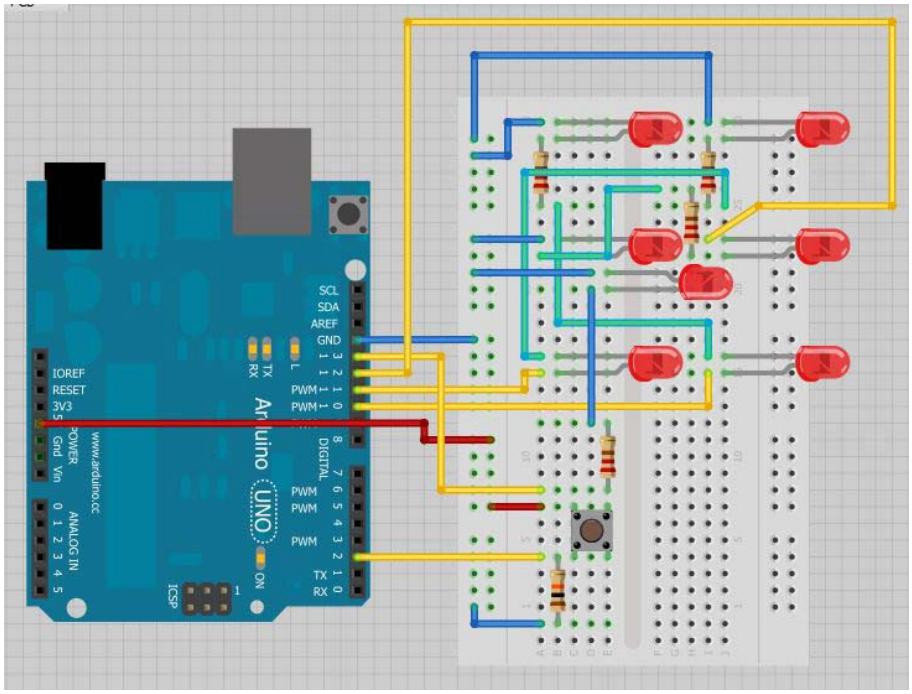
```
/*
  Vannvarmer Oppgave 1A,B,C
  Det skal lages en vannvarmer som varmer opp vann.
  Ved å måle strøm og spenning over varmeelementet skal en beregne
  hvor lang tid det vil ta å varme opp 20 ml vann fra 20C til 25C
  Strøm x Spenning = Effekt (W), 1 Watt x 1 sek = 1 J
  Det skal 4,18 J til for å varme opp 1ml til 1 grad K
  Dernest skal det legges inn lyd når temperaturen passerer 30C.
  En høy tone på vei opp og dyp tone på vei ned.
*/

int oppvarmingPin = 12;
int lydPin = 11;
boolean lydFlag = true;
int temperaturPin = 0;
int kal_digiLav = 459;
int kal_digiHoy = 637;
float kal_anaLav = 21.3;
float kal_anaHoy = 36.6;
float a,b;
int tempDigital;
float tempAnalog;
float terskel_temp_C = 30;          //


// Setup definerer hva som er utganger og hva som er innganger

void setup()
{
  pinMode(oppvarmingPin,OUTPUT);
  pinMode(temperaturPin,INPUT);
  pinMode(lydPin,OUTPUT);
  Serial.begin(9600);
}

// loop() kjører om og om igjen i det uendelige,
```

```
void loop()
{
  tempDigital = analogRead(temperaturPin);
  a = (kal_anaHoy - kal_anaLav)/(kal_digiHoy - kal_digiLav);
  b = kal_anaHoy - a*kal_digiHoy;
  tempAnalog = a*tempDigital + b;
  Serial.print("Temperatur digital: ");
  Serial.println(tempAnalog);
  delay(1000);

  if (tempAnalog < terskel_temp_C)
  {
    if (lydFlag == true)
      {
        tone(lydPin,2000,200);
        lydFlag = false;
      }
    digitalWrite(oppvarmingPin,HIGH);
    delay(1000);
  }
  else
  {
    if (lydFlag == false)
      {
        tone(lydPin,300,200);
        lydFlag = true;
      }
    digitalWrite(oppvarmingPin,LOW);
    delay(1000);
  }
}
```

C.4 Elektronisk terning

C.4.1 Oppkobling

Ved å trykke på knappen viser terningen et tilfeldig tall mellom 1 og 6.



C.4.2 Løsningsforslag oppgave A

```
// Elektronisk terning
// 1    2
// 3  4  5
// 6    7

int kast;
int tilfeldig;

const int pinne_1_7 = 10;
const int pinne_2_6 = 11;
const int pinne_3_5 = 12;
const int pinne_4 = 13;
const int pinne_kast = 2;                    // IO-port for interrupt
```

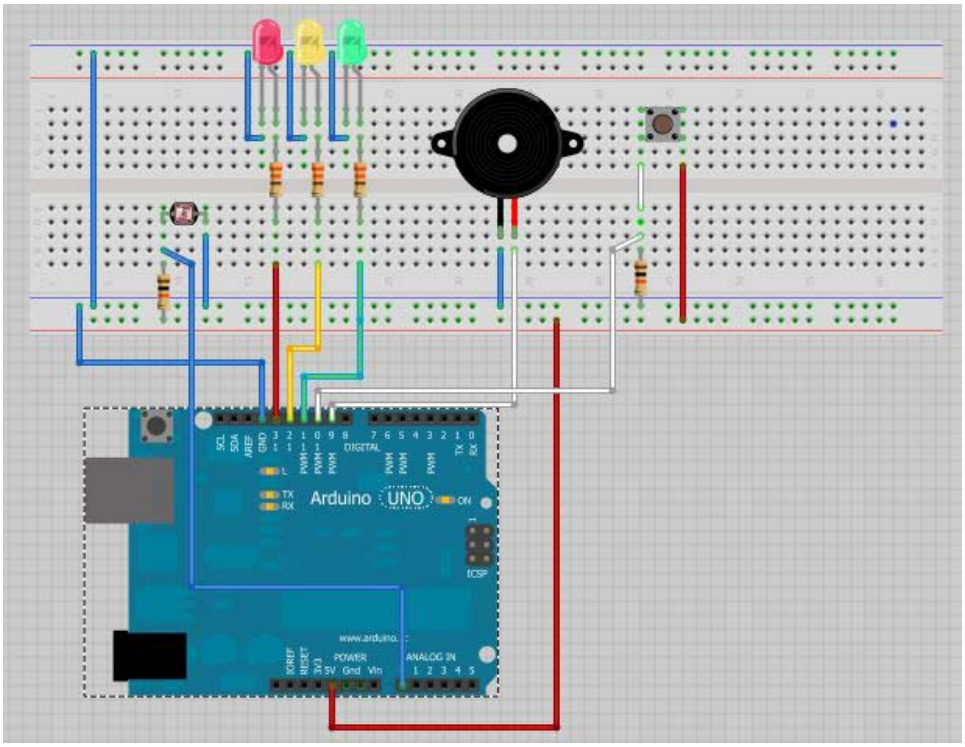
```
void setup ()
{
  attachInterrupt(0,visResultat,RISING);          // Initier interrupt på IO-port 2

  pinMode(pinne_1_7,OUTPUT);
  pinMode(pinne_2_6,OUTPUT);
  pinMode(pinne_3_5,OUTPUT);
  pinMode(pinne_4,OUTPUT);
  pinMode(pinne_kast,INPUT);
}

void loop()
{
  tilfeldig = random(1,7);
  kast = analogRead(pinne_kast);
}

void visResultat()
 {
    // Nullstill alle lysdioder
    digitalWrite(pinne_1_7,LOW);
    digitalWrite(pinne_2_6,LOW);
    digitalWrite(pinne_3_5,LOW);
    digitalWrite(pinne_4,LOW);

    if (tilfeldig == 1)
    {
     digitalWrite(pinne_4,HIGH);
    }

    if (tilfeldig == 2)
    {
     digitalWrite(pinne_1_7,HIGH);
    }
    if (tilfeldig == 3)
    {
     digitalWrite(pinne_1_7,HIGH);
```

```
    digitalWrite(pinne_4,HIGH);
   }
   if (tilfeldig == 4)
   {
    digitalWrite(pinne_1_7,HIGH);
    digitalWrite(pinne_2_6,HIGH);
   }
   if (tilfeldig == 5)
   {
    digitalWrite(pinne_1_7,HIGH);
    digitalWrite(pinne_2_6,HIGH);
    digitalWrite(pinne_4,HIGH);
   }
   if (tilfeldig == 6)
   {
    digitalWrite(pinne_1_7,HIGH);
    digitalWrite(pinne_2_6,HIGH);
    digitalWrite(pinne_3_5,HIGH);
   }
  delay(100);
}
```

C.5 Trafikklys

C.5.1 Oppkobling

Trafikklyset skal gå av og på slik et trafikklys skal gjøre (A). Dernest skal en ved å trykke på knappen bestille grønt for fotgjenger (B), videre skal lyset blinke når det nærmer seg tid for å skifte til rødt (C). I den neste oppgaven skal det blinkende grønne lyset suppleres med lyd (D), før en til slutt skal sørge for at kretsen går over til blinkende gult idet det blir mørkt (E).



