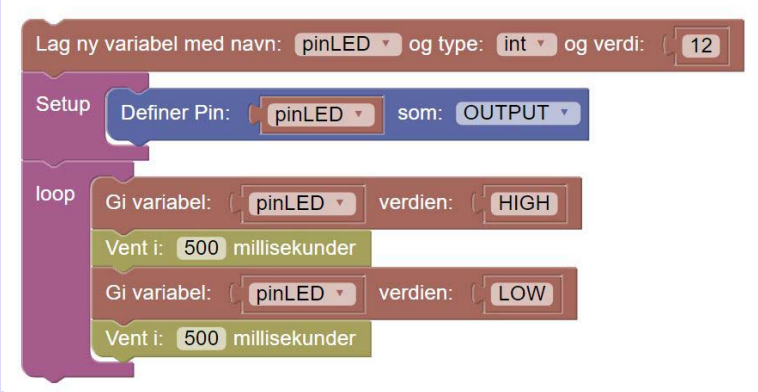
Revisjon 3.0

Nils Kr. Rossing

Arduino blokkprogrammering med fokus på Blockuino

m/løsningsforslag



NTNU
Trondheim

Institutt for lærerutdanning

Skolelaboratoriet
for matematikk, naturfag og teknologi

Mai 2017

Arduino blokkprogrammering – med fokus på Blockuino

m/løsningsforslag

Arduino blokkprogrammering – med fokus på Blockuino m/løsningsforslag

Trondheim 2017

Bidragsytere:
Nils Kr. Rossing, *(nils.rossing@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@ntnu.no

Realfagbygget, Høgskoleringen 5
7491 Trondheim

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

Prøvetrykk 3.0, Rev 3.0 – 07.05.17

Arduino blokkprogrammering – med fokus på Blockuino

Nils Kr. Rossing

Skolelaboratoriet for matematikk, naturfag og teknologi, NTNU

Forord

Dette heftet er blitt til på bakgrunn av et ønske om å tilby kurs i blokkprogrammering og da spesielt rettet mot Arduino. I dag finnes det en rekke blokkprogrammeringsvarianter for programmering av ulike mikrokontrollerkretser. Årsaken til at jeg valgte å bruke Blockuino.no er kanskje litt tilfeldig, men her er noen argumenter:

- Blockuino.no er norskutviklet og i utviklingsfasen hvor det synes lett å ha kontakt med utvikler for ev. å rette opp feil og å komme med ønsker.
- Blockuino.no er nettbasert og trenger å så fall ikke lastes ned på egen maskin hvilket synes å være en fordel dersom verktøyet skal brukes i skolen. Foreløpig kreves det imidlertid at man har installert Arduino editoren (IDE) for kompilering og nedlasting, men det arbeides med å tilby en Blockuino-løsning som kommuniserer dirkete til Arduino-kretskortet.
- Blockuino.no gir en umiddelbar innsyn i C++ koden som genereres hver gang en blokk velges. Dette forbereder brukeren til, på et senere tidspunkt, å kunne gå over til tradisjonell tekstbasert koding.
- Det arbeides med å lage en spesiell versjon av Blockuino.no som kommuniserer direkte med Chromebook datamaskiner, et konsept stadig flere kommuner tilbyr sine elever.
- Blockuino har dessuten et meget enkelt og lett tilgjengelig grensesnitt som burde gjøre det lett å komme i gang. Spesielt er det tiltalende at man kan velge å se den tekstbaserte koden komme opp i et eget vindu etter hvert som man programmerer.

Men foreløpig er det også noen ulemper:

- Blockuino tilbyr pr. dato relativt få blokker. Likeså er utvalget av variabeltyper begrenset, bl.a. mangler float slik at man ikke uten videre kan bruke flyttallsaritmetikk.

Dette heftet er ingen lærebok i blokkprogrammering, men er ment som en støtte for kursleder ved gjennomføring av praktiske brukerkurs for lærere.

En takk til ***Joachim Haagen Skeie*** som har utviklet programverktøyet Blockuino.no og som stadig arbeider med å utvide tilbudet til skoler og andre interesserte, senest har han inkludert mulighet for egen definerte funksjoner (Betaversjon 4. mai 2017). Besøk gjerne hans hjemmeside: kodegenet.no som ved siden av å tilby Blockuino også tilbyr utstyr for bruk i klasserommet i tillegg til lærer og elevkurs.

Nils Kr. Rossing

Skolelaboratoriet ved NTNU
Mai 2017

Innhold

1 Innledning 11

1.1 Blokkprogrammeringsspråk 11

1.2 Oppdagete feil pr. 1. mai 2017 i Blockuino 12

2 Grunnleggende byggeklosser 15

2.1 Komponentoversikter 15

2.1.1 "Sparkfun Inventors Kit" 15

2.2 Komponenter – beskrivelse 17

2.2.1 Koblingsbrettet 17

2.2.2 Motstander 18

2.2.3 Sensorer 19

2.2.4 Halvledere 22

2.2.5 Aktuatorer 23

2.2.6 Arduino – mikrokontrollerkortet 26

3 Oversikt over blokker 29

3.1 Struktur 29

3.2 Oppsett (Setup) 29

3.2.1 Inn- eller utgang 29

3.3 Styring 30

3.3.1 Vent-funksjonen 30

3.3.2 Hvis-setningen (eng. if-setning) 30

3.3.3 Gjenta-sløyfe (eng. for loop) 31

3.4 Operatorer 32

3.4.1 Aritmetiske, komparative og logiske operatorer 32

3.4.2 Tilfeldige tall 33

3.5 Input/Output 33

3.5.1 Digital 33

3.5.2 Analog 34

3.5.3 Seriell kommunikasjon (Serial print) 35

3.5.4 Lyd 35

3.6 Hjelpefunksjoner 36

3.6.1 Kommentar 36

3.6.2 Oversett fra et tallområde til et annet 36

3.6.3 Puls inn 36

3.7 Egne funksjoner 37
3.8 Sensorer 37
3.8.1 Ultrasonisk avstandssensor – SR04 38
3.9 Display 39
3.9.1 OLED display 39
4 Øvingsoppgaver 43
4.1 Blinklys – innledende øvingsoppgave 43
4.1.1 Oppkobling av Blinkende lysdiode 43
4.1.2 Programmering - Tutorial 43
4.2 Morsesenderen 46
4.2.1 Spesifikasjon A: SOS 46
4.2.2 Spesifikasjon C: SOS med variabel hastighet 47
4.2.3 Spesifikasjon D: Reguler hastigheten med en variabel motstand 48
4.3 Trafikklys 49
4.3.1 Spesifikasjon A – Programmer et trafikklys: 49
4.3.2 Spesifikasjon B – Grønt lys på forespørsel: 50
4.3.3 Spesifikasjon C – Grønt lys på forespørsel med blinkende avslutning: 51
4.3.4 Spesifikasjon D – Blinkende grønt lys med lyd: 51
4.3.5 Spesifikasjon E – Blinkende gult lys etter mørkets frambrudd: ... 51
4.3.6 Spesifikasjon F – Det gis grønt lys dersom noen nærmer seg lyset 52
4.3.7 Spesifikasjon G – To eller flere lys som fungerer sammen: 53
5 Andre blokkprogrammeringsverktøy 55
5.1 Scratch for Arduino (S4A) 55
5.1.1 Installasjon 55
5.1.2 Mitt første program i S4A (tutorial) 57
5.1.3 Bruk av porter ved bruk av S4A 59
5.2 Ardublocks 59
5.2.1 Installasjon 60
5.2.2 Bruk av Ardublocks 61
Vedlegg A Løsningsforslag 66
A.1 SOS morsesender 66
A.2 Trafikklys 70

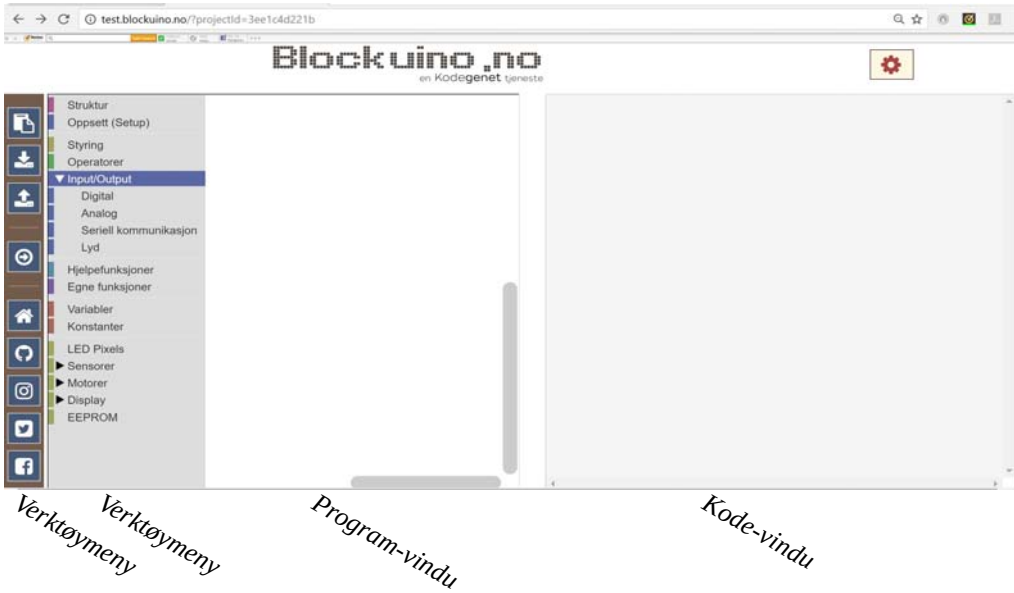
1 Innledning

Blockuino er et høynivå blokkprogrammeringsspråk hvor de ulike kommandoene er formet som byggeklosser som kan stables på hverandre. Dette gjør at strukturen i programmet kommer tydeligere fram og man slipper å tenke på syntaksen, dvs. hvordan hver kommando skal skrives.

Brukergrensesnittet består i hovedsak av fire deler:

- En meny med verktøy - *Verktøymenyen*
- En meny med kommandoer - *Kommandomeny*
- Et vindu for bygging av programstrukturen med blokker - *Programvindu*
- Et vindu med programstrukturen i C-kode - *Kodevindu*

Idet man plukker en kommando (blokk) fra kommandomenyen og legger inn i programvinduet, kommer C-koden automatisk opp i kodevinduet.



Dernest kopierer man innholdet i kodevinduet over i IDE'en som så kompilerer (oversetter koden til maskinspråk) og overfører den kompilerte koden til mikrokontrollerkortet (Arduino).

I tillegg tilbys noen ekstra tjenester via verktøy-ikonet øverst til høyre.

1.1 Blokkprogrammeringsspråk

LabVIEW

Det finnes en rekke ulike varianter av blokkprogrammering, flere av disse har sitt utspring MIT Media lab. Imidlertid var det National Instrument som var de første til å laget et visuelt programmeringsspråk for å sette sammen virtuelle laboratorieoppkoblinger: LabVIEW (Laboratory

Virtual Instrument Engineering Workbench). Dette ble utviklet for Apple Macintosh så tidlig som i 1986.

LEGO ROBOLAB og LEGO MINDSTORMS...

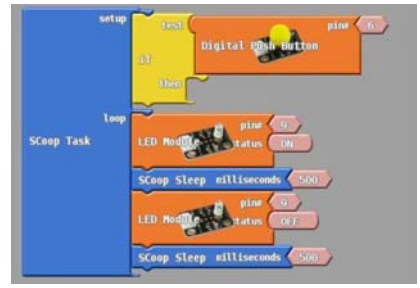
...bygger på LabVIEW og ble utviklet ved TUFTS University Medford, Massachusetts, United States. De første versjonene kom i 1994. Senere ble en ny versjon av denne programvaren utviklet ved MIT og i 1998 ble **LEGO MINDSTORM** introdusert. Både **ROBOLAB** og MINDSTORMS anvender det vi kan kalle visuell blokkprogrammering.

Scratch

I 2003 introduserte MIT Media Labs *Lifelong kindergarten group*, **Scratch**, som var ment som en enkel introduksjon til programmering fra 8 år og oppover. Navnet stammer fra miksing av musikk ved å skrape picupen langs rillene i en vinylplata og på den måten skape ny lyd fra gammel lyd. Dette var også tanken bak Scratch hvor brukerne ble oppfordret til å dele sine programmer med andre, og gjerne bygge nye programvarianter på andres programmer, såkalt *remix*. Med sine bevegelig figurer, *sprites*, egner språket seg godt for å lage spill og visualisering av historier.

Det er også utviklet versjoner av Scratch som kan kommunisere med Arduino, den såkalte Scratch for Arduino eller S4A. Denne forutsetter at det er en kontinuerlig forbindelse mellom verts PC'en og Arduino-kortet.

Ardublocks



Ardublocks er et grafisk programspråk beregnet for Arduino. De seneste utgavene av Arduino programvaren (IDE) inneholder Ardublocks. Programmet har et meget rikholdig utvalg av blokker kan tilpasses en rekke forskjellige sensorer og aktuatorer fra ulike firma. Flere av blokkene som er tilpasset spesiell hårdvare, har en avbildning av hårdvaren på blokken som vist på figuren til høyre.

Om programmet ikke er inkludert i Arduino programvaren kan den lastes ned og installeres fra:
http://blog.ardublock.com/engetting-started-ardublockzhardublock/

1.2 Oppdagete feil pr. 1. mai 2017 i Blockuino

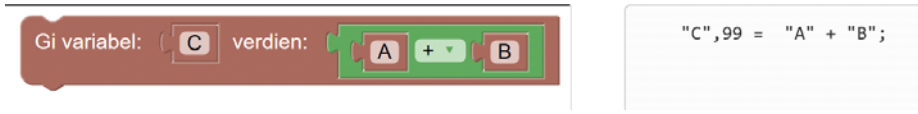
Disse problemene synes kun å eksisterer under MicroSoft Edge:

- Har problemer med å lagre en blokkode, det går første gangen,
  men senere er det litt tilfeldig om det lar seg gjøre eller ikke. Kjører windows 10.
- Senere har jeg erfart at jeg erfart at verktøymenyen slutter å virke slik at jeg mister muligheten til å ta vare på koden samtidig som jeg også mister muligheten til å se C-koden-.

Disse problemene eksisterer både under MicroSoft Edge og Chroma

- Ved bruk av funksjonen ”tone” så kan det se ut som det oppstår en feil når en setter inn ”verdi” for frekvens. I dette tilfellet kommer det opp to verdier (440,99). Dersom man bruker konstanten ”Tone” går det bra. Se figur under.

- En lignende feil framkommer når man tilordner variabler som vist under



Det kan se ut som at dette skjer stadig når man tilordner variabler.

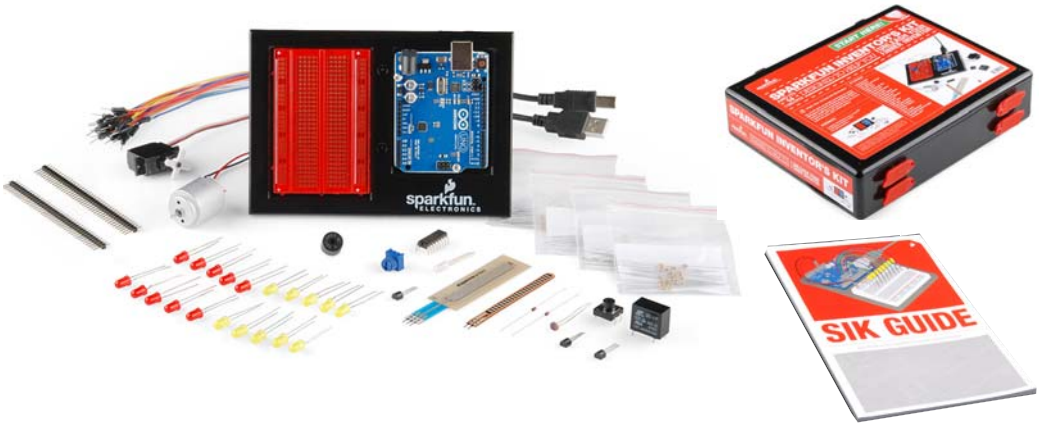
2 Grunnleggende byggeklosser

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

2.1 Komponentoversikter

2.1.1 "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
- 1 stk. Display 16x2 karakterer - Standard fra V3.2

Mer informasjon komponentene og datablader finnes i avsnitt 2.2 på side 17 eller på følgende nettside: https://www.sparkfun.com/products/11227

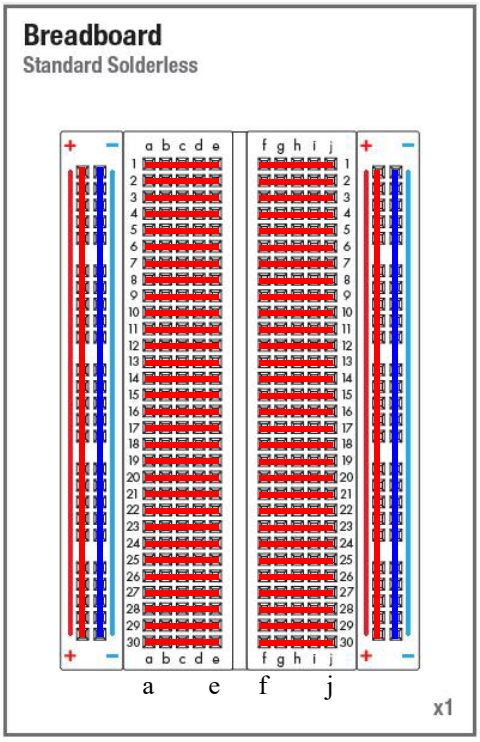
Supplement til Inventors kit

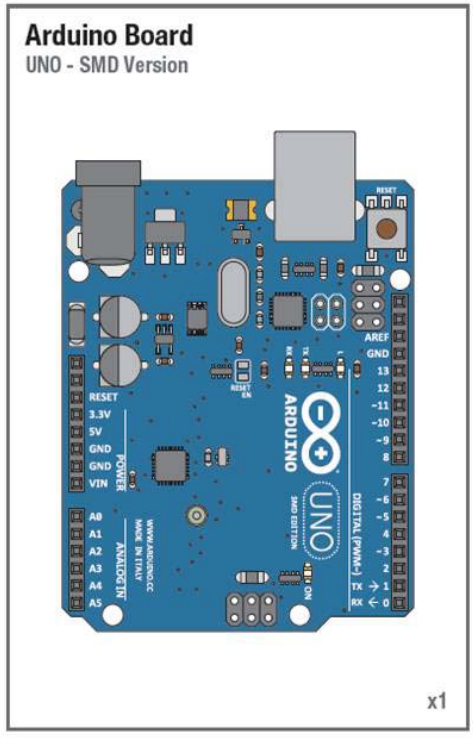
- 10 stk. Grønn lysdiode

2.2 Komponenter – beskrivelse

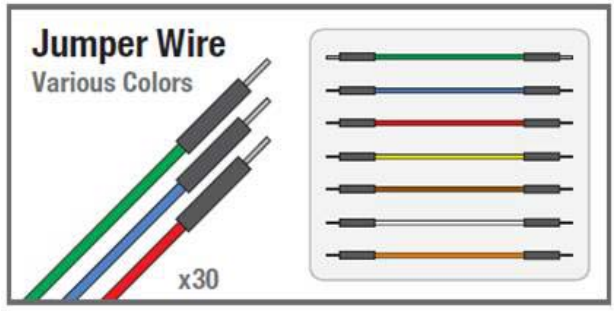
2.2.1 Koblingsbrettet

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



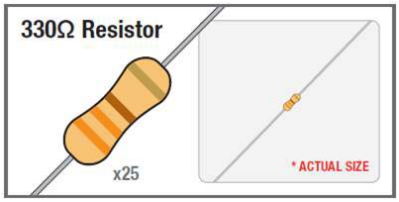


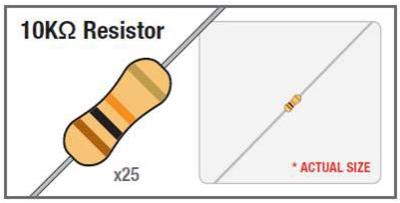
Strekene på koblingsbrettet viser hvilke koblingspunkter som er koblet sammen på undersiden. To komponentbein som er koblet til samme rad i to av hullene a – e eller f – j er koblet elektrisk sammen. Komponentene plasseres på koblingsbrettet og forbindes med Arduino-en ved hjelp av jumperer.

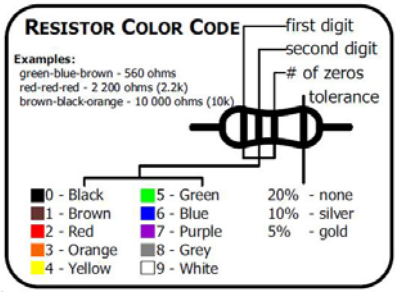


2.2.2 Motstander

Faste motstander







Beskrivelse:

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

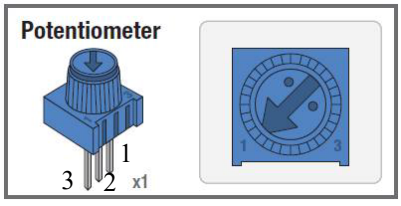
Bruksområder:

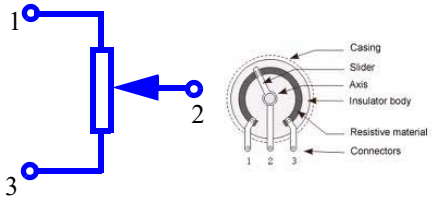
Motstander brukes til å begrense strømmen i en komponent, eller for å etablere et spenningsnivå slik som i spenningsdeleren. I følge Ohms lov vil spenningen over en motstand øke proporsjonalt med strømmen. Dette utnyttes når en ønsker å omdanne en varierende motstandsverdi (resistans) til en varierende spenning som ofte er tilfelle ved bruk av resistive sensorer. 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 (tier-potensen).

Potensiometer





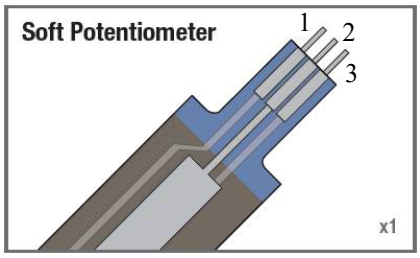
Beskrivelse:

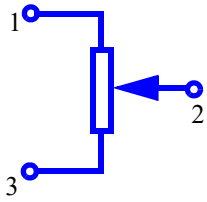
Et potensiometer er en motstand med et variabelt uttak. Ved hjelp av et lite ratt kan uttaket flyttes langs motstanden. På denne måten kan en på pinne 2 ta ut en brøkdel av spenningen mellom pinne 1 og 3.

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:

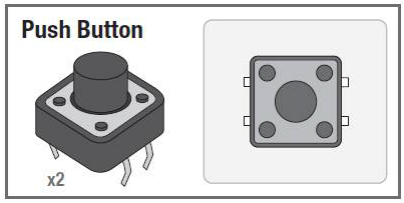
En trykkfølsom motstand fungerer på samme måte som et potensiometer. Istedet for å dreie på en knott, presser man på metallfilmen et sted langs sensoren og som gir den spenningen man ønsker at sensoren skal gi ut.

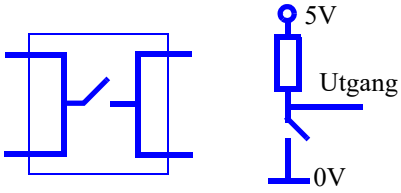
Bruksområder:

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

2.2.3 Sensorer

Bryter





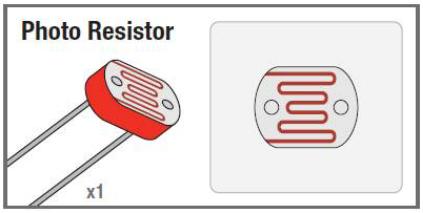
Beskrivelse:

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

Bruksområder:

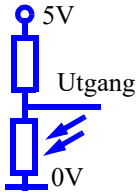
Brytere brukes til å gi en enkel på/av informasjon til kretsen, på samme måte som en lysbryter. For at kretsen skal forstå informasjonen må trykket omdannes til en spenning. Ved å koble bryteren mellom en motstand (10 kOhm) til 5 V og jord, vil en på utgangen få en spenning som går fra 5 V til 0 V når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

Fotomotstand

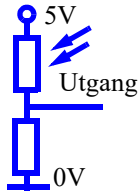


Beskrivelse:

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



Høy spenning ut ved mørke



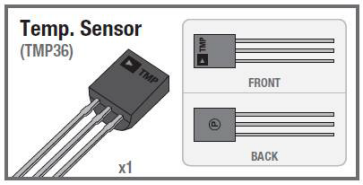
Høy spenning ut ved lys

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

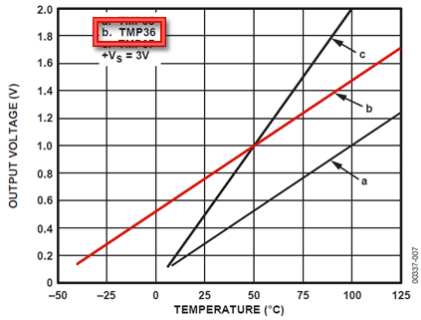


Beskrivelse:

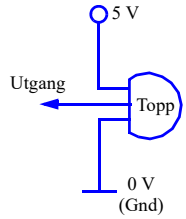
En temperatursensor (TMP36) av denne typen registrerer temperaturen og gir ut en spenning som er proporsjonal med temperaturen. OBS! Unngå å forbytte med transistorene!

Bruksområder:

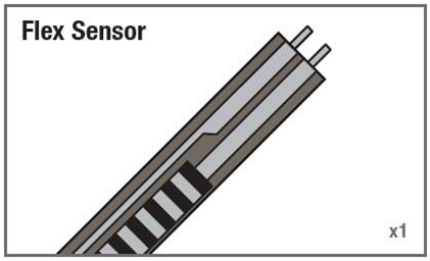
Temperatursensorer brukes i elektroniske termometre eller i termostater for å regulere en varmeovn. Den kan også brukes for å beskytte elektronikk, ved at strømmen brytes når temperaturen overskrider et maksimalt nivå. Hos TMP36 øker spenning med 10 mV pr. grad C. Ved 0° C er spenningen 0,5 V.



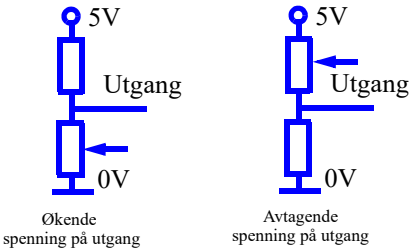
Spenning som funksjon av temperatur (TMP36 rød kurve)



Bøyesensor



Zebrastripenen på motsatt side av trykket



Økende spenning på utgang

Avtagende spenning på utgang

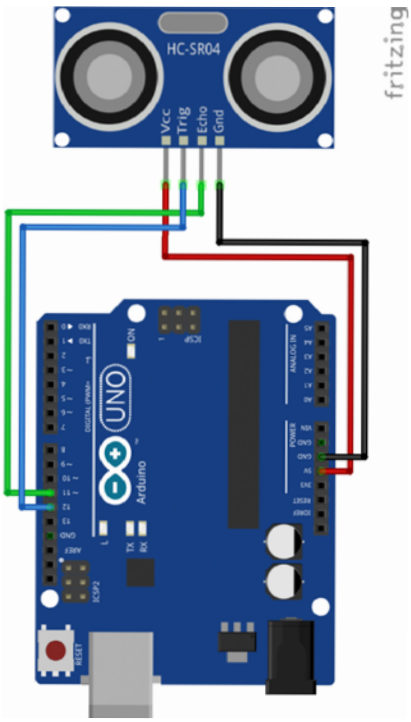
Beskrivelse:

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 med påfølgende nedbøyning. I en slik sammenheng går en slik sensor under betegnelsen *strekklapp*.

Ultrasonisk avstandssensor – SR04 (tillegg til SIK V3.2)



Beskrivelse:

Sensoren finnes i forskjellige utgaver, men alle fungerer stort sett likt. Arduinoen sender en puls til trigger-inngangen til sensoren (Trig). Sensoren sender ut en ultrasonisk puls omkring 40 kHz og venter på at det skal komme tilbake et ekko. Når ekkoet kommer tilbake måles, tidsforsinkelsen mellom utsendt og mottatt signal. Ut fra kjennskap til lydhastigheten i luft, kan avstanden til ekkokilden beregnes.
Sensoren har fire eller fem terminaler, vår har fire:

VCC	5,0 V
Trig	Triggersignal fra Arduino kobles til en digital utgang
Echo	Ekkosignal til Arduino kobles til en digital inngang
GND	GND (minus/jord)

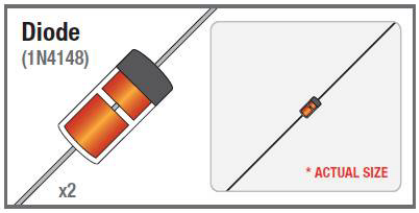
Bruksområder:

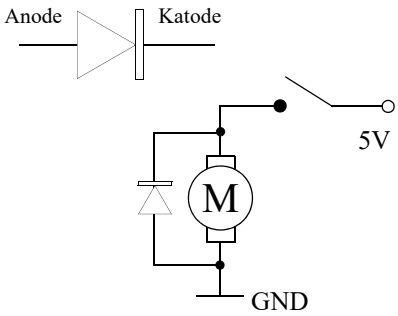
Slike sensorer har mange ulike bruksområder f.eks. for å måle avstanden til et fotoobjekt for å justere avstandsinnstillingen hos et kamera, eller måle nivået i en tank eller i robotteknologi for å oppdage hindringer.

2.2.4 Halvledere

I denne sammenhengen er dette dioder og transistorer.

Diode





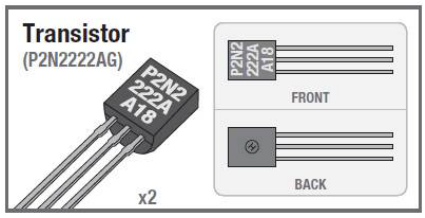
Beskrivelse:

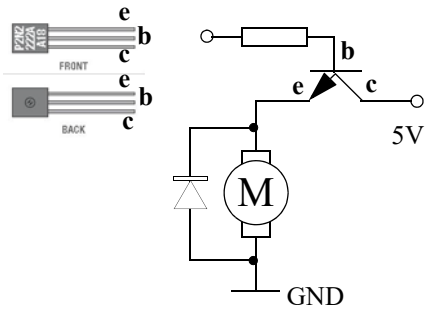
Dioden er en halvleder som kun leder strøm en vei, fra anode til katode.

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 som styrer motoren eller releet. Den må derfor monteres slik at den normalt stenger for strømmen slik at den kan gå gjennom motoren. Når strømmen i motoren brytes, oppstår en motspenning 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 nederst til venstre, vil denne motspenningen bli 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) (egentlig styrt av strømmen inn i basen). Når den blir stor nok begynner det å gå en strøm i transistoren.

Bruksområder:

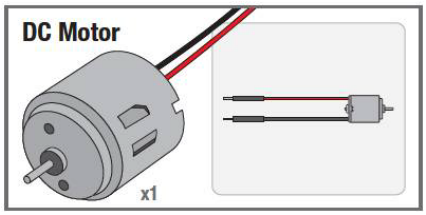
En transistor har gjerne en av to funksjoner. Som *forsterker* når basespenningen/strømmen varierer innen et lite område. Som *bryter* når basespenningen/strømmen er så stor at den slår transistoren på eller av.