C.5.2 Løsningsforslag oppgave A

```
/*
  Trafikklys Oppgave A
  Lag et trafikklys som lyser slik som et trafikklys skal lyse
  når det skifter fra rødt til grønt og tilbake:
  -
*/

int ledPinRed =  13;                    // Kontakt rød LED til pinne pin 13
int ledPinYellow =  12;                 // Kontakt rød LED til pinne pin 12
int ledPinGreen =  11;                  // Kontakt rød LED til pinne pin 11

// Setup definerer hva som er utganger og hva som er innganger

void setup()  {
```

```
  pinMode(ledPinRed, OUTPUT);
  pinMode(ledPinYellow, OUTPUT);
  pinMode(ledPinGreen, OUTPUT);
}

// loop() kjører om og om igjen i det uendelige,

void loop()
{
  digitalWrite(ledPinRed, HIGH);          // Slå på rød lysdiode
  delay(5000);
  digitalWrite(ledPinYellow, HIGH);       // Slå på gul lysdiode
  delay(1000);
  digitalWrite(ledPinRed, LOW);           // Slå av rød lysdiode
  digitalWrite(ledPinYellow, LOW);        // Slå av gul lysdiode
  digitalWrite(ledPinGreen, HIGH);        // Slå av grønn lysdiode
  delay(5000);
  digitalWrite(ledPinGreen, LOW);         // Slå av grønn lysdiode
  digitalWrite(ledPinYellow, HIGH);       // Slå på gul lysdiode
  delay(1000);
  digitalWrite(ledPinYellow, LOW);        // Slå på gul lysdiode
}
```

C.5.3 Løsningsforslag oppgave B

```
/*
  Trafikklys Oppgave B
  Lag et trafikklys som lyser slik som et trafikklys skal lyse
  når det skifter fra rødt til grønt og tilbake.
  Skifte skal imidlertid kun inntreffe ved trykk på knappen.
*/

int ledPinRed =  13;                      // Kontakt rød LED til pinne pin 13
int ledPinYellow =  12;                   // Kontakt rød LED til pinne pin 12
int ledPinGreen =  11;                    // Kontakt rød LED til pinne pin 11
int knappBestilling = 10;                 // Kontakt knapp for bestilling av grønn mann
boolean bestilling = false;               // Boolsk variabel som holde bestilling
```

```
// Setup definerer hva som er utganger og hva som er innganger

void setup()  {
  pinMode(ledPinRed, OUTPUT);
  pinMode(ledPinYellow, OUTPUT);
  pinMode(ledPinGreen, OUTPUT);
  pinMode(knappBestilling, INPUT);
}

// loop() kjører om og om igjen i det uendelige,

void loop()
{

  bestilling = digitalRead (knappBestilling);       //Les av knappens innstilling

  if (bestilling == true)
   {
    digitalWrite(ledPinRed, HIGH);                  // Slå på rød lysdiode
    delay(5000);
    digitalWrite(ledPinYellow, HIGH);               // Slå på gul lysdiode
    delay(1000);
    digitalWrite(ledPinRed, LOW);                   // Slå av rød lysdiode
    digitalWrite(ledPinYellow, LOW);                // Slå av gul lysdiode
    digitalWrite(ledPinGreen, HIGH);                // Slå av grønn lysdiode
    delay(5000);
    digitalWrite(ledPinGreen, LOW);                 // Slå av grønn lysdiode
    digitalWrite(ledPinYellow, HIGH);               // Slå på gul lysdiode
    delay(1000);
    digitalWrite(ledPinYellow, LOW);                // Slå på gul lysdiode
    bestilling = false;                             // Sett bestilling til null/falsk
   }
   else
   {
     digitalWrite(ledPinRed, HIGH);                 // Slå på rød lysdiode
   }
}
```

C.5.4 Løsningsforslag oppgave C

```
/*
  Trafikklys Oppgave C
  Lag et trafikklys som lyser slik som et trafikklys skal lyse
  når det skifter fra rødt til grønt og tilbake.
  Skifte skal imidlertid kun inntreffe ved trykk på knappen. Fra nå av skal
  det grønne lyset lyse i 8 sek. Hvorav de siste 5 sek skal blinke
  med et halvt sekund på og et halvt sekund av
*/

int ledPinRed =  13;                    // Kontakt rød LED til pinne pin 13
int ledPinYellow =  12;                 // Kontakt rød LED til pinne pin 12
int ledPinGreen =  11;                  // Kontakt rød LED til pinne pin 11
int knappBestilling = 10;               // Kontakt knapp for bestilling av grønn mann
boolean bestilling = false;             // Boolsk variabel som holde bestilling


// Setup definerer hva som er utganger og hva som er innganger

void setup()  {
  pinMode(ledPinRed, OUTPUT);
  pinMode(ledPinYellow, OUTPUT);
  pinMode(ledPinGreen, OUTPUT);
  pinMode(knappBestilling, INPUT);
}

// loop() kjører om og om igjen i det undelige,

void loop()
{

  bestilling = digitalRead (knappBestilling);       //Les av knappens innstilling

  if (bestilling == true)
   {
    digitalWrite(ledPinRed, HIGH);                  // Slå på rød lysdiode
    delay(5000);
    digitalWrite(ledPinYellow, HIGH);               // Slå på gul lysdiode
```

```
    delay(1000);
    digitalWrite(ledPinRed, LOW);          // Slå av rød lysdiode
    digitalWrite(ledPinYellow, LOW);       // Slå av gul lysdiode
    digitalWrite(ledPinGreen, HIGH);       // Slå av grønn lysdiode
    delay(3000);
    for (int i = 0; i < 5; i++)
    {
      digitalWrite(ledPinGreen, LOW);      // Slå av grønn lysdiode
      delay(500);
      digitalWrite(ledPinGreen, HIGH);     // Slå på grønn lysdiode
      delay(500);
    }
    digitalWrite(ledPinGreen, LOW);        // Slå av grønn lysdiode
    digitalWrite(ledPinYellow, HIGH);      // Slå på gul lysdiode
    delay(1000);
    digitalWrite(ledPinYellow, LOW);       // Slå på gul lysdiode
    bestilling = false;                    // Sett bestilling til null/falsk
   }
   else
   {
     digitalWrite(ledPinRed, HIGH);        // Slå på rød lysdiode
   }
}
```

C.5.5 Løsningsforslag oppgave D

```
/*
  Trafikklys Oppgave D
  Lag et trafikklys som lyser slik som et trafikklys skal lyse
  når det skifter fra rødt til grønt og tilbake.
  Skifte skal imidlertid kun inntreffe ved trykk på knappen. Fra nå av skal
  det grønne lyset lyse i 8 sek. Hvorav de siste skal blinke samtidig som det
  gis lyd
*/

int ledPinRed =  13;                       // Kontakt rød LED til pinne pin 13
int ledPinYellow =  12;                    // Kontakt rød LED til pinne pin 12
int ledPinGreen =  11;                     // Kontakt rød LED til pinne pin 11
int knappBestilling = 10;                  // Kontakt knapp for bestilling av grønn mann
```

```
int tonePin = 9;                          // Kontakt for tilkobling av piezoelektrisk lydkilde
boolean bestilling = false;               // Boolsk variabel som holde bestilling


// Setup definerer hva som er utganger og hva som er innganger

void setup()  {
  pinMode(ledPinRed, OUTPUT);
  pinMode(ledPinYellow, OUTPUT);
  pinMode(ledPinGreen, OUTPUT);
  pinMode(tonePin, OUTPUT);
  pinMode(knappBestilling, INPUT);
}

// loop() kjører om og om igjen i det uendelige,

void loop()
{

  bestilling = digitalRead (knappBestilling);      //Les av knappens innstilling

  if (bestilling == true)
   {
    digitalWrite(ledPinRed, HIGH);                 // Slå på rød lysdiode
    delay(5000);
    digitalWrite(ledPinYellow, HIGH);              // Slå på gul lysdiode
    delay(1000);
    digitalWrite(ledPinRed, LOW);                  // Slå av rød lysdiode
    digitalWrite(ledPinYellow, LOW);               // Slå av gul lysdiode
    digitalWrite(ledPinGreen, HIGH);               // Slå av grønn lysdiode
    delay(3000);
    for (int i = 0; i < 5; i++)
    {
      digitalWrite(ledPinGreen, LOW);              // Slå av grønn lysdiode
      Lyd(100);                                    // Gi lydstøt på 100ms
      delay(500);
      digitalWrite(ledPinGreen, HIGH);
      Lyd(100);                                    // Gi lydstøt på 100ms
```

```
    delay(500);
   }
   digitalWrite(ledPinGreen, LOW);          // Slå av grønn lysdiode
   digitalWrite(ledPinYellow, HIGH);        // Slå på gul lysdiode
   delay(1000);
   digitalWrite(ledPinYellow, LOW);         // Slå på gul lysdiode
   bestilling = false;                      // Sett bestilling til null/falsk
  }
  else
  {
   digitalWrite(ledPinRed, HIGH);           // Slå på rød lysdiode
  }
}

void Lyd(int varighet)
{
 tone(tonePin, 1500, varighet);             // Angir pinnenummer for piezoelektrisk lydgiver,
                                            // frekvens og varighet i msek

}
```