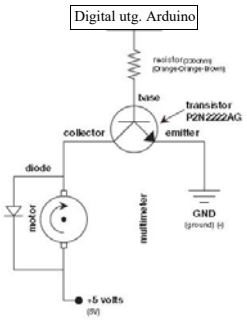
Transistorer som signalforsterker brukes i elektroniske forsterkere, radiosendere og mottakere, TV-er osv. Som brytere i datamaskiner og i styringssystemer.

2.2.5 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), lys (LED, lyspærer) eller varme opp en gjenstand eller et rom (glødetråd).

Motor





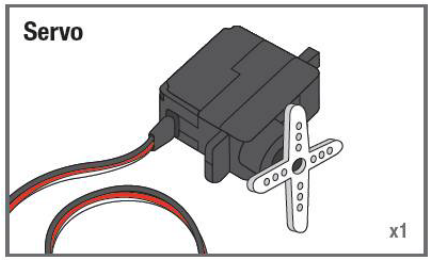
Beskrivelse:

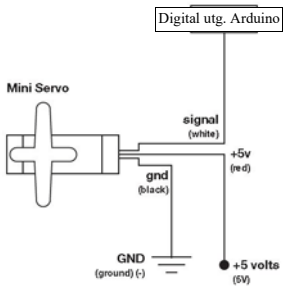
Akslingen begynner å rotere når det tilføres spenning over fra 3 V. Motoren som følger med kitet kan brukes på spenninger opp til 40 V og strømmer på opp mot 200 mA. Siden Arduinoen ikke klarer å levere så store, benytter vi en transistor som bryter som tåler strømmen. Legg merke til "fly back" dioden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

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, 3D-printere og etter hvert elektriske biler.

Servo





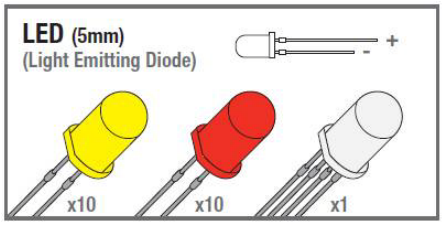
Beskrivelse:

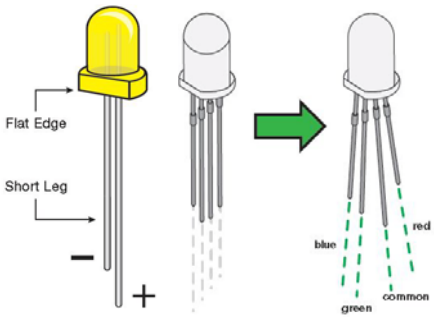
En servo er en slags motor som kan dreie akslingen en bestemt vinkel. Denne servoen kan på kommando fra mikrokontrolleren dreie akslingen en vinkel på fra 0 - 180°. Dreiningen skjer ved at servoen mottar pulser, hvor lengden av pulsen bestemmer dreievinkelen. En pulslengde på 1,5 ms gir en vinkel på 90°. Pulsene må gjentas med jevne mellomrom omtrent som når 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

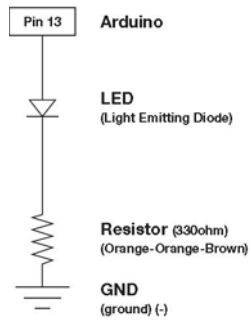




Beskrivelse:

Siden disse også er en dioder, 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.

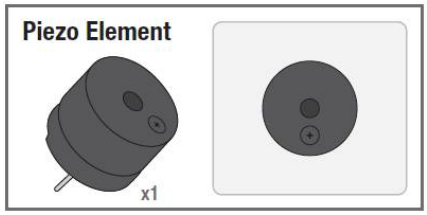
I settet er det vedlagt røde, grønne og gule lysdioder. I tillegg finnes en RGB-diode hvor en rød-, en grønn og en blå lysdiode er montert i samme kapsel.

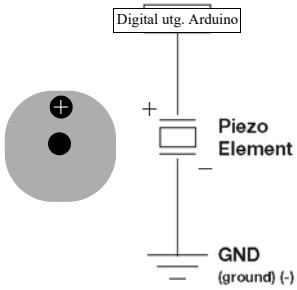


Bruksområder:

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

Buzzer





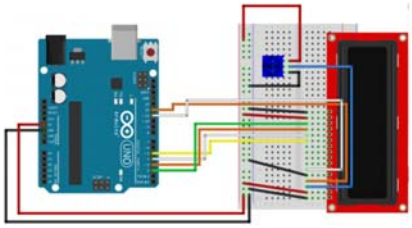
Beskrivelse:

Piezo-elementet 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:

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

Display



Se Sparkfun's SIK for oppkobling:
https://learn.sparkfun.com/tutorials/sik-experiment-guide-for-arduino---v32/all

Beskrivelse:

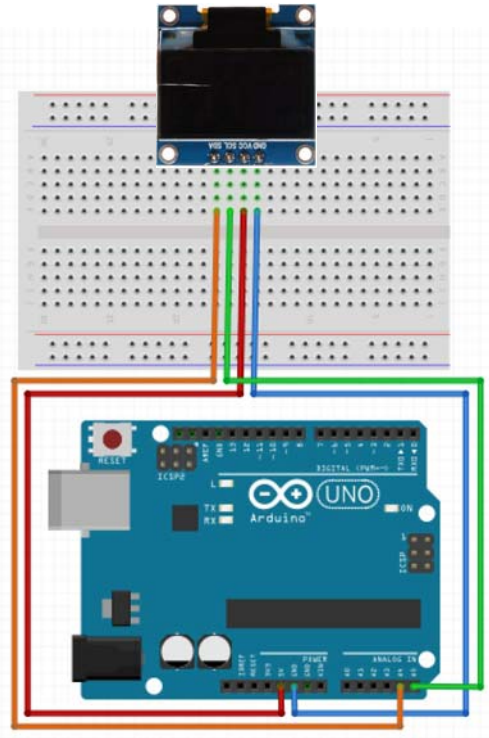
I de seneste utgavene av Sparkfun's inventors kit følger det med et alfa-numerisk display ADM1602K som kan vise to rader med opp til 16 karakterer i hver rad. Som vist på figuren over krever displayet flere tilkoblingspunkter til Arduino-en. I tillegg er det også vanlig å koble til et potensiometer for styre styrken til tallene på displayet. Bruk av displayet krever et eget bibliotek av kommandoer som følger med:

```
#include <LiquidCrystal.h>
```

Bruksområder:

Display er praktisk når man ønsker å vise et måleresultatet, f.eks. temperatur eller lysstyrke. og ellers kommunisere med brukeren av mikrokontrolleren.

OLED-display (tillegg til SIK V3.2)



Beskrivelse:

Funksjonen er tilpasset OLED display 32 x 128 pixler (0,96"). Displayet har fire terminaler:

VCC	3,0 - 5,0 V
GND	GND (minus/jord)
SCL	Serie klokkesignal (I^2C)
SDA	Serie datasignal (I^2C).

Displayet kommuniserer over en serielinje. Dette gjør det særdeles enkelt å koble opp. Blockuino er tilpasset driverkretsen SSD1316. Det er derfor viktig at displayet har nettopp denne. Displayet krever spesielle biblioteker.

Bruksområder:

Displayer brukes over alt for å kommunisere med brukeren, enten gjennom tekst eller grafikk (ikoner).

Oppkobling:

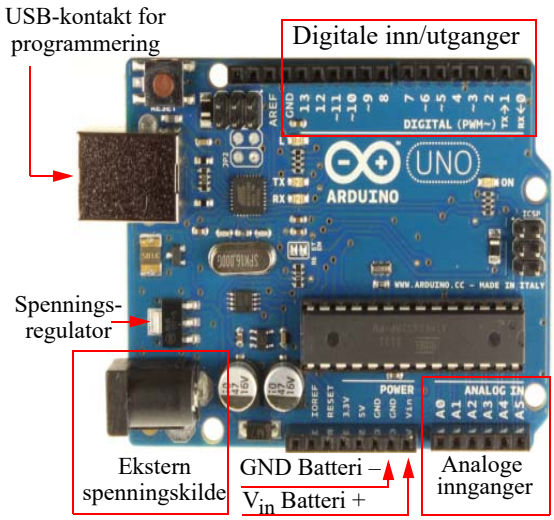
Figuren til venstre viser oppkoblingen. Legg spesielt merke til at seriekommunikasjonen gjøres via A5 (SCL) og A4 (SDA).

2.2.6 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 programmer 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 av mikrokontrollerkortene i Arduino-familien, og dermed leveres til en overkommelig pris (KultogBilligs pris kr. 129,- inkl. MVA + frakt eller ELFA pris kr. 237,50,- inkl. MVA)



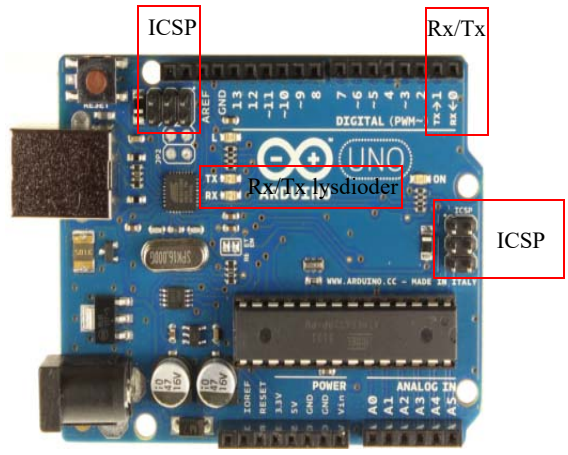
Kortet er bygget opp omkring Atmel mikrokontrolleren ATmega328P med en klokkefrekvens på 16 MHz og et flash lager på 32 kbyte, SRAM 2 kbyte og EEPROM 1 kbyte.

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. Inngangene A4 (SDA - Serial DAta) og A5 (SCL - Serial CLock) kan også brukes ved seriekommunikasjon (I^2C).
- **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 for en tid inntil strømtrekket reduseres under denne grensen.
- **Strømtilførsel**
  Tilkoblingsplugg for batterieliminator har anbefalt spenning fra 7 – 12 V (grenseverdier 6 – 20 V). Batteri kan enten tilkobles eliminatorpluggen (2.1 mm + i senter) eller via V_{in} (+) og GND (–). 5V utgangen lever spenning til f.eks. kretser koblet opp på koblingsbrett e.l.. Enkelte komponenter trenger lavere spenning som ev. kan leveres fra 3.3 V utgangen.
- **Reset**
  Kortet inneholder en RESET-knapp som resetter programmet. Ved å trykke på denne starter programmet på nytt.

Kantkontaktene er montert slik at tilleggskort kan monteres rett ned på Arduino-kortet så kalt "shield"-kort eller skoldkort.

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 program-editoren (IDE) for monitorering av data på PC-skjermen.

- **I^{2}C-databus**
  I ^{2}C 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), og flere komponenter kan kobles opp samtidig på de to linjene dette er mulig da 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.

1. For mer informasjon se: http://www.i2c-bus.org/

3 Oversikt over blokker

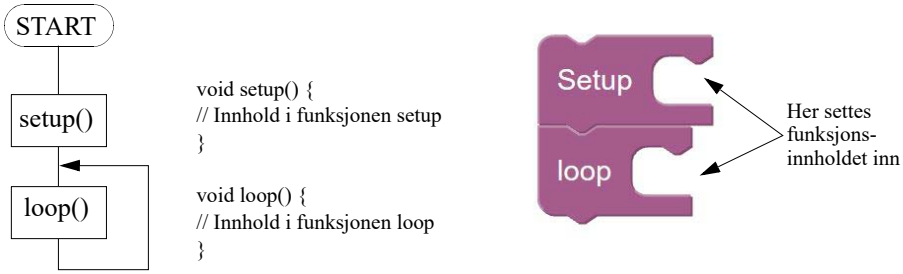
I dette avsnittet skal vi se nærmere på de vanligste blokkene innen hver kategori og se hvordan de brukes.

3.1 Struktur

Programmer laget for Arduino består normalt av to grunnleggende funksjoner, disse finner vi menyen "Struktur":

setup () som kun kjøres hver gang programmet starter opp og

loop () som kjøres om og om igjen så lenge strømmen er tilkoblet eller et nytt program lastes inn.

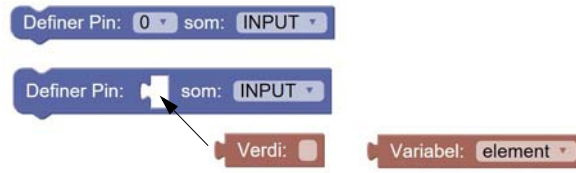


På figuren er tre ulike representasjoner for det samme. Det kan ofte være lurt å begynne med å tegne et blokkdiagram for å strukturere programmet. Flytdiagrammet forteller i hvilken rekkefølge de forskjellige operasjonene skal gjøres. Dernest kan man velge inn de to funksjonene og henge dem etter hverandre som vist i C-kode og blokkode.

3.2 Oppsett (Setup)

Innholdet i denne menyen er ment for kommandoer for å sette opp portene og innholder kommandoer som naturlig hører hjemme i setup() funksjonen.

3.2.1 Inn- eller utgang



De digitale portene på mikrokontrollere kan enten være innganger eller utganger og ofte er det slik at vi som brukere kan bestemme om en port skal være en inn- eller en utgang. Normalt gjøres dette en gang i programmet for deretter å være uforandret så lenge programmet kjører. Det er derfor vanlig å gjøre dette i setup() funksjonen.

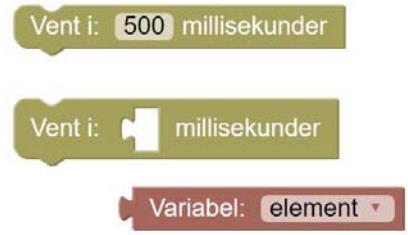
Vi har tre valg. Enten velger vi en blokk hvor de digitale portene er forhåndsdefinert fra 0 til 13. Eller vi kan velge blokken som gjør det mulig å velge en verdi eller en variabel som vist nederst på figuren over. Dette kan være praktisk om det er en verdi som går igjen på ulike plasser i programmet. Ved å definere en variabel (eller konstant) og tilordne den en verdi i starten av programmet, kan en senere på en enkel måte, endre denne verdien i hele programmet ved kun å forandre innholdet av variabelen i starten av programmet.

3.3 Styring

Menyen "Styring" er programelementer som gjør det mulig å gå forskjellige veier i programmet på bakgrunn av betingelser. Det kan f.eks. være en sensor som måler temperatur. Er temperaturen over 20°C skal varmeovnen slås av, er temperaturen under 20°C skal den slås på. Her finner vi også "Vent" funksjonen

3.3.1 Vent-funksjonen

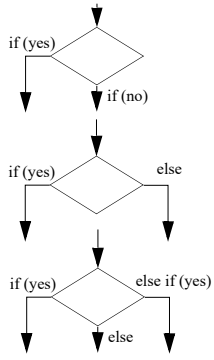
Vent-funksjonen er en enkel kommando som stopper programmet i en oppgitt tid. Siden kommandoen stopper programmet er det særdeles ineffektiv, dvs. programmet gjør ingenting annet enn å vente. Den er imidlertid praktisk å bruke når man har god tid.



Vent-funksjonen kommer i to utgaver:

- Vent en gitt tid - skriv tiden direkte inn i funksjonen
- Vent en variabel tid - ventetiden bestemmes av en variabel.