C.5.6 Løsningsforslag oppgave E

```
/*
 Trafikklys Oppgave E
 Lag et trafikklys som lyser slik som et trafikklys skal lyse
 når det skifter fra rødt til grønt og tilbake.
 Skifte skal imidlertid kun inntreffe ved trykk på knappen. Fra nå av skal
 det grønne lyset lyse i 8 sek. Hvorav de første 3 skal gi lange støtt og de siste 5
 skal blinke samtidig som det gis korte lydstøt (100 ms). I tillegg skal lyset gå over i blinkende
 gult når mørket faller på
*/

int ledPinRed =  13;       // Kontakt rød LED til pinne pin 13
int ledPinYellow =  12;    // Kontakt rød LED til pinne pin 12
int ledPinGreen =  11;     // Kontakt rød LED til pinne pin 11
int knappBestilling = 10;  // Kontakt knapp for bestilling av grønn mann
int tonePin = 9;           // Kontakt for tilkobling av piezoelektrisk lydkilde
int blinkendeGult = 0;     // Kontakt for tilkobling av LDR
```

```
int lysintensitet;          // Angir hvor lyst det er i ved trafikklyset
boolean bestilling = false; // Boolsk variabel som holde bestilling


// Setup definerer hva som er utganger og hva som er innganger

void setup()  {
  pinMode(ledPinRed, OUTPUT);
  pinMode(ledPinYellow, OUTPUT);
  pinMode(ledPinGreen, OUTPUT);
  pinMode(tonePin, OUTPUT);
  pinMode(knappBestilling, INPUT);
  pinMode(blinkendeGult, INPUT);
}

// loop() kjører om og om igjen i det uendelige,

void loop()
{

  lysintensitet = analogRead(blinkendeGult);  //Leser lysintensiteten på LDR
  bestilling = digitalRead (knappBestilling); //Les av knappens innstilling

  if (bestilling == true)
   {
    digitalWrite(ledPinRed, HIGH);                  // Slå på rød lysdiode
    delay(5000);
    digitalWrite(ledPinYellow, HIGH);               // Slå på gul lysdiode
    delay(1000);
    digitalWrite(ledPinRed, LOW);                   // Slå av rød lysdiode
    digitalWrite(ledPinYellow, LOW);                // Slå av gul lysdiode
    for (int i = 0; i < 4; i++)                     // Kontinuerlig grønt med lange lydstøt (500 ms)
    {
      digitalWrite(ledPinGreen, HIGH);              // Slå av grønn lysdiode
      Lyd(500);                                     // Gi lydstøt på 100ms
      delay(1000);
    }
    for (int i = 0; i < 5; i++)                     // Blinknde grønt med korte lydstøt (100 ms)
```

```
  {
    digitalWrite(ledPinGreen, LOW);          // Slå av grønn lysdiode
    Lyd(100);                                // Gi lydstøt på 100 ms
    delay(500);
    digitalWrite(ledPinGreen, HIGH);
    Lyd(100);                                // Gi lydstøt på 100 ms
    delay(500);
  }
  digitalWrite(ledPinGreen, LOW);            // Slå av grønn lysdiode
  digitalWrite(ledPinYellow, HIGH);          // Slå på gul lysdiode
  delay(1000);
  digitalWrite(ledPinYellow, LOW);           // Slå på gul lysdiode
  bestilling = false;                        // Sett bestilling til null/falsk
 }
 else
 {
  if ((lysintensitet > 400) && (bestilling == false))
  {
    digitalWrite(ledPinRed, LOW);            // Slå på rød lysdiode
    digitalWrite(ledPinYellow, HIGH);        // Slår på gult
    delay(500);
    digitalWrite(ledPinYellow, LOW);         // Slår av gult
    delay(500);
  }
  else
  {
    digitalWrite(ledPinRed, HIGH);           // Slå på rød lysdiode
  }
 }
}

void Lyd(int varighet)
{
 tone(tonePin, 1500, varighet);              // Angir pinnenummer for piezoelektrisk lydgiver,
                                             // frekvens og varighet i msek
}
```

C.6 Løsningsforslag kolorimeter (turbidimeter)

Når det benyttes melk i løsningen vil fettpartiklene i melken ikke absorbere men spre lyset. Ett slikt instrument kalles et *turbidimeter* og ikke et kolorimeter.

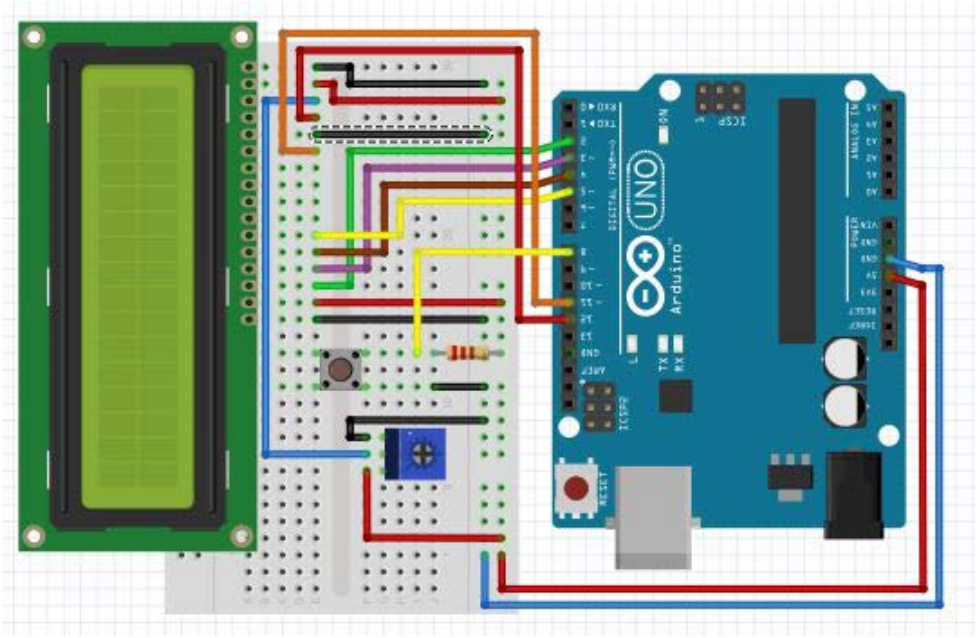
C.6.1 Kode for kolorimeter

```
/* Oppgave 6 - Kolorimeter */

#include <LiquidCrystal.h>

LiquidCrystal lcd(12,11,5,4,3,2);

int lysIntensitet = 0;
float sumIntensitet = 0;
float Intensitet = 0;
int pinHvit = 10;

void setup()
{
 // Setup LCD display
 lcd.begin(16, 2);
 lcd.clear();
 pinMode(pinHvit, OUTPUT);
}
void loop()
{
 digitalWrite(pinHvit, HIGH); // Slå på lyskilde
 delay(10);
 int i;
 // Gjør en midlet måling
 for (i = 0; i < 500; i++)
 {
  lysIntensitet = analogRead(0);
```

```
    sumIntensitet = sumIntensitet + lysIntensitet;
  }
    digitalWrite(pinHvit, LOW);   // Slå av lyskilde

    Intensitet = sumIntensitet/i; // Beregn middelverdien av måleserien
    sumIntensitet = 0;            // Nullstill summevariabel

    // Skriv resultatet til display
    lcd.clear();
    lcd.print("Lysintensitet");
    lcd.setCursor(0,1);
    lcd.print(Intensitet);
    delay(500);
}
```

C.7 Løsningsforslag for oppgaven “Lag en stoppeklokke”

C.7.1 Forslag til oppkobling



C.7.2 Forslag til kode

Koden er for Spesifikasjon D:

```
/*
Stoppeklokke Nils Kr. Rossing 12.12.14
*/

#include <LiquidCrystal.h>

LiquidCrystal lcd(12,11,5,4,3,2);

long int subSek = 0;
long int startSubSek = 0;
long int sekunder = 0;
long int startSek = 0;
long int minutter = 0;
long int startMin = 0;
      int startFlag = 0;
      int stopFlag = 1;
      int knapp = 1;

void setup()
{
 lcd.begin(16, 2);
 lcd.clear();
 lcd.print("Klar til start? Trykk knapp");
}

void loop()
{
knapp = digitalRead (8);        // Les av trykknappen
if (startFlag == 0)             // Går klokka eller står den?
```

```
{
  while (knapp == 0)              // Stopp programmet ved ved trykk
  {
    knapp = digitalRead (8);
    startSubSek = millis()/10;     // Les av millisek. ved oppstart etter trykk
    startFlag = 1;
    subSek = 0;
    sekunder = 0;
    minutter = 0;
    lcd.clear();
  }
}
else if (startFlag == 1)          // Aksjon når programmet går og knappen trykkes
{
  while (knapp == 0)              // Aksjon når knappen trykkes mens programmet gåt
  {
    knapp = digitalRead (8);
    startFlag = 0;
  }
}

if (startFlag == 1)               // Måler tiden når knappen har vært trykket
{
  subSek = millis()/10 - startSubSek;  // Definer starttiden i forhold til millis()

  if (subSek >= 100)              // Nullstill hundredeler
  {
    startSubSek = startSubSek + 100;
    subSek = 0;
    sekunder++;
    lcd.clear();
  }
```

```
  if (sekunder >= 60)            // Nullstill sekunder
  {
   sekunder = 0;
   minutter++;
   lcd.clear();
  }

  if (minutter >= 60)            // Nullstill minutter
  {
  minutter = 0;
  lcd.clear();
  }

  lcd.setCursor(0,1);            // Skriv minutter til displayet
  lcd.print(minutter);

  lcd.setCursor(3,1);            // Skriv sekunder til displayet
  lcd.print(sekunder);

  lcd.setCursor(6,1);            // Skriv hundredeler til displayet
  lcd.print(subSek);
}
}
```