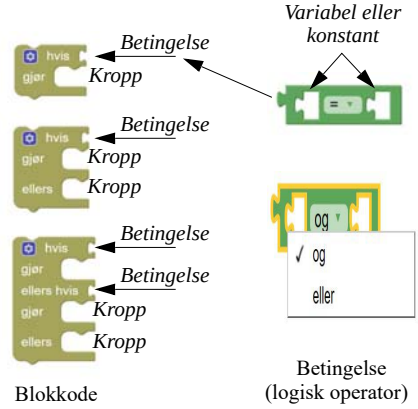
3.3.2 Hvis-setningen (eng. if-setning)



Flytdiagram

```
if (<betingelse>) {
}

if (<betingelse>) {
} else {
}

if (<betingelse>) {
} else if (<betingelse>) {
} else {
}
```

C-kode



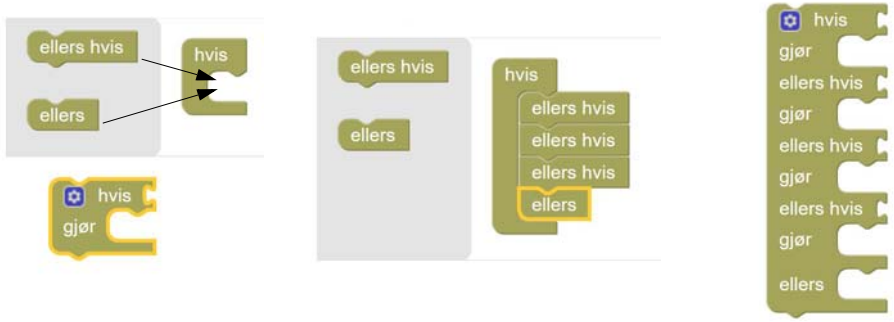
Blokkode

Betingelse (logisk operator)

En hver hvis-setningen inneholder en betingelse. Betingelsen er en logisk operator som sammenligner to tall. Det kan f.eks. være en variabel og en konstant. Operatorene kan være mindre (<)

eller større (>) enn, lik (=) eller ulik (=) o.l. Avhengig av hvilke betingelser som er oppfylt (eller ikke oppfylt) utføres koden innen hver seksjon.

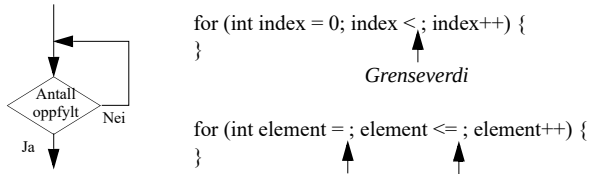
Vi legger merke til et lite tannhjul øverst i venstre hjørne av symbolet. Dette kan vi bruke dersom vi ønsker å utvide antall betingelser i hvis-setningen som vist på figuren under.

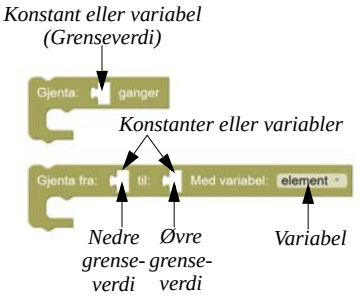


Når vi igjen trykker på tannhjulet vil vi sitte igjen med hvis-setningen til høyre i figuren, her med fire betingete if-setninger avsluttet med en else-setning.

3.3.3 Gjenta-sløyfe (eng. for loop)

Gjenta-sløyfen brukes når den samme operasjonen skal gjentas flere ganger. I Blockuino finner vi denne i to utgaver. En hvor vi kun angir antall ganger innholdet (kroppen) i *Gjenta-sløyfa* skal gjentas, og en der vi kan telle opp en selvvalgt variable fra en nedre grense til en øvre grense.





Flytdiagram C-kode Blokkode

Vi legger merke til at den øverste *Gjenta-sløyfa* skrevet i C-kode inneholder variabelen "index" denne er definert av blokka og kan ikke endres. Vi ser at denne alltid vil starte på 0 og telle opp til *grenseverdien* som her står åpen.

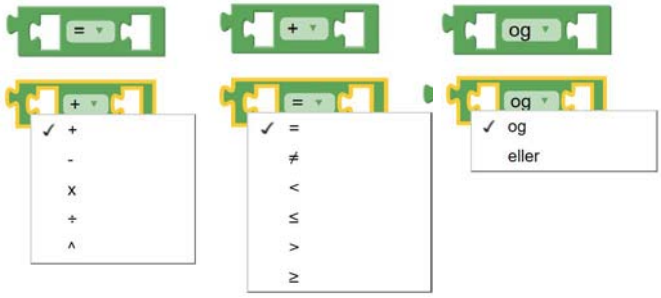
I den andre varianten av Gjenta-sløyfa kan vi velge å bruke en variabel samtidig som vi kan sette inn en nedre og øvre grenseverdi for denne variabelen. Variabelen vil normalt økes med én for hver runde, men kan også endres inne i Gjenta-sløyfa om dette er ønskelig. Her må man imidlertid være klar over at den uansett, automatisk vil øke verdien til variabelen med én for hver runde.

Normalt vil det finnes flere slik styrekommandoer, men foreløpig er ikke disse tatt med i Blockuino.

3.4 Operatorer

3.4.1 Aritmetiske, komparative og logiske operatorer

Operatorer er ikke frittstående kommandoer, men inngår gjerne i kombinasjon med andre blokkoder, for eksempel i forbindelse med betingelsene i *hvis-setninger* som vist foran. Under vist et eksempel satt inn i en *hvis-blokk*.



Aritmetisk operator Komparativ operator Logisk operator

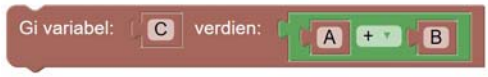
I hovedsak tilbyr Blockuino tre operatorer:

Aritmetisk operatorer – som utfører en aritmetisk operasjon mellom to tall eller variabler, det være seg addisjon (+), subtraksjon (–), multiplikasjon (x) eller divisjon (/). I tillegg har den "opphøyet i (^).

Komparativ operator – som sammenligner to tall eller variabler, det være seg større enn (>) eller mindre enn (<) eller lik (=) eller varianter av disse.

Logiske operatorer – som kombinerer to variabler (A, B), for eksempel:
A *og* B > C A *eller* B < C

Disse kan brukes sammen med andre blokkoder. Vi har alt sett et eksempel bruk av en komparativ operator i hvis-setningen. Aritmetiske operatorer brukes vanligvis i tilordning av variabler. F.eks. kan vi sette at C = A + B (variabelen C = variabelen A + variabelen B)



3.4.2 Tilfeldige tall

Skal man f.eks. lag en elektronisk terning har man behov for å generer tilfeldige tall i tallområdet 1 – 6. I så fall velger man blokken *Tilfeldige tall* og setter enten inn to tall *Fra* og *Til* eller man kan sette inn variable som vist på figuren under.

Dersom man ønsker å legge inn en fast verdi, velger man blokken *Verdi* fra menyen *Variabler* og skriver inn ønsket verdi boksen. Ønsker man å bruke en variabel som *Fra* og *Til* verdier så velger man blokken *Variabel: element* og velger ønsket element fra elementmenyen.

3.5 Input/Output

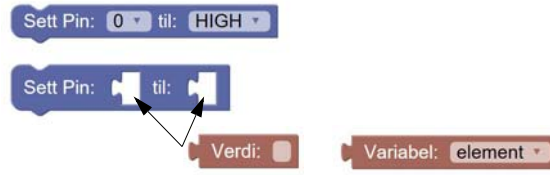
I denne menyen finner vi blokkommandoer styrer hårdvaren som f.eks. å sette ut en 5 V spenning (høy) på en av de digitale portene, lese fra eller skrive til porter o.l. Dessuten finner vi kommandoer for seriekommunikasjon med monitoren og styring av lyd. Menyen inneholder følgende undermenyer:

- Digital (skrive til eller les fra digitale porter
- Analog (les fra analoge porter, ev. sett ut et pulsbreddemodulert signal)

3.5.1 Digital

Skriv til eller les fra digital pinne

Med blokkene *Sett Pin* kan vi sette en av pinnene høy (5 V) eller lav (0 V). Vi kan enten gjøre dette ved å velge Pin nummer og verdi (høy (1) eller lav (0)) eller vi kan sette inn *verdier* og *variabler* om det er ønskelig som vist nederst på figuren under.



Med blokkommandoene *Les fra Digital Pin*: kan man lese av verdien på en digital inngang (0 eller 1). Disse verdiene leses så inn i en variabel ved hjelp av blokken *Gi variabelen: verdien:*

3.5.2 Analog

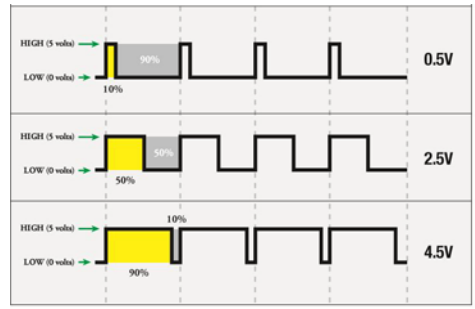
Les fra analog port (pinne)

Med blokkommandoene *Les fra Analog Pin*: kan man lese av verdien på en analog inngang 0 – 5 V (0 – 1023). Disse verdiene leses så inn i en variabel ved hjelp av blokken *Gi variabelen: verdien:*

Den analoge spenningen (0 – 5 V) som tilføres den analoge inngangen, konverteres til et tall ved hjelp av en *Analog til digital konverter* (ADC). Dette tallet kan ha verdier fra 0 – 1023 avhengig av spenningen på den analoge inngangen. 0 tilsvarer spenningen 0 V og 1023 tilsvarer 5 V, mens alle mellomliggende spenninger gis et tall mellom 0 – 1023 avhengig av hvor på skalaen spenningen befinner seg. Analoge inngangen må ikke tilføres spenningen større enn 5 V.

Skriv analog verdi til port (PWM)

Normalt kan digitale porter ikke gi ut annet en 0 og 1, dvs. 0 og 5 V og ikke noe i mellom disse to spenningsverdiene. Det finnes imidlertid en annen mulighet som kalles *puls bredde modulasjon* (Puls Width Modulation - PWM).



La oss tenke oss at vi ønsker å kunne dempe lysstyrken i en lysdiode. Dette får man til ved å sende korte eller lengre pulser på lysdioden, hver puls på 5V. Siden pulsene kommer svært tett på hverandre vil man ikke være istand til å se at lysdioden flimrer, men se ut som den gir fra seg et jevnt lys. Dersom på-tiden er relativt kort i forhold til av-tiden, vil en oppfattet at lysdioden lyser med et svakt lys. Dersom på-tiden er lang i forhold til av-tiden, vil det se ut som om lysdioden lyser kraftig. Ved hjelp av blokkoden: *Skriv til Pin (PWM)* kan både spesifisere hvilken digital port signalet skal komme ut på og forholdet mellom på og av tiden. En høy verdi (maks. 255) gir høy lysstyrke, en lav verdi (min. 0) svak lysstyrke. Verdien 128 gir en lysstyrke omtrent midt mellom minimum og maksimum..

Porten

Telleren i forholdstallet

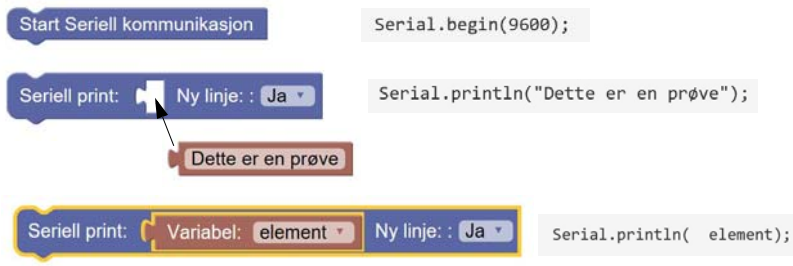

```
analogWrite((9), (128));
```

Forholdet mellom på-tid og av-tid kan deles inn i 256 deler, fra 0/256 hvor lysdioden er helt av hele tiden og 255/255 hvor lysdioden er på hele tiden. *Verdien* angir telleren i denne brøken (nederst).

Det ikke alle digitale porter som kan levere et puls bredde modulert signal, dette gjelder bare portene D3, D5, D6, D9, D10 og D11, som er merket med en ~ (tilde).

3.5.3 Seriell kommunikasjon (Serial print)

Noen ganger har man behov for skrive fra programmet og ut til en monitor. Dette kan være nødvendig for å få svar på beregnede verdier eller under feilfinning for å se om programmet har vært innom et spesielt steg i programmet.



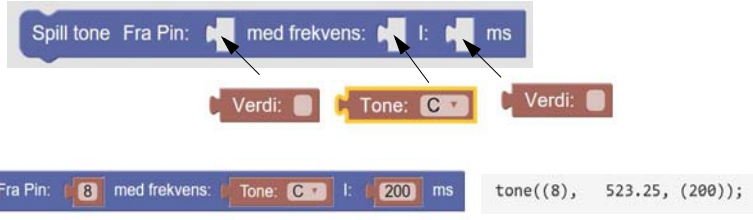
Kommandoblokken *Start Seriell kommunikasjon* plasseres i *setup()* funksjonen og skal bare kjøres en gang. Denne kommandoen brukes for å sette opp farten til denne serielle linken mellom PC og Arduino-kortet. Normalt vil denne sette opp en hastighet på 9600 baud pr. sek. Enten kan vi sette inn en verdi og skrive inn en tekst.

Ønsker man å skrive ut en tekst velger man *Verdi:* fra menyen *Variabler* og skriver inn teksten. Ønsker man derimot å skrive ut innholdet i en variabel, velger man blokken *Variabel:* og hvilket element man ønsker å skrie ut innholdet fra.

Om man ønsker ny linje for hver utskrift velges *Ny linje: Ja.*

3.5.4 Lyd

Ved bruk av en piezoelektrisk høyttaler (lydgiver) kan man generere toner av ulike slag og med ønsket lengde. *Fra Pin:* angir fra hvilken pinne "lyden" skal komme – denne oppgis ved hjelp av blokken *Verdi:*. Frekvensen angis med *Verdi:* eller konstanten *Tone:*. Lengden av tonen oppgis i ms (millisekunder) og angis med blokken *Verdi:*.,



Vær oppmerksom på at programmet ikke stopper opp av denne kommandoen, selv om lengden av lyden er satt til 200 ms.

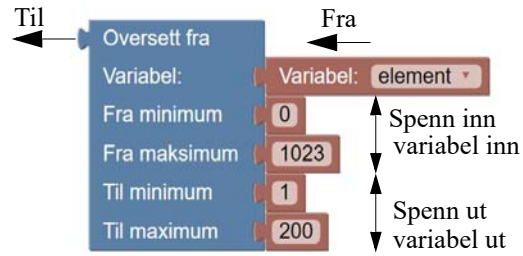
3.6 Hjelpefunksjoner

Alle blokker kan betraktes som funksjoner som utfører en spesiell funksjon. Blokkene i menyen Funksjoner tilbyr derfor noen få spesielle funksjoner som vi skal se på i dette avsnittet.

3.6.1 Kommentar

Noen ganger har man behov for å legge inn et mellomrom eller en kommentar i programkoden som en hjelp til selv å få oversikt. Siden det ikke er mulig å legge inn mellomrom i blokkoden er det laget en egen blokk som heter kommentar. Her kan man legge inn tekst eller mellomrom for å lage en blank linje i C-koden. Denne hjelpefunksjonen vil ikke bli overført til mikrokontrollerkortet.

3.6.2 Oversett fra et tallområde til et annet

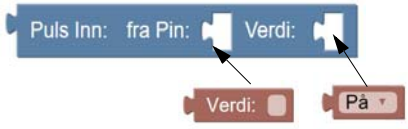


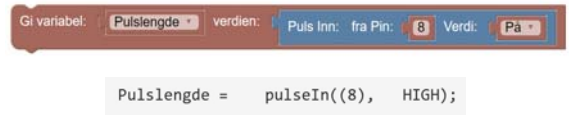
Sidene verdiene 0 – 1023 kan være et litt upassende område for vår anvendelse, så kan vi konvertere 0 – 1023 til f.eks. 1 – 200. Dette kan vi gjøre med blokken "Oversett fra" som finnes i menyen "Hjelpefunksjoner" (se figuren til høyre).

Denne konverterer ett verdiområde (spenn) fra "fra minimum" – "fra maksimum" hos en variabelen , til et verdiområde (spenn) fra "til minimum" – "til maksimum" for verdien som kommer ut av funksjonen.

3.6.3 Puls inn

Denne funksjonen er praktisk dersom man ønsker å måle lengden av en puls på en av de digitale inngangene. Funksjonen måler tiden fra spenningen på inngang *fra Pin:* som skal måles går høy (0 til 5 V) og til den igjen går lav dersom *Verdi:* er satt lik *På*, og måler tiden fra høy til lav dersom *Verdi:* er satt lik *Av*. Funksjonen leverer så verdien til en variabel.



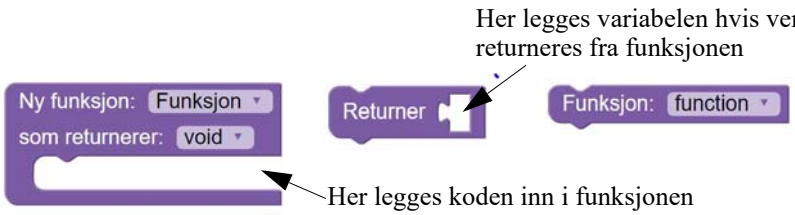


Dette er vist til høyre på figuren over. Den tilsvarende C-koden står under. Her leveres pulslengden til variabelen *Pulslengde*.

3.7 Egne funksjoner[2]

Her kan definere sine egne funksjoner. Undermenyen inneholder tre blokker:

- *Ny funksjon*: Definisjon av ny funksjon. Legg inn et unikt navn.
  Det er lagt opp til at funksjonen skal returnere en verdi
  (ennå ikke implementert). Innholdet i funksjonen legges inn i "gapet" hos
  funksjonen.
- *Returner*: Returpunkt i funksjonen. Denne kommandoen legges inn i funksjonen der
  men ønsker at programmet skal returnere til hovedprogrammet (loop).
- *Funksjon*: Funksjonskallet som sørger for å utføre funksjonen.
  Denne blokken brukes for å kalle funksjonen fra hovedprogrammet (loop).
  Kallet kan også skje fra en annen egen definert funksjon.



3.8 Sensorer

Det fleste resistive sensorer kan lett inkluderes i et prosjekt ved å lese av en analog pinne for så å beregne den fysiske verdien ut fra målt digitla verdi og kjennskapet til sensoren. Andre sensorer har et digitalt grensesnitt som krever mer sofistikert databehandling. Slike sensorer leverers med skreddersydde biblioteker.

Blockuino støtter to slike sensorer[3]:

- Ultrasonisk avstandsmåler
- Temperatur og fukstighetssensor DHT11

Her skal vi beskrive blokkene som støtter den ultrasoniske sensoren SR04.

2. Siden dette er en beta-versjon har egendefinerte funksjoner ennå ikke fått noen returverdi. En må dermed benytte globale variabler for retur av verdier.
3. Pr. 1. mai 2017

3.8.1 Ultrasonisk avstandssensor – SR04

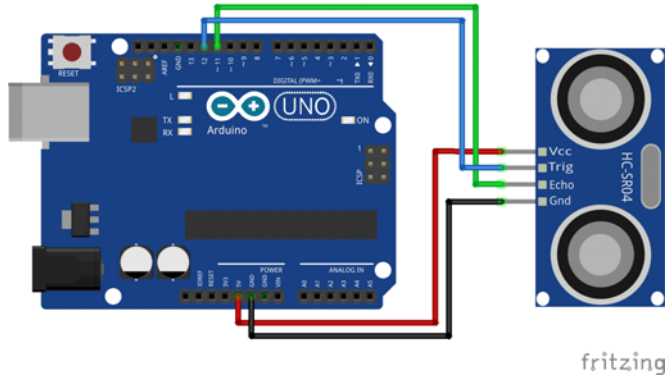
Sensoren finnes i forskjellige utgaver, men alle fungerer stort sett likt. Arduino-en sender en puls til triggerinngangen til sensoren (Trig). Sensoren sender ut en ultrasonisk puls omkring 40 kHz og venter på at det skal komme tilbake et ekko. Når ekkoet kommer tilbake, måles tidsforsinkelsen mellom utsendt og mottatt signal. Ut fra kjennskap til lydhastigheten i luft, kan avstanden til ekkokilden beregnes.

Sensoren har fire eller fem terminaler, vår har fire:

VCC	5,0 V
Trig	Triggersignal fra Arduino kobles til en digital utgang
Echo	Ekkosignal til Arduino kobles til en digital inngang
GND	GND (minus/jord)

Oppkobling

Figuren til høyre viser oppkoblingen. I dette eksempelet er Trig koblet til pin D12 og Echo returneres til pin D11. Men her står man fritt til å velge.



Blokkoder

Figuren til høyre viser de tre blokkene Blockuino tilbyr for å sette opp og lese av den ultrasoniske sensoren. Blokken "Initialiser Ultrasonisk Sensor" settes før setup() funksjonen, her hentes eventuelle biblioteker inn og alle nødvendige variabler deklareres som f.eks. hvilke pinner som kobles til "Trig" (f.eks. D9) og "Echo" (f.eks. D8). Dernest defineres "Trig" (D9) som OUTPUT og "Echo" (D8) som INPUT i blokken "Ultrasonisk setup". Tilslutt leser funksjonen "Les Ultrasonisk Avstand (cm)" av tidsforsinkelsen mellom "Trig" og "Echo" og beregner avstanden i cm som leveres fra denne funksjonen.

Verdien som leses fra sensoren settes inn i en variabel som vist i figuren under:

Her leses avstanden fra sensoren inn i variabelen "Avstand" i cm. Dersom man vil at noe skal skje bare når avstanden er mellom 10 og 20 cm kan man benytte en "Hvis"-setning hvor avstanden inngår i betingelsen.

3.9 Display

3.9.1 OLED display

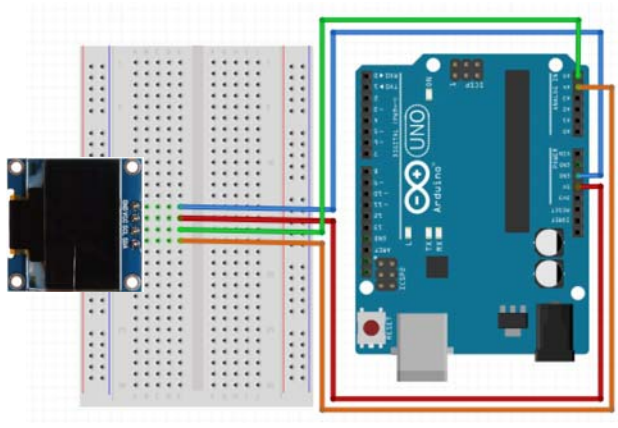
Funksjonen er tilpasset OLED display 32 x 128 pixler (0,96"). Displayet at fire terminaler:

VCC	3,0 - 5,0 V
GND	GND (minus/jord)
SCL	Serie klokkesignal (I^2C)
SDA	Serie datasignal (I^2C).

Displayet kommuniserer over en serielinje. Dette gjør det særdeles enkelt å koble opp. Driverkretsen er en SSD1316, det er viktig at displayet har nettopp denne driverkretsen da blokkene i Blockuino er tilpasset denne.

Oppkobling

Figuren til høyre viser oppkoblingen. Legg spesielt merke til at seriekommunikasjonen gjøres via A5 (SCL) og A4 (SDA).



Biblioteker

For at det skal være lett å programmere er det laget biblioteker av funksjoner. Disse må installeres i Arduino IDE'en før programmene kan kommunisere med displayet. Disse finnes flere plasser, men denne lenken synes gi en enkel å grei løsning.

https://learn.adafruit.com/monochrome-oled-breakouts/arduino-library-and-examples

Installasjon av biblioteker pakke som zip-filer er enkelt.

Åpne Arduino IDE

Velg Skisse/Inkluder bibliotek/Legg til ZIP: Biliotek fra menylinjen

Finn det aktuelle biblioteket i Nedlastinger katalogen. Velg den og biblioteket installeres automatisk.

Følgende biblioteker skal installeres:

Adafruit_GFX.cpp

Adafruit_SDD1306.cpp

Programmering

Følgende blokker leveres med Blockuino:

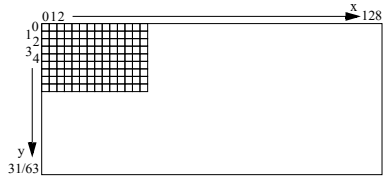
- "Inkluder OLED Biblioteket" inkluderer bibliotekene:
  - Adafruit_GFX.h (grafisk display)
  - Adafruit_SDD1306.h (spesial funksjoner for akkurat dette displayet)
  - Wire.h (seriekommunikasjon I^2C)
  - SPI.h (seriekommunikasjon SPI)

- Initierer OLED display og setter opp noen viktig parametere.

- Tømmer all informasjon på displayet slik at ny skriving kan begynne. Kommandoen må etterfølges av "Oppdater displayet".

- "Oppdater Display", denne funksjonen oppdaterer displayet i henhold til hva som er oversendt displayet

- "Set Cursor til X: og Y:". Denne funksjonen posisjonerer startpunktet for skriving på displayet. Utgangspunktet for tellingen er øverste venstre hjørne.

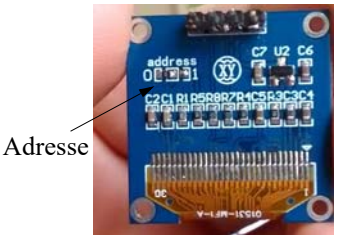


- "Skriv Tekst:" skriver den inntastede teksten ut på skjermen fra der cursoren er posisjonert. Enten kan man skrive teksten rett inn i blokken "Verdi" eller man kan generer verdien og legge den i en variabel.

- "Tegn Pixel" slår på et piksel på en angitt posisjon på skjermen. På denne måten kan en tegne grafikk. Også her kan man enten bruke bokken "Verdi:" eller "Variabel:".

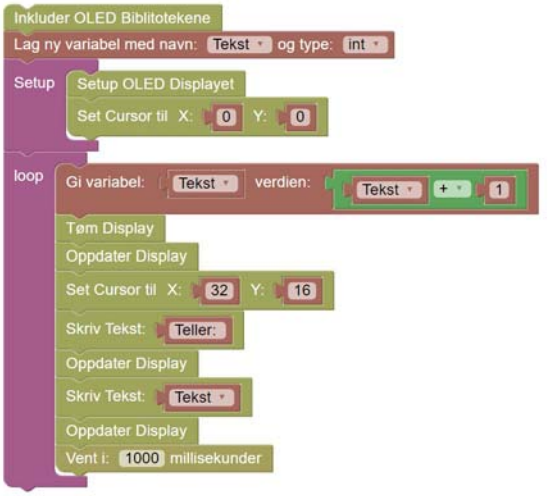
Adresse

Displayet har en adresse, på denne måten kan i prinsippet flere displayer kobles i parallell og adresseres individuelt. Det varier imidlertid om displayet har en fast adresse eller om den kan endres. På de som brukes her er adressen satt fast til 0x3C (ev. 0x3D). Dette er heksadesimale tall. På bildet til høyre er vist et display hvor adressen kan endres, ved å lodde ut eller inn forbindelser på ulike stder.



Eksempel

Eksempelet vist under er en blokkode som skriver ut en variabel som teller fra 1 og opp så langt en integer går.



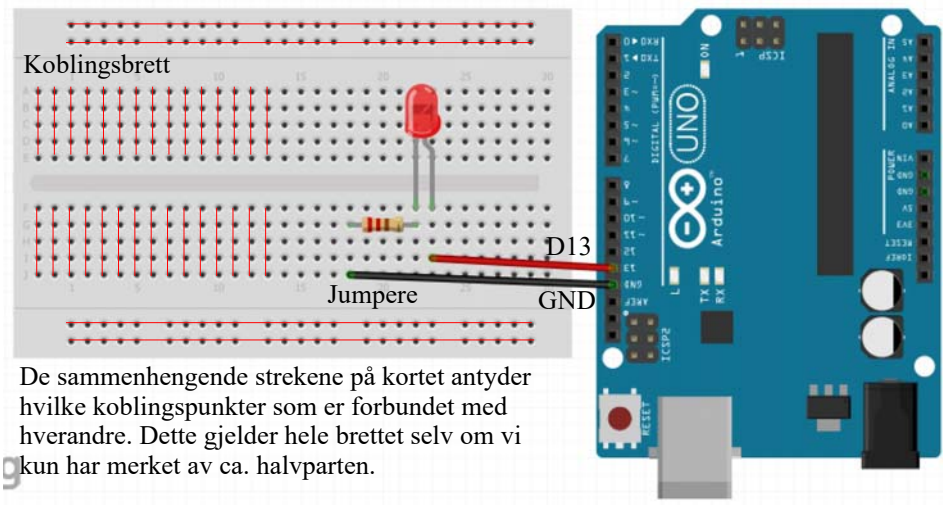
4 Øvingsoppgaver

Programmering er egentlig å lage en oppskrift for hva mikrokontrolleren skal gjøre. Den fungerer på mange måte som en bakeoppskrift hvor man begynner med den første oppgaven som står på lista for så å fortsetter nedover oppgave for oppgave. I det neste avsnittet skal vi bygge opp en programstruktur fra bunnen av Blockuino for en morsesender.

4.1 Blinklys – innledende øvingsoppgave

4.1.1 Oppkobling av Blinkende lysdiode

Det første vi må gjøre er å montere lysdioden med en motstand og forbinde disse elektrisk til Arduino mikrokontrollerkortet. Pluss på lysdioden (langt bein) kobles til Digital utgang D13 og minus på dioden (kort bein) kobles til minus eller jord (GND). For å begrense strømmen kobles en motstand på 330 Ohm (oransje, oransje, brun, gull) inn i kretsen. Bruk *jumpere* (ledninger) for å koble det hele sammen på koblingsbrettet som vist på figuren under.



De sammenhengende strekene på kortet antyder hvilke koblingspunkter som er forbundet med hverandre. Dette gjelder hele brettet selv om vi kun har merket av ca. halvparten.

4.1.2 Programmering - Tutorial

Oppgave:

Vi skal nå bygge opp programmet som skal få lysdioden til å blinke 500 ms på og 500 ms av.

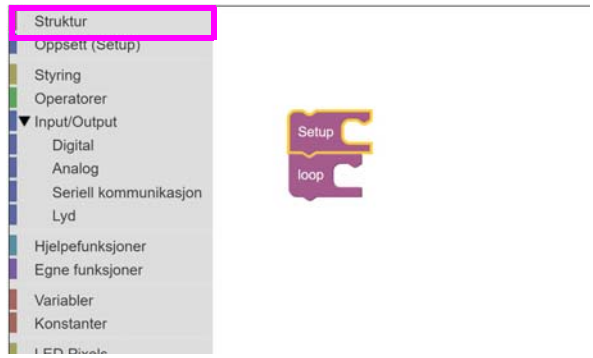
For å gjøre programmet mer oversiktlig deler vi det inn i små underprogrammer eller *funksjoner* som så kobles sammen. Hovedstrukturen består av to funksjoner:

- *setup()* - I setup-funksjonen plasserer vi kommandoer som bare trenger å bli utført når vi starter programmet og ikke ellers.

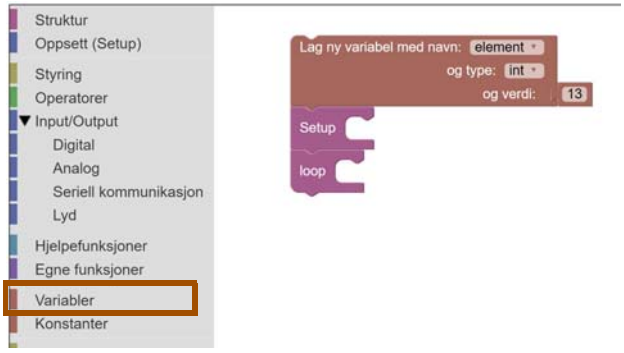
- *loop()* - I loop-funksjonen legger vi alle kommandoer som skal gjentas om og om igjen. Siden vi skal lage et blinklys så vil kommandoer som slår av og på lyset plasseres her.