Vedlegg D Løsningsforslag til oppgaver til Mr. Abot

D.1 Styring av Mr. Abot med laserpeker

```
// Programfor styring av Mr. Abot ved hjelp av laserpeker Nils Kr. Rossing 27.01.2014

#include <Servo.h>

Servo myleftservo;          // Definer venstre servomotor
Servo myrightservo;         // Definer høyre servomotor

int solcelle_H = 0;         // Høyre solcelle sett forfra
int solcelle_M = 0;         // Midtre solcelle
int solcelle_V = 0;         // Venstre solcelle sett forfra

int terskel_H;
int terskel_M;
int terskel_V;

int right_start_point = 1583; // Verdi der høyre servo starter
int left_start_point = 1472;  // Verdi der venstre servo starter
int right_speed = 200;        // Høyre servos hastighet
int left_speed = 200;         // Vesntre servos hastighet

void setup()
{
  pinMode(12, OUTPUT);
  pinMode(13, OUTPUT);

  myleftservo.attach(12);     // Attaches the left servo on pin 12 to the servo object
  myrightservo.attach(13);    // Attaches the right servo on pin 13 to the servo object

  Serial.begin(9600);
```

```
  terskel_H = analogRead(A5) + 10; // Les av bakgrunnlys og bestem høyre terskelnivå
  terskel_M = analogRead(A4) + 10; // Les av bakgrunnlys og bestem midtre terskelnivå
  terskel_V = analogRead(A3) + 10; // Les av bakgrunnlys og bestem venstre terskelnivå

}


void loop()
{

  solcelle_H = analogRead(A5);    // Les av lysnivået for høyre solcelle
  solcelle_M = analogRead(A4);    // Les av lysnivået for midterste solcelle
  solcelle_V = analogRead(A3);    // Les av lysnivået for venstre solcelle

  /*Serial.print("terskel_V: ");  // Mnitorering av terskelnivåene
  Serial.print(terskel_V);
  Serial.print(" terskel_M: ");
  Serial.print(terskel_M);
  Serial.print(" terskel_H: ");
  Serial.println(terskel_H);*/

  if (solcelle_H > terskel_H) // Test om høyre solcelle belyses
   {
   myrightservo.writeMicroseconds(right_start_point + 60);
   myleftservo.writeMicroseconds(1500);
   }
  else if (solcelle_M > terskel_M) // Test om midtre solcelle belyses
   {
   myrightservo.writeMicroseconds(right_start_point + right_speed);
   myleftservo.writeMicroseconds(left_start_point - left_speed);
   }
```

```
  else if (solcelle_V > terskel_V) // Test om venstre solcelle belyses
  {
   myrightservo.writeMicroseconds(1500);
   myleftservo.writeMicroseconds(left_start_point - 200);
  }
 else
  {
   myleftservo.writeMicroseconds(1500);  // Stopp venstre motor
   myrightservo.writeMicroseconds(1500); // Stopp høyre motor
  }

}
```

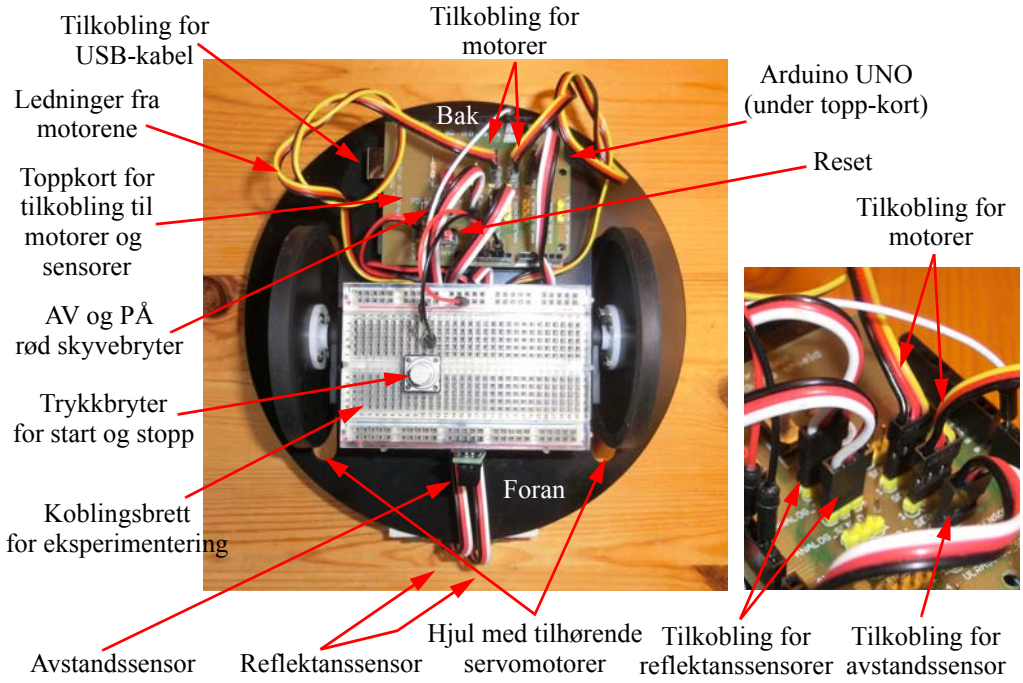
Vedlegg E Informasjons-ark

E.1 Kort presentasjon av Mr. Abot

Disse informasjon-arkene brukes som underlag for en kort presentasjon av Mr. Abot for lærere eller ansatte ved vitensentere uten at det kreves mye bakgrunn. De vil også bli brukt som fakta- og oppgaveark for deltagerne som skal løse oppdragene.

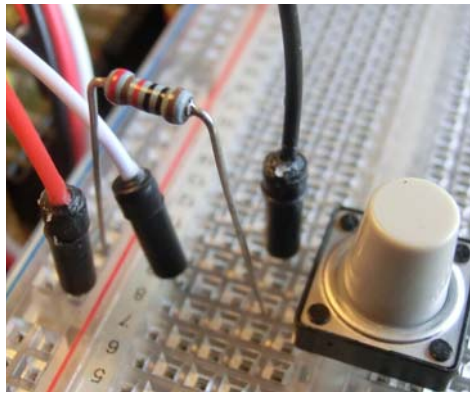
E.1.1 Oppbygging og montering av Mr. Abot

Figuren under viser Mr. Abot sett ovenfra:



1. **Koble til servomotorene:**
   Høyre servomotor –> *Servo 1*
   Venstre servomotor –> *Servo 2*

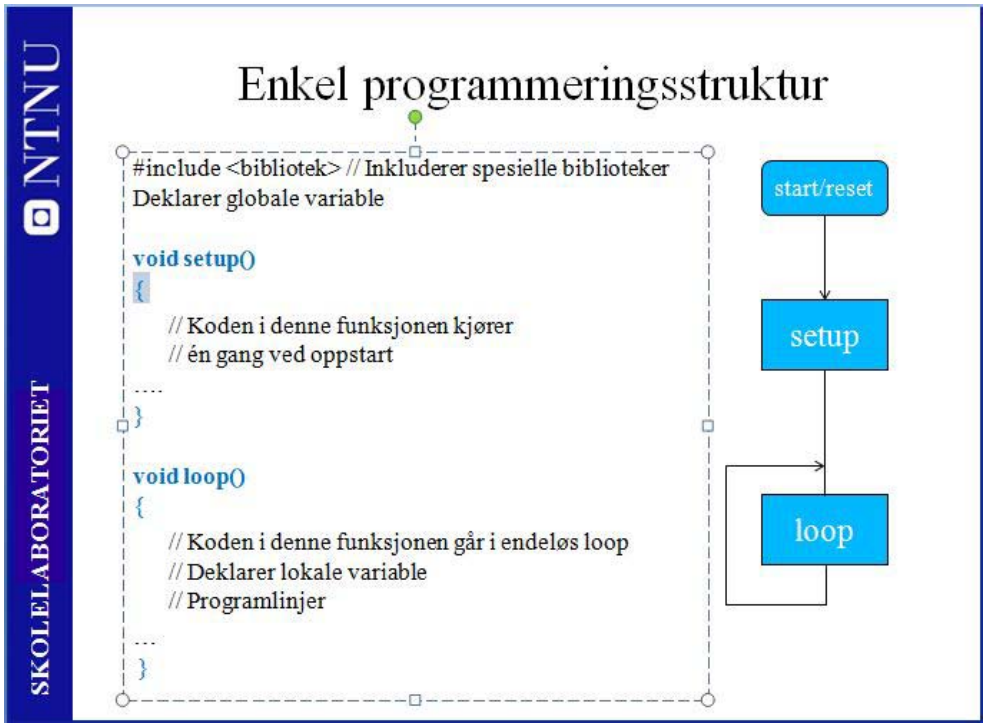
2. **Koble til reflektansensorene**
   Høyre reflektanssensor –> *Analog_sensor_0*
   Venstre reflektanssensor –> *Analog_sensor_1*
3. **Koble til avstandssensor**
   Avstandssensor –> *Digital sensor*
4. **Koble opp bryter**
   Trykkbryter, motstand (10 kOhm) og ledninger kobles opp som vist
   Sort ledning –> *Jord (GND)*
   Rød ledning –> *+ 5V*
   Hvit ledning –> *digital inngang 7*



Sorte ledninger skal alltid til kontakter merket 1

E.1.2 Grunnleggende Programstruktur

Dette arket viser den grunnleggende programstrukturen for et typisk Arduino program.