La oss gå igang med å bygge opp programmet.

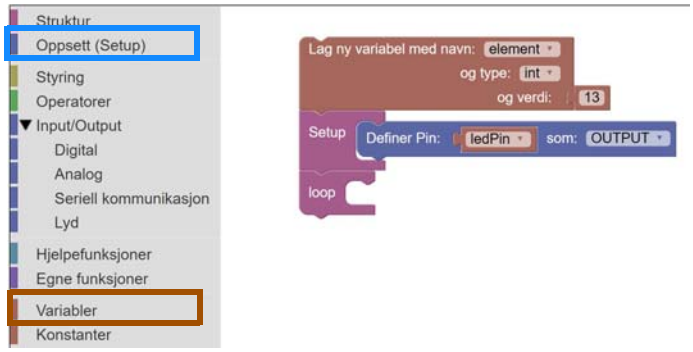
1. **Bygg opp program-strukturen**:
   Hent:
   setup() {}
   loop() {}
   fra menyen: ***Struktur***



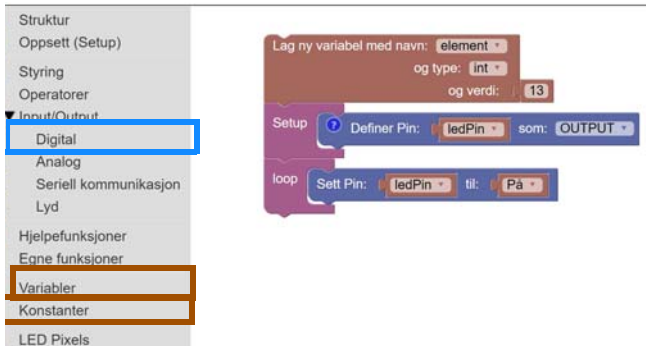
2. **Legg til en variabel.**
   Vi velger å gi den digitale porten *navnet*: *ledPin* med type *int* (heltall) og *verdi 13*:
   Hent: ***Lag ny variabel med navn: - og type: - og verdi: -***
   Fra menyen: ***Variabler***
   Her defineres variablene utenfor funksjonene setup() og loop(), dermed blir de *universelle*, dvs. at de gjelder innen i begge funksjonene.



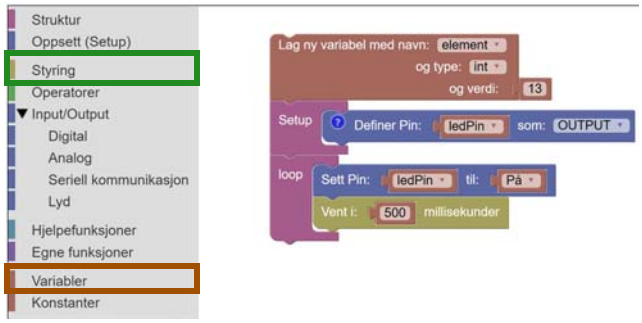
3. **Definer I/O port:** Her bestemmes om porten skal være en *inngang* eller en *utgang*.
   Hent: ***Definer Pin: - som: -***
   Fra menyen: ***Oppsett (setup)***.
   Siden pinnen har fått et navn velges:
   Og sett inn: ***Variabel element*** og velg element *ledPin*. Dermed har pinnen med navnet *ledPin* og verdien 13 blitt definert som en "OUTPUT".



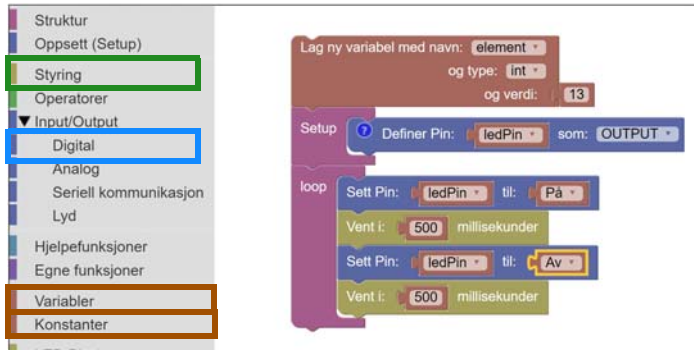
4. **Slå på lysdiode:** Siden lysdioden skal slås av og på i det uendelige så skal vi legge inn denne kommandoen i loopen.
   Hent: ***Sett pin: - til: -***
   Fra under menyen: ***Digital***
   Denne blokken gir mulighet til å velge en variabel med et gitt navn.
   Hent: ***Variabel element***
   Fra menyen: ***Variabler*** og velg "ledPin"
   Hent: ***På***
   fra menyen: ***Konstanter*** og velg "På" fra lista



5. **Vent i 500 ms:** Vi legger inn en vente-kommando som holder lyset på i 500 ms til vi igjen slår det av.
   Hent: ***Vent i 500 millisekunder***
   Fra menyen: ***Styring***
   Hent **Verdi** fra menyen ***variabler*** og skriv inn riktig antall millisekunder om det ikke alt står 500 millisekunder.



6. **Slå av og vent i 500 ms:** Her gjentar vi kommandoene i punkt 4 og 5, men sørger for å endre "På" til "Av". Slik at vi slår av lysdioden på pinne 13, venter i 500 ms før vi igjen går tilbake til toppen av loopen for så å slå lysdioden på igjen.



4.2 Morsesenderen

Vi skal nå bygge opp en morsesender som sender SOS, dvs. vi skal utvide funksjonen til blinklyset i avsnitt 4.1.2.

4.2.1 Spesifikasjon A: SOS

SOS som morse er: * * * - - - * * * (prikk, prikk, prikk, strek, strek, strek, prikk, prikk, prikk)

Prikklengde skal være: 200 msek

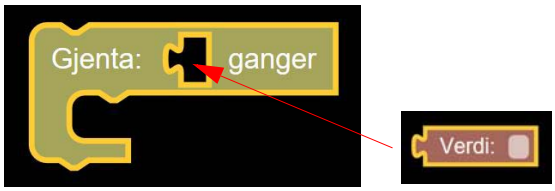
Streklengde skal være: 600 msek

Mellomrom mellom streker og prikker: 200 msek

Avstanden mellom bokstaver skal være: 600 msek

Avstanden mellom ord skal være: 1800 msek

Tips til programmeringen



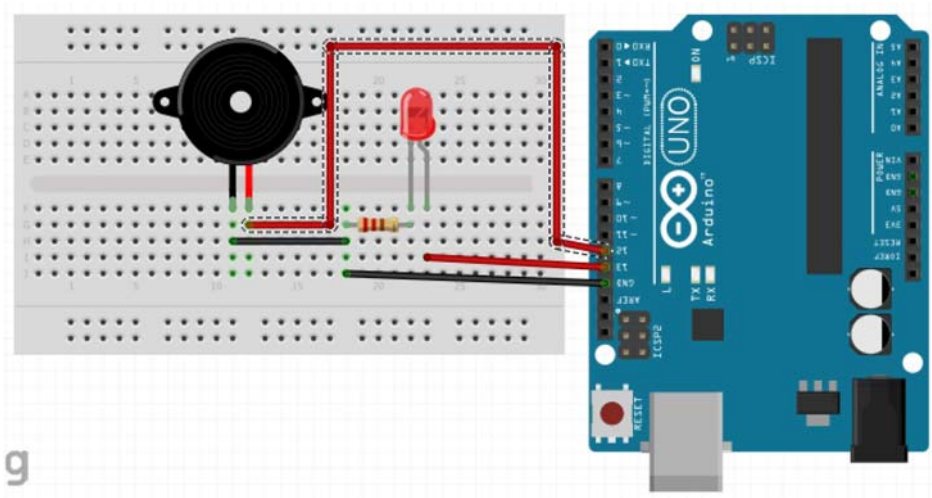
Siden vi skal lage 3 like prikker og 3 like strekker, kan det være lurt å bruke blokken **"Gjenta"** fra menyen **"Styring"**. For å sette antallet gjentagelser, benytter man blokken **"Verdi"** fra menyen **"Variabler"**. Antallet gjentakelser settes i input-boksen i **"Verdi"**.

Spesifikasjon B: SOS med lys og lyd

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

Oppkobling

Koble opp lydgiveren i tillegg til lysdioden. Vi velger å koble den positive ledningen fra lydgiveren til port D12 og jord (GND)



Tips til programmeringen

Bruk blokken "Spill tone". Her angis hvilken pinne tonen spilles fra, hvilken frekvens og hvor lang tonen er. Variablene "lyd" og "frekvens" må defineres. Det er også verdt å merke seg at lengden av tonen ikke forsinker programmet. For frekvensen kan det være lurt å velge konstanten "Tone:".

4.2.2 Spesifikasjon C: SOS med variabel hastighet

Det skal lages en SOS morsesender som spesifisert i B, men hvor det også skal være lett å variere sendehastigheten. Selv om hastigheten endres skal forholdet mellom lengden av *streker*, *prikker* og *mellomrom* være den samme.

Senderen skal kun sende lys.

Tips til løsning

For å lage en SOS-sender med variabel hastighet, så må vi kunne endre lengden av alle tegn og alle mellomrom slik at forholdet mellom lengden på mellomrom og tegn blir den samme. Dette kan vi gjøre ved å definere en variabel i starten (f.eks. "Speed") som styrer de aktuelle intervallene.

Dette kan vi gjøre på forskjellig måte. Her har vi laget en variabel venteløkke ved hjelp av blokken "Gjenta:" og "Vent i:", hvor Speed definerer hvor mange runder vi skal gå i løkka og dermed bestemme hvor mange multiplum av 40 ms vi vil at ventefunksjonen skal ta..

Et enkle alternativ er å sette en variabel rett inn i blokken "Vent i"

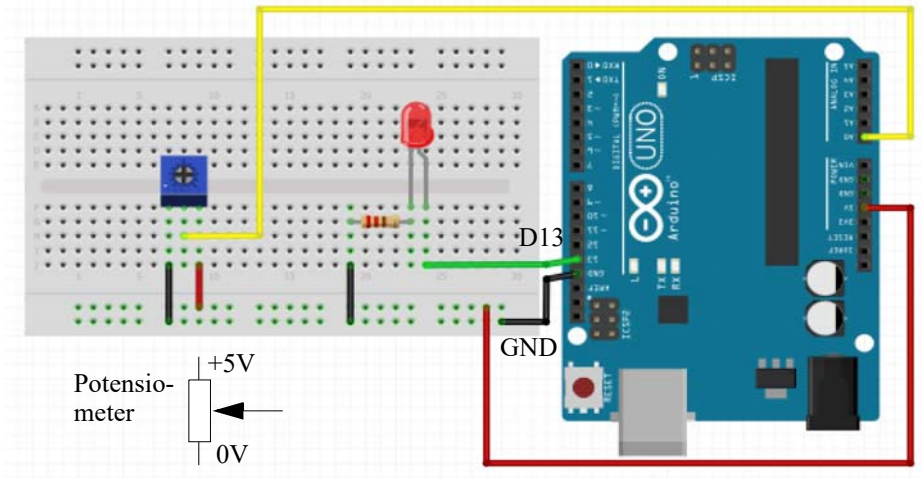
Dermed kan man lett endre hastigheten ved å gi variabelen "Speed" forskjellige verdier (se nederst på figuren til høyre).

4.2.3 Spesifikasjon D: Reguler hastigheten med en variabel motstand

I denne oppgaven skal vi koble opp en variabel motstand, et potensiometer, og bruke dette til å øke og redusere farten på mosesenderen.

Oppkobling av kretsen

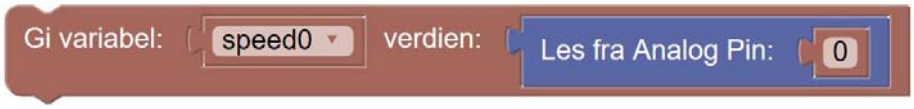
Vi bruker et potensiometer for å koble opp kretsen som vist på figuren under:



Potensiometeret er en motstand med en variabel tapping på "midten". Siden potensiometeret er koblet mellom 0 og 5 V så vil en mellom 0V og midtuttaket kunne hente ut fra 0 – 5 V avhengig av hvordan potensiometeret er innstilt.

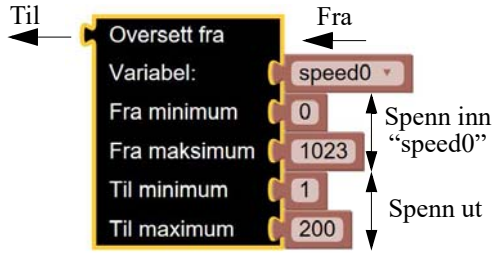
Tips til programmeringen

Kobler vi midtuttaket på potensiometeret til den analoge inngangen A0, så kan vi lese av spenning på analog inngang A0. Til dette bruker vi blokken "Gi variabel" fra menyen "Variabler".:



En analog spenning inn på pinne A0 vil når den leses, gi en verdi fra 0 – 1023, hvor 0 tilsvarer 0 V og 1023 tilsvarer 5 V. Denne verdien leses inn i variabelen "speed0" som er deklarert øverst i programmet. Kommandoen "Les fra Analog Pin 0", hentes fra menyen "Maskinvarekontroll".

Sidene verdiene 0 – 1023 kan være litt upassende for å styre farten i morsekoden, så kan det være lurt å konvertere 0 – 1023 til f.eks. 1 – 200. Dette kan vi gjøre med blokken "Oversett fra" som finnes i menyen "Funksjoner" (se figuren til høyre).



Denne konverterer ett spenn i verdiområdet fra "fra minimum" – "fra maksimum" hos variabelen "speed0", til et spenn fra "til minimum" – "til maksimum" hos verdien som kommer ut av funksjonen.

4.3 Trafikklys

I denne oppgaven skal vi lage et trafikklys, dvs. en stolpe med rødt, gult og grønt lys.

Det skal lages et trafikklys etter følgende spesifikasjoner:

4.3.1 Spesifikasjon A – Programmer et trafikklys:

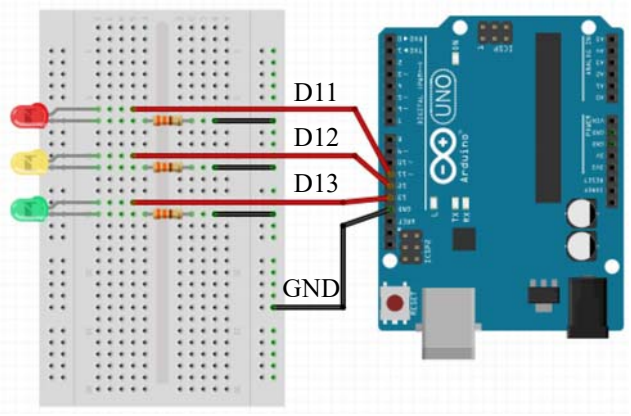
- *Bestem rekkefølgen på lysene i et trafikklys når lyset skifter fra rød til grønt og fra grønt til rødt.*
- *Programmer trafikklyset slik at: Det er rødt i 5 sek, gult i 1 sek og grønt i 5 sek.*
- *Koble opp trafikklyset med:*
  Rød, gul og grønn lysdiode og skriv programmet

Oppkobling av kretsen

I dette eksempelet er:

- Rød koblet til D11
- Gul koblet til D12
- Grønn koblet til D13

Figuren til høyre viser oppkoblingen av trafikklyset.



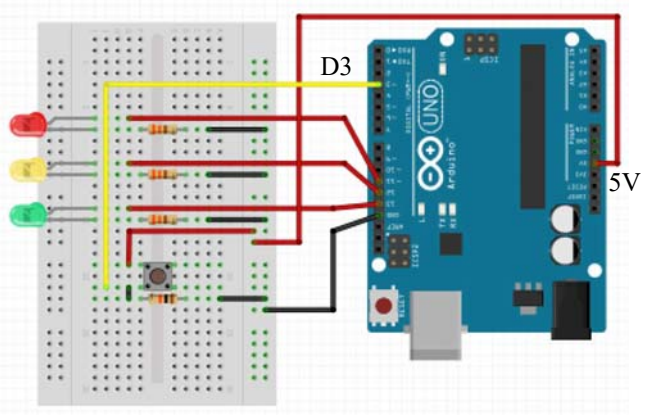
Tips til programmeringen

Tenk nøye etter hvordan lysene skifter fra rødt til grønt og fra grønt til rødt.

4.3.2 Spesifikasjon B – Grønt lys på forespørsel:

- *Endre sekvensen i spesifikasjon A slik at trafikklyset blir stående på rødt lys, helt til noen trykker på fotgjengerknappen.*
- *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 fotgjengerknappen.*
- ***Nyttige opplysninger:***
  - Bruk trykknappen som følger med settet

Oppkobling av kretsen



Figuren til høyre viser oppkoblingen av kretsen med en bryter. Bryteren er koblet til digital inngang D3. Legg merke til motstanden som er koblet mellom bryteren og jord. Dette gjør at inngangen D3 normalt vil ligge til jord så lenge bryteren ikke er trykket inn, men vil gi en positiv verdi (5 V) så lenge bryteren er koblet inn. Grunnen til at man gjør det slik er for å unngå at inngang D3 henger i "løse luften" når bryteren ikke er trykket inn, dermed unngår en at den gule ledningen (se figuren over) samler opp støy.

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

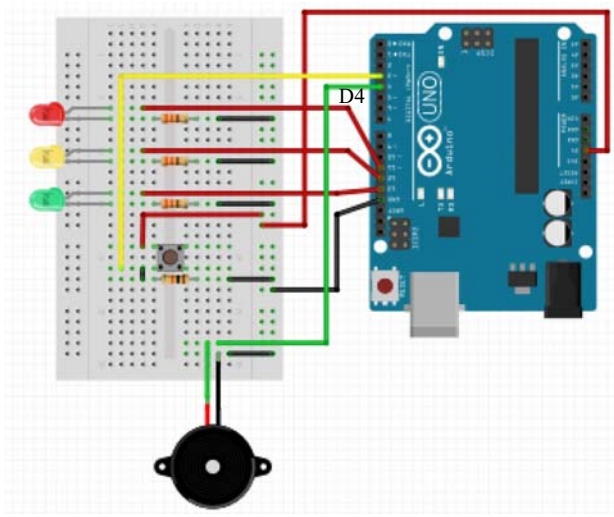
- *Endre programmet fra oppgave B slik at det blir grønt lys i 8 sek. hvorav de siste 5 sek. skal være blinkende. Hvert blink skal være 500 ms på og 500 ms av. Trafikklyset skal ellers oppføre seg som i spesifikasjon B.*

4.3.4 Spesifikasjon D – Blinkende grønt lys med lyd:

- *Hvert grønt lysblink skal etterfølges av et kort pip på 100 ms for synshemmede.*

Oppkobling av kretsen

Man kobler til en ekstra piezoelektrisk lydgiver til utgang D4 (+). Mens den negative ledningen kobles til GND (minus/jord).

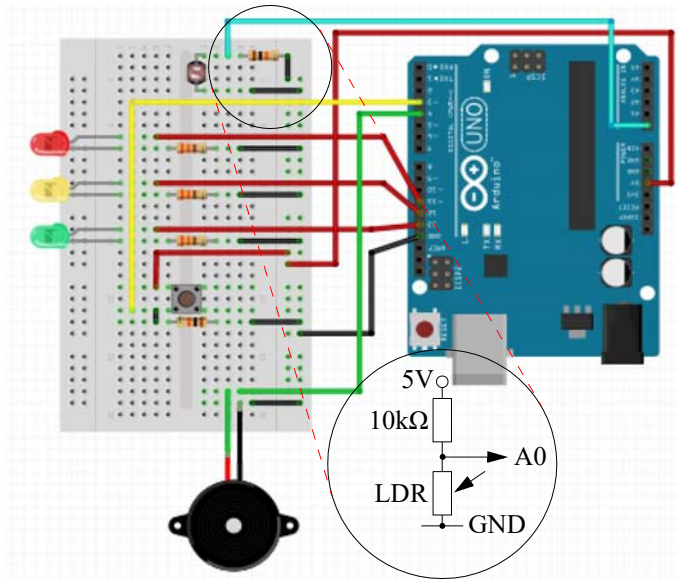


4.3.5 Spesifikasjon E – Blinkende gult lys etter mørkets frambrudd:

- *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.*
- *Trafikklyset skal ellers oppføre seg som beskrevet foran når det er lyst.*

Oppkobling av kretsen

I denne delen av oppgaven skal vi benytte en lysfølsom motstand (LDR) til å starte det blinkende gule lyset.

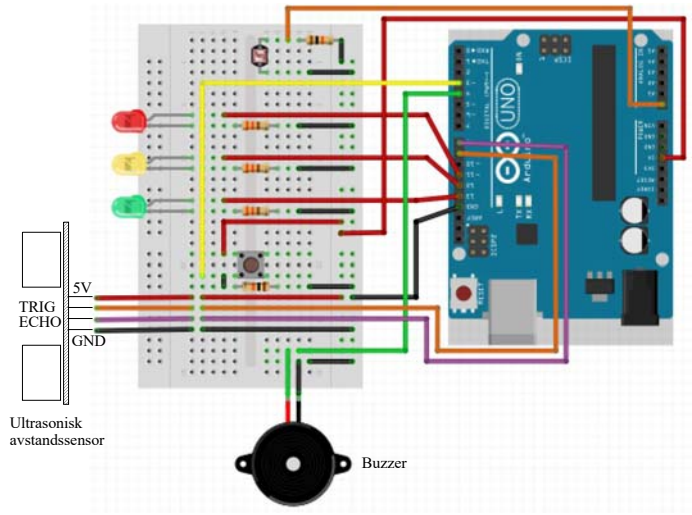


Nederst i tegningen til høyre har vi tegnet opp skjema for oppkoblingen av den lysfølsomme motstanden. Vi ser at den er koblet i serie med en motstand på 10 kΩ. Når LDR'en belyses, reduseres motstandverdien i komponentene og spenningen på A0 nærmer seg GND (–), dvs. den målte verdien blir lav. Når LDR'en dekkes til slik at den blir mørkt, vil motstandsverdien øke i komponenten og den målte spenningen på A0 vil også øke. Ved å legge terskelen riktig kan vi bruke LDR'en til å slå på det blinkende gule lyset når mørket faller på.

4.3.6 Spesifikasjon F – Det gis grønt lys dersom noen nærmer seg lyset

- *Det planlegges en modernisering av trafikklyset. I stedet for at fotgjengerne skal trykke på knappen skal det være tilstrekkelig at de blir stående nærmere enn 30 cm fra lyset, da skal de få grønt lys.*
- *Bruk en ultralyd sensor, bygg opp kretsen og lag et program som oppfyller disse betingelsene. Man kan godt bytte ut trykknappen med ultralydsensoren. Eller beholde begge funksjonmene.*

Oppkobling av kretsen



Vi har flyttet litt på noen komponenter for å få plass til en ultrasonisk avstandssensor. Denne fungerer slik at den sender ut en ultrasonisk puls på ca. 40 kHz som reflekteres fra et mål et stykke unna sensoren. Siden lyd beveger seg med en fart lik ca. 340 meter i sekundet, vil det ta litt tid før et ekko kommer tilbake og fanges opp av sensoren. Med en kjent lydhastighet kan man så beregne avstanden på grunnlag av tidsforsinkelsen til ekkoet. I tillegg til 5V og jord (GND/minus), har den to terminaler: "Trig" som vi kobler til port D9 på Arduinoen og "Echo" som vi kobler til port D8.

Tips til programmeringen

Blockuino har en egen ferdig definert funksjon som kan brukes til å måle avstanden til en reflektor fra sensoren. Denne er vist på figuren til høyre og består av tre blokker. Blokken "Initialiser Ultrasonisk Sensor" settes før setup() funksjonen, her hentes nødvendige biblioteker inn og alle nødvendige variabler deklareres som f.eks. hvilke pinner som kobles til "Trig" (D9) og "Echo" (D8). Dernest defineres "Trig" (D9) som OUTPUT og "Echo" (D8) som INPUT i blokken "Ultrasonisk setup". Tilslutt leser funksjonen "Les Ultrasonisk Avstand (cm)" av tidsforsinkelsen mellom "Trig" og "Echo" og beregner avstanden i cm som leveres fra denne funksjonen.

4.3.7 Spesifikasjon G – To eller flere 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 Arduino-sett?*

Til denne oppgaven er det ikke laget noe løsningsforslag.

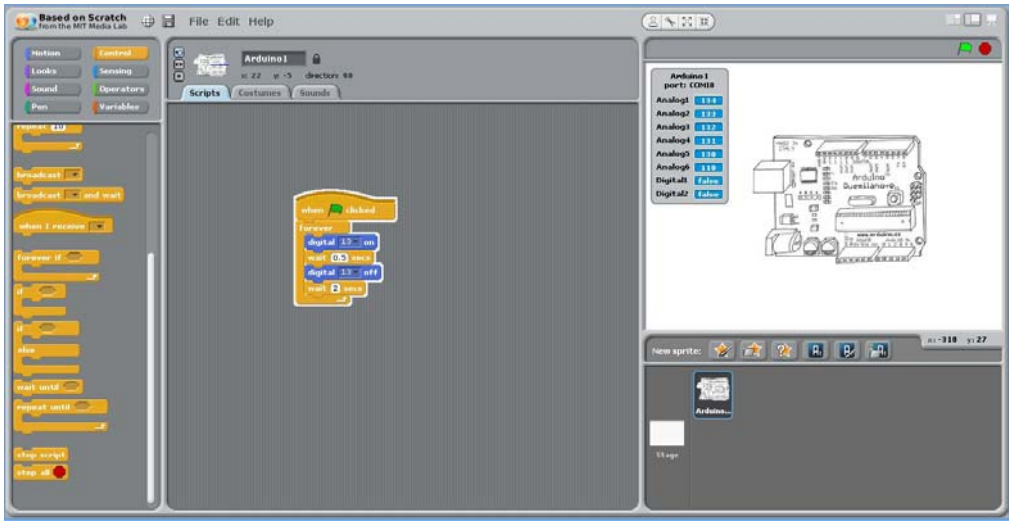
5 Andre blokkprogrammeringsverktøy

5.1 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 programmere 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 mikrokontrollerkortet. Programmene kan derfor ikke kjøres med mindre kabelen er tilkoblet.

La oss først se hvordan vi skal installere programmet.

5.1.1 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 programmene fra *S4A*. Denne programvaren går under navnet "firmware".

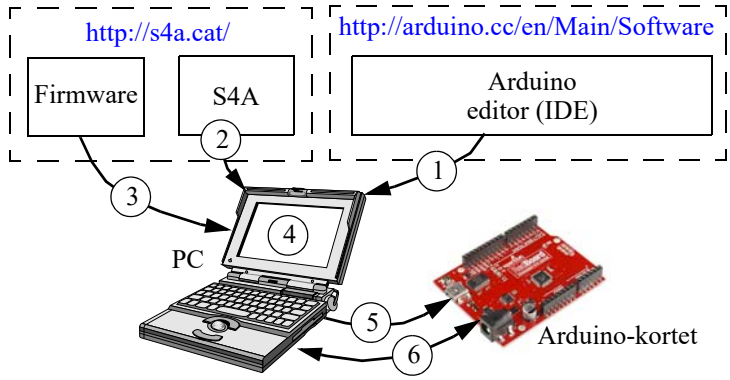
6. **Nedlasting av *Firmware***
Firmware er en Arduino programfil skrevet i Arduino C (*S4AFirmware15.ino*). Denne lastes ned og legges på et sted hvor du lett kan finne igjen den.

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 Arduino-kortet på vanlig måte.
6. S4A startes og skal nå automatisk koble seg opp mot Arduino-kortet og starte programmet.

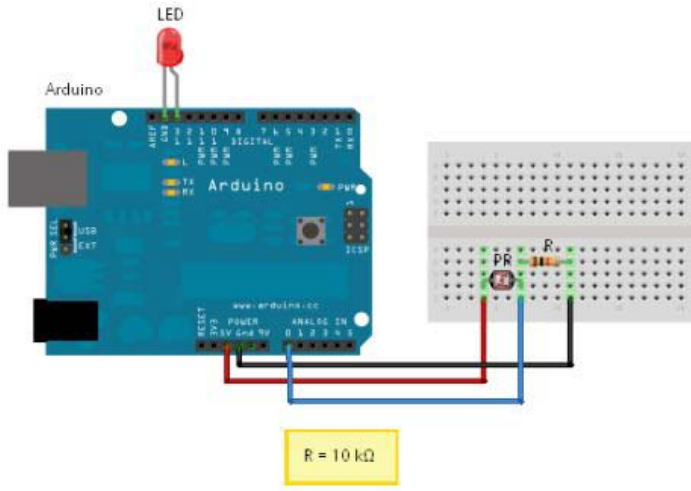
Normalt vil programmer som overføres til Arduino-kortene kunne kjøres på kortet uavhengig av PC-en og vil, når kortet får spenning, kunne kobles fra PC-en. Slik er det *ikke* med Scratch for Arduino, her kjøres programmet i PC-en, som så sender kommandoer over til firmwaren som utfører kommandoene og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

Ord og begreper[4]:

- **IDE** - *Integrated Development Environment* - utviklingsverktøy 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.

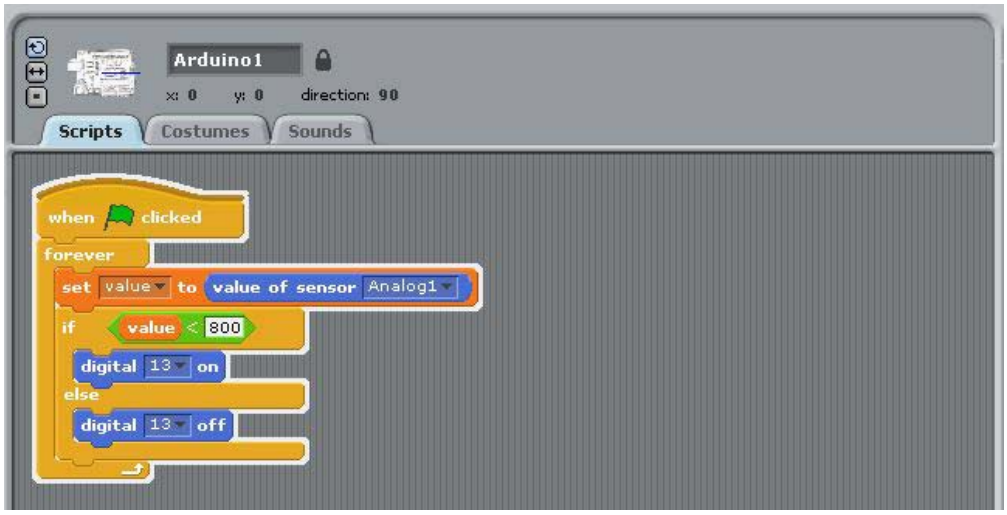
5.1.2 Mitt første program i *S4A* (tutorial)

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:



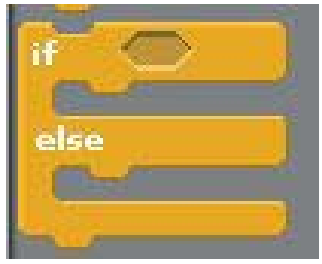
4. Skrevet av Åge Eriksen.