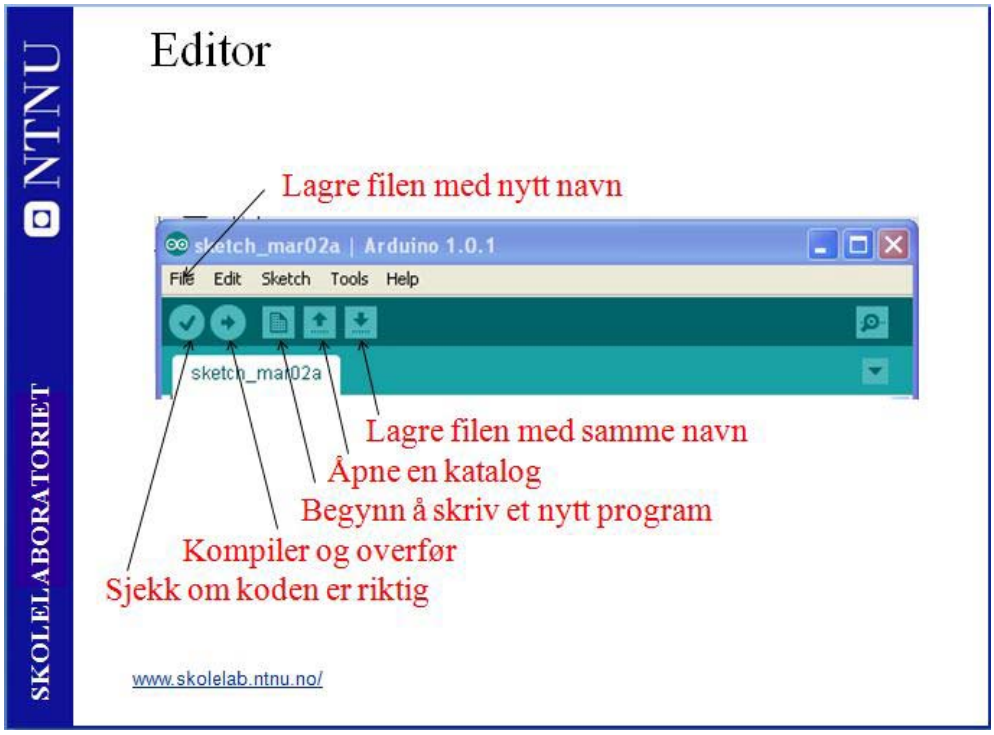
Start: Her legges:
- # funksjonsbiblioteker
- Deklarering av globale variable

Setup: Her legges:
- de kommandoene som bare skal utføres en gang i starten av hver kjøring

Loop: Her legges:
- de kommandoene som skal gjentas i en endeløs loop

E.1.3 Programeditoren

Programmet skrives i *programeditoren*. Når vi er ferdige, oversettes programkoden til en kode som Arduino-kortet forstår. Vi sier at programmet *kompileres*. Så snart programmet er overført til kortet, begynner det å utføre kommandoene i programmet.

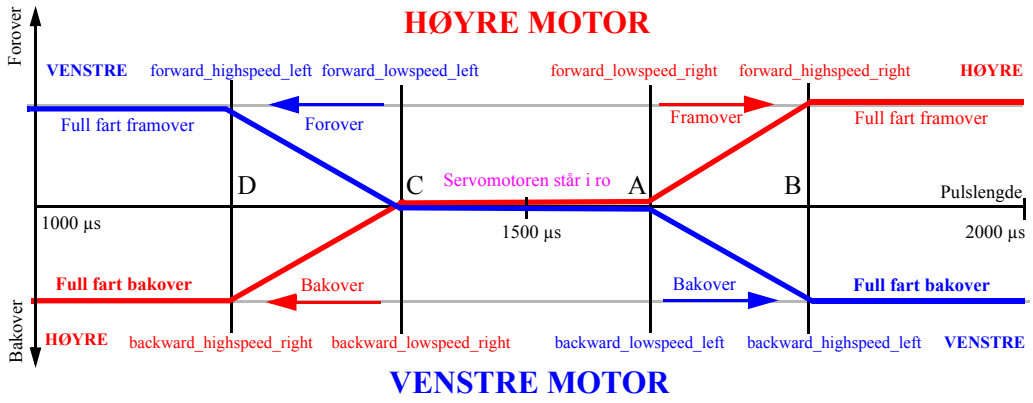


Gangen i programmering:

1. Skrivprogrammet
2. Sjekk om det er feil programkoden
3. Lagre programfilen
4. Overfør programmet til Arduino-kortet

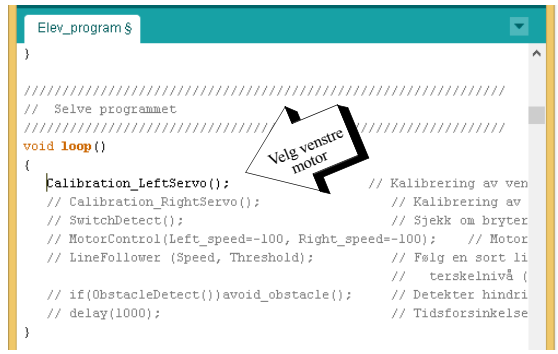
E.1.4 Kalibrering av motorene

Hastigheten til en servomotor styres av et tog av pulser. Lengden av pulsene bestemmer farten. Pulslengden er fra ca. 1000 µs (full fart bakover) til 2000 µs (full fart forover). I et område omkring 1500 µs står motoren stille. Som det framgår av figuren så kan farten bare endres i to områder, (A–B) og (C–D). Legg merke til at de to motorene må gå motsatt vei for å gå bent:



Verdiene i *knekkpunktene* A, B, C og D, er heller ikke like fra motor til motor. Vi må derfor finne disse fire grenseverdiene for høyre og venstre motor, og bruke dem til å kalibrere programmet som styrer roboten:

forward_lowspeed (A)
forward_highspeed (B)
backward_lowspeed (C)
backward_highspeed (D)



Slik bestemmes knekkpunktene:

1. Last opp programmet: *elev_program*
2. Gå til: void **loop**() og velg ***Calibration_Left.Servo();*** ved å fjerne // foran.
3. Overfør programmet til roboten ved å velge
4. Åpne monitorvinduet
5. Skriv inn verdier fra 1000 til 2000 på ***kommandolinjen*** og velg **Send**.

Punktet ***lowspeed*** er der motoren akkurat begynner å gå.
Punktet ***highspeed*** er det punktet hvor man akkurat når maks. fart. For å finne maks. punktet, er det enklest å høre om lyden i motoren forandres fra toppfart og ned til dette punktet.

6. Finn knekkpunktene for venstre servomotor:

   Forward_highspeed_left: _____
   Forward_lowspeed_left: _____
   Backward_lowspeed_left: _____
   Backward_highspeed_left: _____

7. Velg ***Calibration_RightServo();*** i programmet ved å fjerne //. Husk å sette // foran Calibration_LeftServo();

8. Skriv inn verdier fra 1000 til 2000 på ***kommandolinjen*** og velg **Send**. Finn knekkpunktene som for den venstre motoren.

   Forward_highspeed_right: _____
   Forward_lowspeed_right: _____
   Backward_lowspeed_right: _____
   Backward_highspeed_right: _____

   Husk å sette // foran Calibration_RightServo(); når du er ferdig.

9. Gå så lengre fram i programmet, der hvor verdiene for knekkpunktene skal legges inn. Se figuren til høyre.

10. Skriv inn de nye verdiene og lagre programmet.


```
int forward_highspeed_left   = 1390;  //value when left servo is st
int forward_lowspeed_left    = 1490;  //value when left servo stopp
int backward_lowspeed_left   = 1612;  //value when left servo stopp
int backward_highspeed_left  = 1695;  //value when left servo is st

int forward_highspeed_right  = 1625;  //value when right servo is s
int forward_lowspeed_right   = 1552;  //value when right servo stop
int backward_lowspeed_right  = 1433;  //value when right servo stop
int backward_highspeed_right = 1340;  //value when right servo is s
```

E.1.5 Oversikt over funksjoner

Programmet består av en rekke funksjoner som kan kalles opp innenfor programloopen (void loop();). Disse velges inn etter behov ved å fjerne // foran.

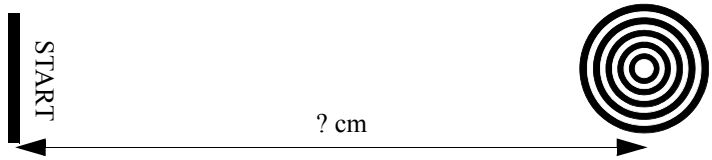
```
//////////////////////////////////////////////////////////////
//  Selve programmet
//////////////////////////////////////////////////////////////
void loop()
{
   // Calibration_LeftServo();                    // Kalibrering
   // Calibration_RightServo();                   // Kalibrering
   // SwitchDetect();                             // Sjekk om bry
   // MotorControl(Left_speed=100, Right_speed=100);// Bestem farte
   // delay(10000);                               // Tidsforsinke
   // LineFollower (Speed = 50, Threshold = 300);  // Følg en sort
                                                  //   terskelniv
   // if(obstacle_detect()){avoid_obstacle();}    // Detekter hin
   // stop_running();                             // Stopp robote
}
```

Følgende funksjoner ligger klare for å velges inn:

- ***Calibration_LeftServo();***
  For kalibrering av venstre servomotor.
- ***Calibration_RightServo();***
  For kalibrering av høyre servomotor.
- ***SwitchDetect();***
  Programmet starter når man trykker på knappen på koblingsbrettet, og stopper når man trykker en gang til.
- ***MotorControl(Left_Speed*** = <venstre verdi>***, Right_Speed*** = <høyre verdi>***);***
  Funksjonen setter farten på hver av servomotorene fra:
  <venstre/høyre verdi> = 1 - 100 –> venstre/høyre servo, sett fart *forover*
  <venstre/høyre verdi> = –1 - –100 –> venstre/høyre servo, sett fart *bakover*
  <venstre/høyre verdi> = 0 (full stopp)
  <venstre verdi> = (1 - 100), <høyre verdi> = –(1 - 100) –> sving mot høyre
  <venstre verdi> = –(1 - 100), <høyre verdi> = (1 - 100) –> sving mot venstre
- ***LineFollower (Speed*** = <fart verdi>***, Threshold*** = <terskelverdi>***);***
  Følg kanten av svart linje.
  <fart verdi> = 1 - 100 –> hastigheten til roboten
  <terskelverdi> = 0 - 1023 –> terskel for å skille mellom lyse (0) og mørke (1023) felt
- ***delay (***<millisekunder>***);***
  Forsinker programmet i et antall millisekunder gitt av argumentet i funksjonen.
- ***stop_running ();***
  Stopper programmet (og roboten) til man trykker reset eller slår AV og PÅ strømmen.
- ***obstacle_detect ();***
  Denne funksjonen detekterer hindringen og går til funksjonen ***avoid_obstacle ();*** slik at roboten kan bevege seg rundt og bak hindringen.
- ***avoid_obstacle ();***
  Funksjonen må programmeres i henhold til størrelsen på hindringen.
- ***if (obstacle_detect ()) avoid_obstacle();***
  Hvis avstandssensoren oppdager en hindring, vil funksjonen ***obstacle_detect ()*** returnere at en hindring er oppdaget (dvs. funksjonen returnerer true = 1). Når det er tilfelle vil funksjonen ***avoid_obstacle ();*** bli kalt opp slik at roboten går rundt og forbi. På baksiden må den på nytt lete etter den svarte stripen.

E.1.6 Oppgave 1 - Nærmest målet i rett linje

A - Nærmest målet:



Oppgaven går ut på å bli kjent med roboten og finne ut hvor langt den går pr. tidsenhet. Oppgaven avsluttes med en konkurranse hvor dere fra et startpunkt skal komme nærmest et mål. Avstanden får dere oppgitt 5 min. før konkurransen starter.

1. **Kalibrer motorene** om ikke det alt er gjort, se informasjons-arket "Kalibrering av motorene"
   Bruk funksjonene:
   Calibration_LeftServo(); og ***Calibration_RightServo();*** for å kalibrere servomotorene
   SwitchDetect(); slik at dere kan starte og stoppe roboten med knappen
   Kommenter bort funksjonene når dere er ferdige

2. **Start og kjør servomotorene**
   Bruk funksjonen:
   MotorControl(Left_speed = <venstre verdi>***, Right_speed*** = <høyre verdi>***);***
   til å sette fart på høyre og venstre servomotor.
   <høyre/venstre verdi> kan ha verdier fra –100 til +100
   Funksjonen kan også brukes når roboten skal svinge til høyre og venstre:
   Se informasjons-arket "Oversikt over funksjoner".

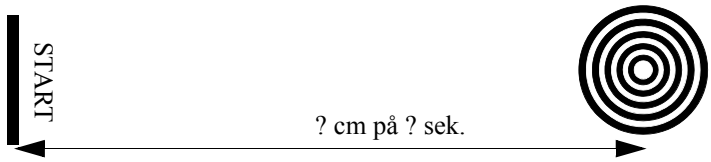
3. **Kjøretid:**
   Bruk funksjonen:
   delay(<millisekunder>***);***
   Med denne funksjonen bestemmer dere hvor lenge foregående funksjon skal være aktiv før neste funksjon utføres.

4. **Start og stopp** av programmet
   Bruk funksjonen:
   SwitchDetect();
   For manuell start og stopp av programmet
   stop_running ();
   Stopper programmet. Programmet startes på nytt ved å slå av og på strømmen eller trykke "Reset" (en liten rød knapp på topp-kortet).

E.1.7 Oppgave 1B - Nærmest målet i rett linje og på oppgitt tid

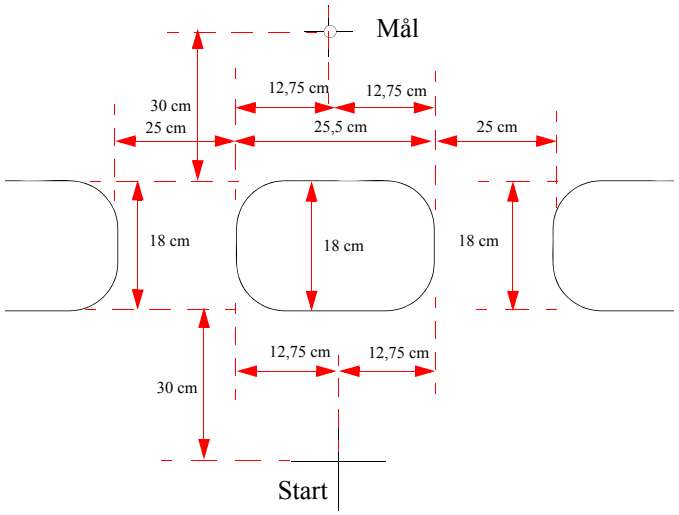
B - Nærmest målet på angitt tid:

Oppgave 1B er som oppgave 1A bare at nå skal dere komme nærmest målet innen en bestemt tid. Tid og ny avstand blir oppgitt 5 minutter før konkurransen.



E.1.8 Oppgave 2 - Nærmest målet bak hindring

Oppgaven går ut på å komme nærmest et mål som er skjult bak et hinder. Figuren under viser målene på banen som skal forseres.



1. **Start og kjør servomotorene**
   Bruk funksjonen:
   MotorControl(Left_speed=-100, Right_speed=-100);
   til å sette fart på høyre og venstre servomotor.
   Funksjonen kan også brukes når roboten skal svinge til høyre og venstre:
   Se informasjons-ark "Oversikt over funksjoner"

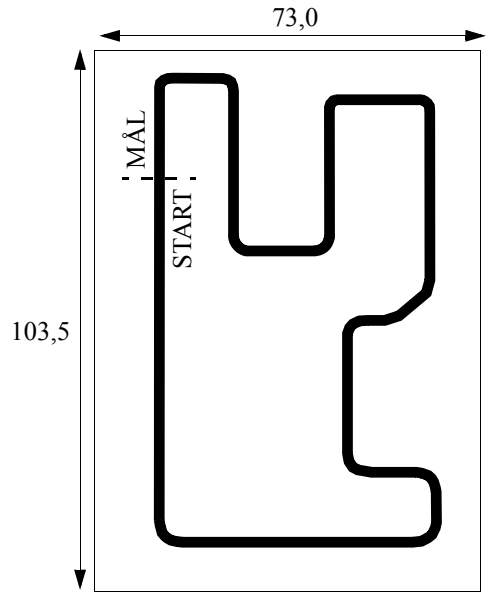
2. **Kjøretid:**
   Bruk funksjonen:
   delay(<millisekunder>***);***
   Med denne funksjonen bestemmer dere hvor lenge foregående funksjon skal være aktiv før neste funksjon utføres.

3. **Start og stopp** av programmet
   Bruk funksjonen:
   SwitchDetect();
   For manuell start og stopp av programmet
   stop_running ();
   Stopper programmet. Programmet startes på nytt ved å slå av og på strømmen eller trykke "Reset" (en liten rød knapp på topp-kortet).

E.1.9 Oppgave 3A - Følg den svarte linjen - Raskest rundt

A - Raskest rundt

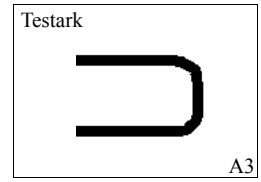
Oppgaven går ut på å følge den svarte linjen banen rundt på kortest mulig tid. Figuren til høyre viser formen på banen.



Følgende funksjoner vil være nyttige å bruke i denne oppgaven:

1. ***LineFollower (Speed*** = <fart verdi>***, Threshold*** = <terskel verdi>***);***
   Følg kanten av svart linje.
   <fart verdi> = 1 - 100 –> hastighet
   <terskel verdi> = 0 - 1023 –> terskel for å skille mellom lyse (0) og mørke (1023) felt
2. ***SwitchDetect();***
   Programmet starter når man trykker på knappen på koblingsbrettet, og stopper når man trykker en gang til.

Tips: Bruk *testarket* for å teste ut sammenhengen mellom fart og robotens evne til å bevege seg rundt svingene.



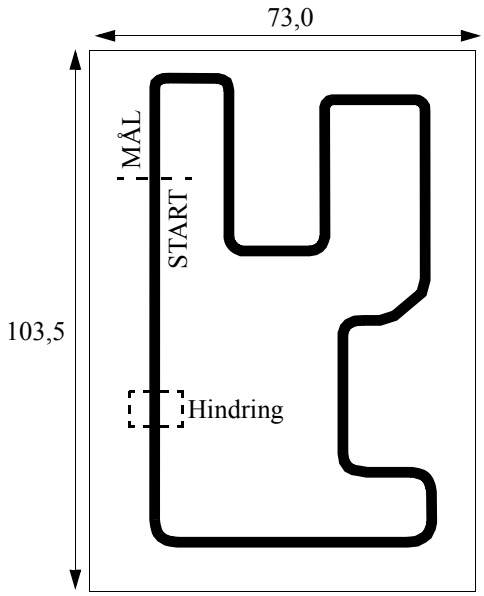
E.1.10 Oppgave 3B - Følg den svarte linjen - Raskest rundt og unngå hindring

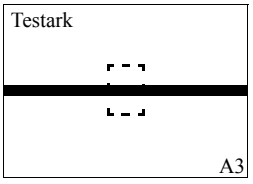
B - Raskest rundt og unngå hindring

Oppgaven går ut på å bevege seg raskest mulig langs den svarte linjen fra START til MÅL, detektere og omgå hindringen:

Følgende tilleggsfunksjoner vil være nyttige å bruke i denne oppgaven:

3. ***avoid_obstacle ();***
   Funksjonen må programmeres i henhold til størrelsen på hindringen som bør være kjent.
4. ***obstacle_detect ();***
   Denne funksjonen detekterer hindringen og går til funksjonen *avoid_obstacle ();* slik at en kommer seg rundt og bak hindringen.
5. ***if (obstacle_detect ()) avoid_obstacle();***
   Hvis avstandssensoren oppdager en hindring, vil funksjonen ***obstacle_detect ();*** returnere at en hindring er oppdaget (dvs. funksjonen returnerer true = 1). Når det er tilfelle vil funksjonen ***avoid_obstacle ();*** bli kalt opp slik at roboten går rundt og forbi. På baksiden må den på nytt lete etter den svarte linjen.





E.1.11 Programmet

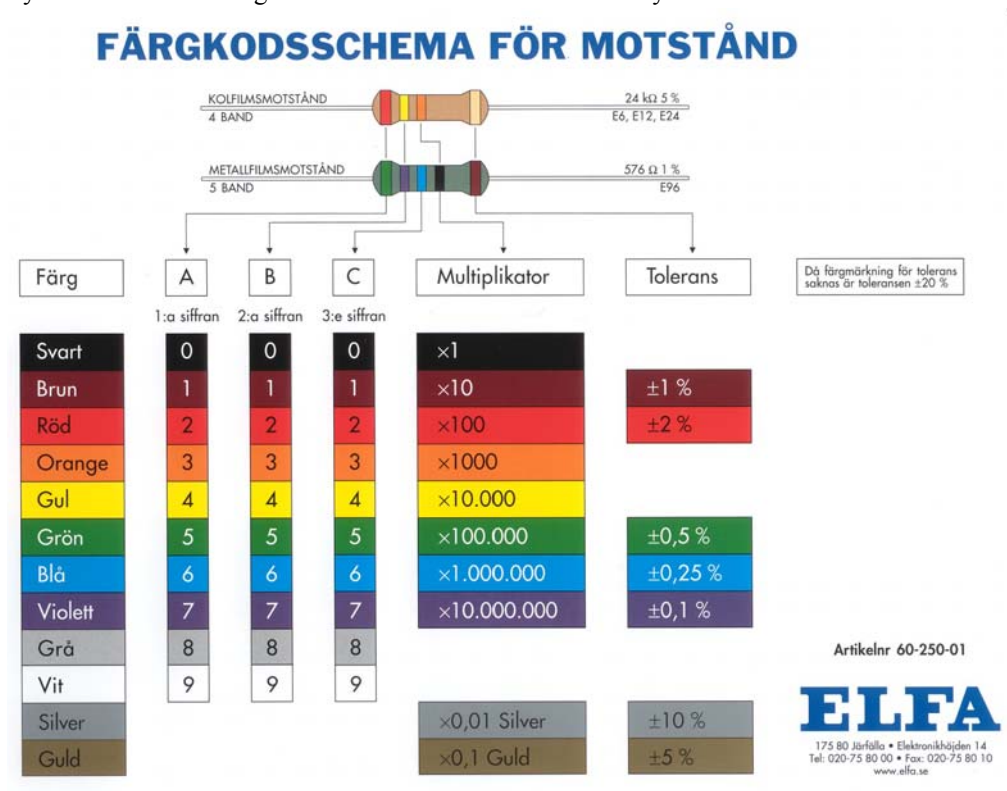
Koden under viser programmet som skal brukes for å programmerer robotens bevegelse. De enkelte kommandoene er forklart under programlistingen.

```
// Program for running Mr. Abot along a strait line

#include <Servo.h>

Servo myleftservo;                      // Definer venstre servomotor
Servo myrightservo;                     // Definer høyre servomotor

int run = 0;
int right_start_point = 1583;           // Value where the right servo start running
int left_start_point = 1472;            // Value where the left servo start running
int right_speed = 60;                   // Right servo speed forward
int left_speed = 60;                    // Left servo speed forward
float running_time = 1000;              // Running time i msek

void setup()
{
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);

  myleftservo.attach(10);               // Attaches the left servo to pin 9
  myrightservo.attach(9);               // Attaches the right servo to pin 10
}

void loop()
{

  myleftservo.writeMicroseconds(1500);  // set left servo to move forward
  myrightservo.writeMicroseconds(1500); // set left servo to move forward
```

```
  delay(2000);                                  // Start running after 2000 msek.

  myrightservo.writeMicroseconds(right_start_point + right_speed); // Right servo move forward
  myleftservo.writeMicroseconds(left_start_point - left_speed);       // Left servo move forward

  delay(running_time);                          // Set running time.

  myleftservo.writeMicroseconds(1500);      // Set left servo to stop
  myrightservo.writeMicroseconds(1500);     // Set right servo to stop

  while(true);                                  // Stop running the program for ever

}
```

Vedlegg F Fargekoding av motstander

Motstander er ofte merket med fargede ringer som angir resistansen eller verdien til motstanden. Antallet ringer er avhengig av hvor nøyaktig resistansen er. Nøyaktigheten angis som en toleranse i prosent. Dvs. at verdien skal garantert være innenfor et angitt prosentvist avvik. Ofte angis toleransen ved hjelp av en gull- eller sølvfarget ring. En sølvfarget ring antyder at toleransen er innen +/-10%, en gullfarget innen +/-5%. Også andre farger kan brukes for mer nøyaktige motstander med mindre toleranse. Når vi leser verdien til motstanden skal gull- eller sølvringen være til høyre. De resterende ringene leses deretter fra venstre mot høyre.



Plansjen over viser sammenhengen mellom farge og tall.

Vedlegg G Læreplaner

G.1 Teknologi og Forskningslære 1

Fargekodene forsøker å antyde hva som ***lett kan*** knyttes til CanSat og romteknologi (rødt/kursiv), hva som er litt på siden (blått) og det som vanskelig lar seg oppfylle innen prosjektet (sort).

Den unge ingeniøren - Beskrivelse av hovedområde

Mål for opplæringen er at eleven skal kunne:

- *planlegge og bygge en konstruksjon som er fast eller bevegelig, og som har en definert funksjon*
- bruke tredimensjonale tegninger eller skisser i utvikling av konstruksjoner
- bruke forskjellige materialer og former for sammenføyninger og begrunne valg av materialer og byggemåte ut fra materialenes egenskaper og konstruksjonens funksjon
- *bruke sensorer og styringssystemer i forbindelse med forsøk og konstruksjoner*
- *dokumentere og vurdere konstruksjoners fysiske egenskaper og funksjonalitet ved hjelp av målinger og enkle beregninger*

Den unge forskeren - Beskrivelse av hovedområde

Mål for opplæringen er at eleven skal kunne:

- *gjøre rede for hvordan et naturvitenskapelig prosjekt planlegges, gjennomføres og etterarbeides før det blir publisert (planlegge målinger utført i løpet av droppet)*
- *planlegge, gjennomføre, analysere og dokumentere systematiske målinger*
- *om støy, luftforurensning, inneklima og vannkvalitet, og drøfte virkninger på helse og miljø*

Teknologi, naturvitenskap og samfunn - Beskrivelse av hovedområde

Mål for opplæringen er at eleven skal kunne

- drøfte etiske, miljømessige, kulturelle og politiske sider ved teknologisk utvikling
- *beskrive den historiske utviklingen av en teknologisk innretning, forklare virkemåten og drøfte anvendelser i samfunnet*
- gjøre rede for utvikling og produksjon av et teknologisk produkt og vurdere produktets brukervennlighet, utviklingsmuligheter og miljøpåvirkning
- *beskrive prinsipper og virkemåte for noen moderne instrumenter i industri, helsevesen eller forskning, og gjøre rede for nytten og eventuelle skadevirkninger*
- kartlegge og presentere praktisk bruk av realfag i en lokal bedrift eller institusjon

Design og produktutvikling - Beskrivelse av hovedområde

Mål for opplæringen er at eleven skal kunne

- *gjøre rede for funksjonen til vanlige komponenter i elektroniske kretser, og gjenkjenne komponentene i en krets*
- *lage elektroniske kretser ved å lodde komponenter og simulere og teste kretsene*
- *forme og utvikle produkter som har en definert funksjon og inneholder elektronikk*
- *dokumentere og presentere designprosesser fra idé til ferdig produkt*
- *begrunne valg av materialer i produkter og vurdere produktenes form og funksjon, miljømessige konsekvenser,* estetikk og forbedringsmuligheter
- *utføre målinger med eller teste et eget produkt, og vurdere kvaliteten på produktet med tanke på funksjonalitet*

G.2 Teknologi og Forskningslære 2

Den unge forskeren - Beskrivelse av hovedområde

Mål for opplæringen er at eleven skal kunne

- gjøre rede for et forskningsprosjekt i en bedrift eller institusjon, og beskrive problemstillinger, organisering, måleutstyr, resultater og finansiering
- *planlegge og gjennomføre naturvitenskapelige undersøkelser basert på egne ideer, og presentere arbeidet i en vitenskapelig form*
- *drøfte resultater fra egne undersøkelser i forhold til relevant kunnskap på området, og vurdere hvordan kontroll av variabler og reproduserbarhet er ivaretatt*

Naturvitenskapelige arbeidsmetoder - Beskrivelse av hovedområde

Mål for opplæringen er at eleven skal kunne

- *forklare hva som menes med modell, teori og hypotese, og gjøre rede for hvordan de brukes og utvikles i forskning*
- drøfte ved å bruke eksempler hvordan empiriske data kan styrke eller forkaste en hypotese
- *gjøre rede for hvordan forskning utvikles og kvalitetssikres gjennom samarbeid, kritisk vurdering og argumentasjon*
- *gjøre rede for strukturen i en vitenskapelig publikasjon eller presentasjon*

Forskning, teknologi og samfunn - Beskrivelse av hovedområde

Mål for opplæringen er at eleven skal kunne

- beskrive kjennetegn ved grunnforskning, anvendt forskning og utviklingsarbeid og gjøre rede for hovedtrekk ved finansiering og styring

- gjøre rede for betydningen av naturvitenskapelig forskning og teknologiutvikling for næringsliv og samfunn
- drøfte økonomiske, miljømessige og etiske spørsmål i forbindelse med naturvitenskapelig forskning og teknologiutvikling
- *drøfte og gi eksempler på hvordan forskningsresultater og ny teknologi formidles og brukes av forskningsinstitusjoner, medier, bedrifter, interessegrupper og myndigheter*

Vitenskapsfilosofi og vitenskapsteori - Beskrivelse av hovedområde

Mål for opplæringen er at eleven skal kunne

- beskrive hovedtrekk i den historiske utviklingen av vitenskapelige tenkemåter og drøfte teknologiens rolle i denne utviklingen
- gjøre rede for hovedideene til noen sentrale vitenskapsteoretikere og vitenskapsfilosofer vurdere hvordan argumentasjon i aktuelle naturvitenskapelige debatter bygger på empiriske resultater, teoretisk kunnskap og ideologisk ståsted

G.3 Teknologi i Praksis (TiP) - ungdomsskolen

Formål

Valfaga skal medverke til at elevane, kvar for seg og i fellesskap, styrker lysta til å lære og opplever meistring gjennom praktisk og variert arbeid. Valfaga er tverrfaglege og skal medverke til heilskap og samanheng i opplæringa.

Teknologi handlar om den menneskeskapte verda og om innretningar og system som kan gjere kvardagen betre. Opp gjennom tidene har menneska brukt kreativitet og skaparevner til å utvikle reiskapar, maskinar og andre teknologiske produkt og løysingar. Teknologien grip inn på mange område, og har gjeve og kan gje både moglegheiter og utfordringar, både for den einskilde og for samfunnet. Innanfor teknologien finn vi dei enklaste verktøy og produkt og den mest avanserte elektronikken. Erfaring med og innsikt i teknologi kan fremje personleg utvikling, demokratisk deltaking og medverke til eit aktivt forhold til ein teknologisk kvardag.

Valfaget teknologi i praksis skal motivere elevane til å utvikle teknologiske produkt med utgangspunkt i lokale behov og problemstillingar. Prosessen frå idé til eit ferdig produkt kan medverke til skaparglede og meistringsoppleving. Gjennom eige arbeid og i samarbeid med andre kan elevane utvikle ferdigheiter og innsikt. Det inneber å prøve ut eigne talent og moglegheiter på ulike steg i prosessen, vurdere prosessar og produkt og få tilbakemeldingar frå andre.

Valfaget handlar om å planleggje, konstruere og framstille gjenstandar og produkt med varierte materiale og teknologiske løysingar. Kunnskap om teknologiske produkt som blir brukte i dagleglivet, gjev eit godt grunnlag for å forbetre produkt og utvikle nye produkt.

Valfaget hentar hovudelement frå matematikk, naturfag og kunst og handverk/duodji. Element frå norsk/samisk, RLE og samfunnsfag kan også inngå.

Hovedområder

Undersøkingar

Hovedområdet handlar om korleis teknologiske produkt er konstruerte og verkar, kva for prosessar som inngår i utvikling og bruk, og kva for behov produkta dekkjer. Utvikling, konstruksjon og produksjon av teknologi inngår i hovudområdet, i tillegg til helse, miljø og sikkerheit (HMS). Kunnskap om korleis teknologien byggjer på nokre grunnleggjande prinsipp, og korleis ny teknologi byggjer på tidlegare erfaringar, høyrer også med til hovudområdet.

Idéutvikling og produksjon

Hovudområdet omfattar planlegging, framstilling og utprøving av eigne produkt og konstruksjonar. Planar for framstilling og utprøving av eigne produkt og konstruksjonar byggjer på kravspesifikasjon.

I utviklingsfasen er kjennskap til design og verkemåte til andre produkt viktig. Diskusjon omkring ulike sider ved produkta er viktig i alle fasar av produktutviklinga og kan også medverke til å forbetre prosessar og produkt.

Kompetansemål

Undersøkingar

- undersøkje teknologiske produkt og dei vala som er gjorde med omsyn til bruk, tekniske løysingar, funksjonalitet og design
- *demonstrere riktig bruk av utvalde verktøy*
- vurdere teknologiske produkt ut frå brukartilpassing, HMS-krav og miljøtilpassing

Idéutvikling og produksjon

- *utvikle ein realistisk kravspesifikasjon for eit teknologisk produkt og beskrive kva behov produktet skal dekkje*
- *framstille produktet med eigna materiale, komponentar, og funksjonelle teknologiske løysingar*
- *bruke kunnskap om andre produkt i arbeidet med eige produkt*
- *teste eigne produkt og foreslå moglege forbetringar*

G.4 Forskning i Praksis (FiP) - Ungdomsskolen

Formål

Valgfagene skal bidra til at elevene, hver for seg og i fellesskap, styrker lysten til å lære og opplever mestring gjennom praktisk og variert arbeid. Valgfagene er tverrfaglige og skal bidra til helhet og sammenheng i opplæringen.

Forskning handler om å utvikle ny kunnskap og innsikt. Formuleringer som ”forskning viser at” eller ”vitenskapelige undersøkelser har vist at” brukes ofte og viser at tilliten til forskning er stor. Forskning skal også være kritisk, prøve ut om forskningsresultater er riktige og bidra til debatt. Erfaring med utforskning kan derfor danne grunnlag for egne meninger og kritisk tenkning, og gi elevene bedre mulighet til å forholde seg til debatt om forskning på en hensiktsmessig måte.

Opplæringen i valgfaget forskning i praksis skal bidra til at elevene får erfaring med vitenskapelige metoder og arbeidsmåter. Valgfaget skal stimulere til undring, aktiv handling for å teste ut løsninger og utvikle evnen til å stille nye spørsmål. Opplæringen skal bidra til å finne forklaringer på det som er observert, og gjennom kildegransking, eksperiment og observasjon kontrollere om forklaringene holder.

I valgfaget forskning i praksis blir elevene utfordret til å undersøke aktuelle spørsmål relatert til natur, miljø og klima, samt kultur-, samfunns- og arbeidsliv. Dette innebærer at de skal finne problemstillinger de ønsker å undersøke, vurdere mulige forklaringer, planlegge og gjennomføre undersøkelser, bruke utstyr og teknikker for datainnsamling, bearbeide data, og vurdere og formidle resultatene. Slik bidrar valgfaget med erfaringer og ferdigheter om de praktiske sidene ved forskning. Valgfaget stimulerer til nysgjerrighet og bruk av fantasi for å finne forklaringer. Det bidrar også til innsikt i hvordan etablert kunnskap og andres forskning kan trekkes inn i egen utforskning. Opplæringen skal legge til rette for at elevene får erfare hvordan det å gi og få tilbakemeldinger underveis i en utforskende prosess fremmer kvaliteten på resultatene.

Valgfaget henter hovedelementer fra naturfag, matematikk og samfunnsfag, men tema kan også hentes fra ungdomstrinnets øvrige fag.

Hovedområder

Idéutvikling

Hovedområdet omfatter de kreative sidene som inngår i alle stadier av forskningsprosessen. Utgangspunktet er en problemstilling som man lurer på og vil finne svar på. Det omfatter videre å lage gode forskningsspørsmål, få innspill fra andre, reformulere i lys av innspillene og, planlegge undersøkelser. I dette hovedområdet utvikler elevene mulige hypoteser, henter inn andre relevante data og sammenlikner og diskuterer ideer med andre elever.

Praktisk utforskning

Hovedområdet omfatter arbeid med ulike prosesser ved praktisk gjennomføring av de planlagte undersøkelsene. Det er viktig at elevene prøver ut og bruker forskjellige metoder, gjør seg kjent med og tar i bruk ulike typer utstyr og samler inn og systematiserer data. I tillegg skal elevene prøve ut tolkninger og sammenlikne egne funn med andres samt formidle resultatet.

Kompetansemål

Idéutvikling

- *finne problemstillinger, formulere forskbare spørsmål og forslag til hypoteser*
- *planlegge undersøkelser basert på egne eller gruppas forskningsspørsmål og hypoteser*
- *delta i samtaler om egen og andres utforskning*

- *argumentere for egne hypoteser i lys av funn og andre relevante undersøkelser*

Praktisk utforskning

- *gjennomføre planlagte undersøkelser og foreta relevante justeringer underveis*
- *bruke relevante metoder og utstyr for innsamling og analyse av data*
- *demonstrere og forklare metoder, virkemåten til utstyr og prosedyrer for datainnsamling i gjennomførte forskningsprosjekter*
- *systematisere data slik at mønstre kommer tydelig fram og vurdere usikkerheter*
- *formidle resultater fra egne prosjekter*

Sparkfun Inventor's kit V3

Heftet er ikke ment som noen lærebok, men som et ressurshefte for den som ønsker å anvende Arduino som et undervisningsopplegg i faget Teknologi og forskningslære (ToF) i videregående skole eller i Teknologi i praksis (TiP) i ungdomsskolen. Heftet vil gradvis bli utviklet etter som tilbudet utvides og vil både inneholde teknisk bakgrunnstoff og erfaringer fra de tiltakene som blir gjennomført. Roboten Mr. Abot er også omtalt med flere mulige oppgaver.

Nils Kr. Rossing
Førstelektor ved Skolelaboratoriet og
E-post: nils.rossing@plu.ntnu.no
Prosjektleder ved Vitensenteret
E-post: nkr@vitensenteret.com

Trykk: NTNU-trykk

NTNU
Trondheim

Program for lærerutdanning

Skolelaboratoriet
for matematikk, naturfag og teknologi

Tlf. 73 55 11 43
Faks 73 55 11 40
http://www.skolelab.ntnu.no