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



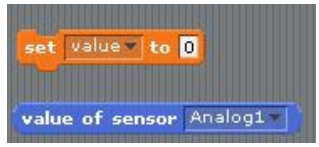
- **"When click on flag"**
  Ved å starte programmet med dette flow-symbolet vil programmet starte når man trykker på det grønne flaget, enten på ikonet eller på det grønne flagget øverst lengst til høyre i skjermbildet.

- **"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.

5.1.3 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 (kontinuerlig 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.

5.2 Ardublocks

Ardublocks er et grafisk programmeringsverktøy 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:

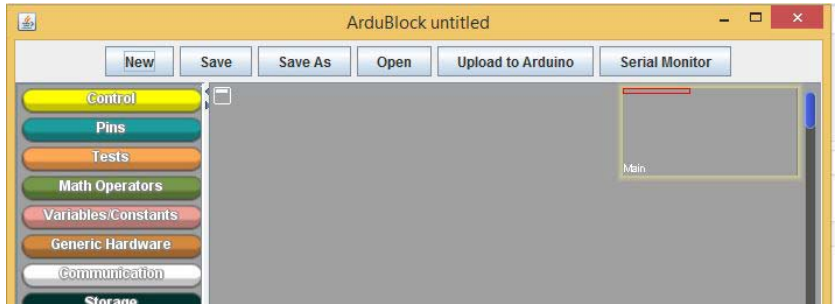
- Ardublocks er integrert i Arduino's editor (IDE) og kan velges inn fra Tools-menyen
- Det grafisk skrevne programmet oversettes til Arduino C-kode og vises i IDE etter at det er kompilert. På denne måten kan en lett se sammenhengen mellom Ardublocks-kommandoer og de grafiske programmeringselementene
- 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 Scratch har.

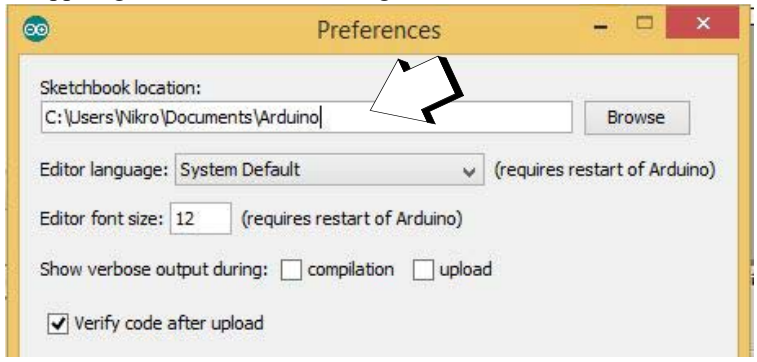
5.2.1 Installasjon

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:



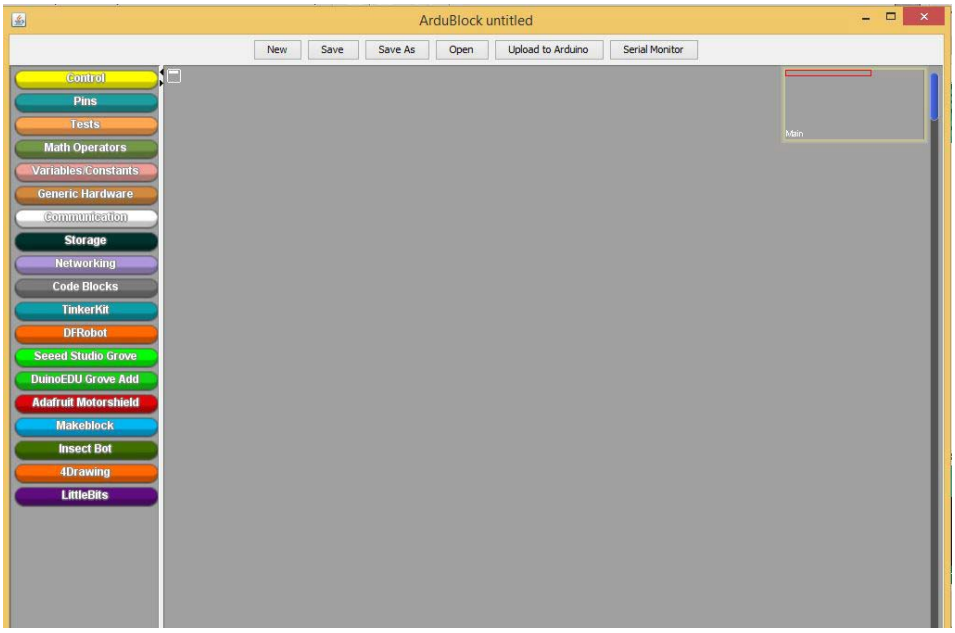
4. Vi skal nå integrere denne i den tradisjonelle Arduino editoren (IDE)
   Åpne Arduino editoren og velg: ***File***
   Deretter velg: ***Preferences***
   Du vil da få opp følgende bilde som viser deg: ***Sketchbook location***



   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



5.2.2 Bruk av Ardublocks

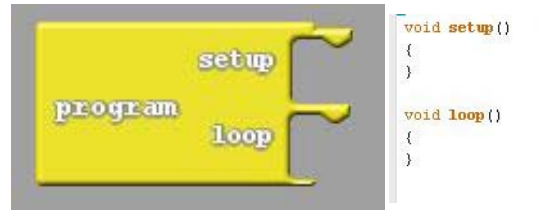
Vi skal nå ganske kort vise hvordan man kan bygge opp et program ved hjelp av Ardublocks. I menyen langs venstre side er de ulike blokkene organisert i grupper. Disse er de viktigste gruppene:

- ***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 "pullup" 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 tillegg kan du definere *arrays*. Selv om menyen inneholder også noen konstanter (f.eks. 3.14) så kan den ennå ikke definere konstanter slik overskriften synes å love.
- ***Communikasjon* (hvit)**
  Her finner man alle kommandoer som kommuniserer med monitoren (seriell datalinje) og I^2C porten.
- ***Code blocks* (grå)**
  Her finner du 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 veriabelnavn 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 funksjonens navn. Funksjonskallet kan godt komme før funksjonen defineres.

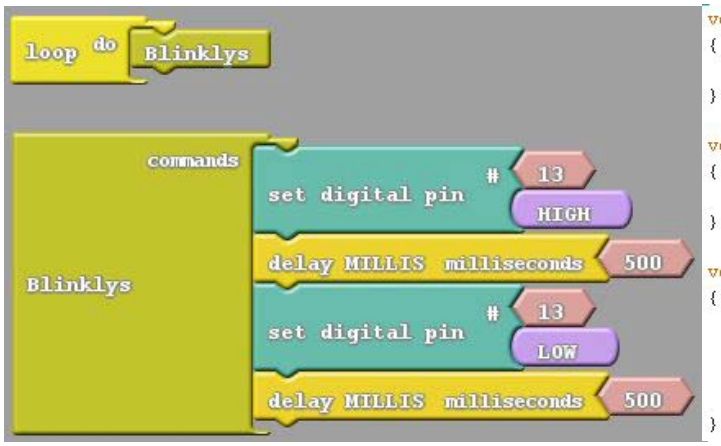
Eksempel 1: Blinklys

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 blinkfunksjonen 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.



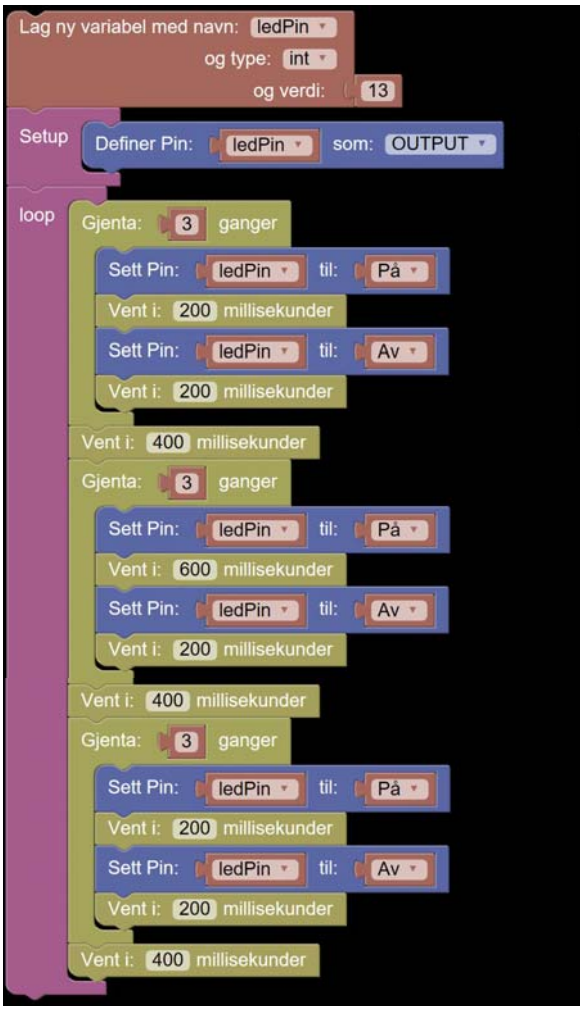

```
void setup()
{
  pinMode( 13 , OUTPUT);
}

void loop()
{
  Blinklys();
}

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

Vedlegg A Løsningsforslag

A.1 SOS morsesender

A.1.1 Løsningsforslag: Morsesender – spesifikasjon A

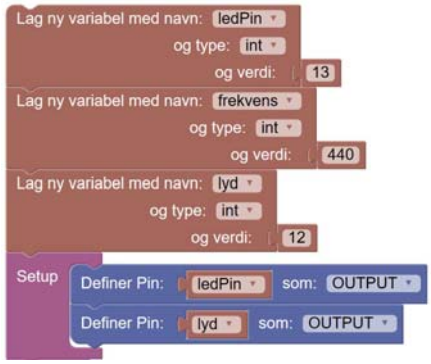
Figuren under viser en mulig løsning for spesifikasjon A



A.1.2 Forslag til løsning: Morsesender – spesifikasjon B

Vi velger å legge inn lyden hver gang lyset slås på. Dernest setter vi inn lengden av tonen lik lengden av prikker og streker.

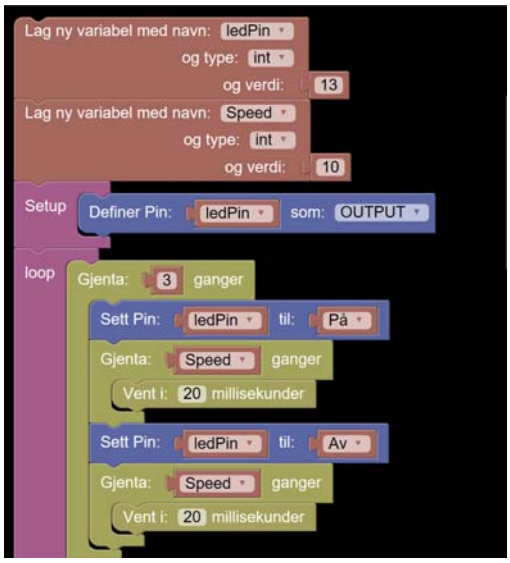
Figuren over viser deklarasjon av variablene "ledPin", "frekvens" og "lyd".

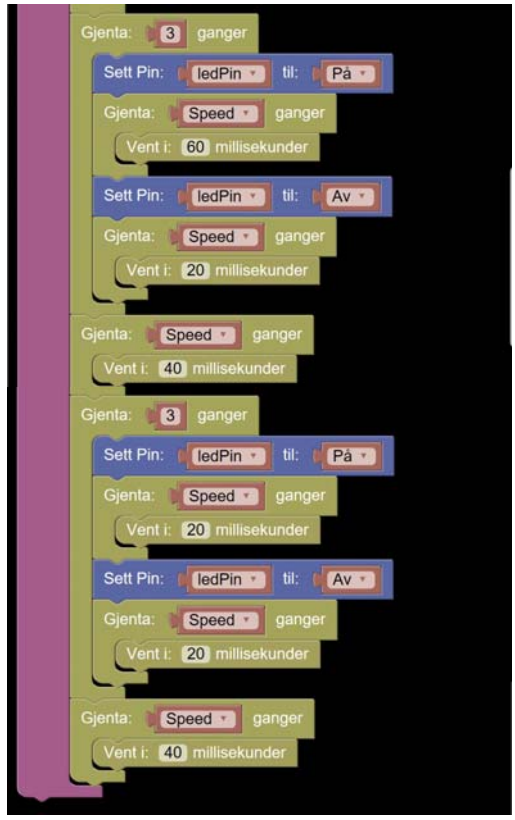


Figuren over viser loopen.

A.1.3 Forslag til løsning: Morsesender – spesifikasjon C

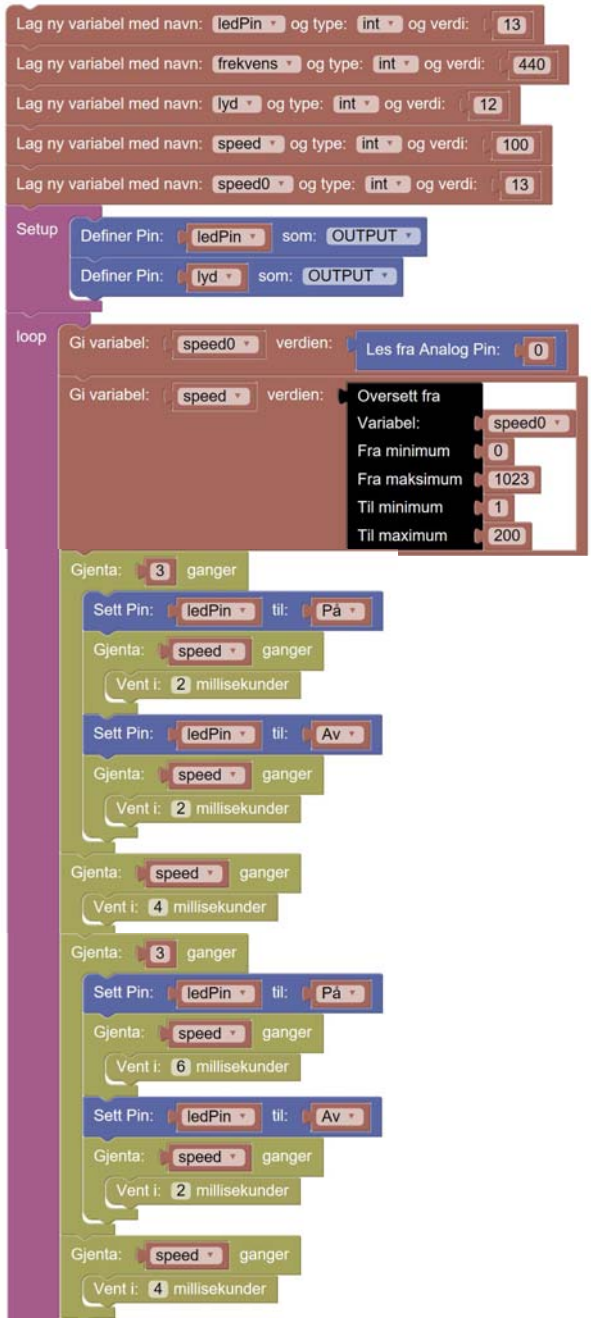
Blokkoden i figuren under viser et forslag til løsning av spesifikasjon C.

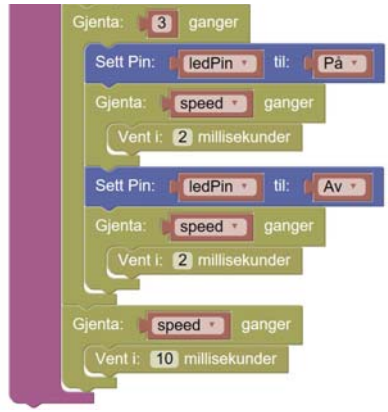




A.1.4 Forslag til løsning: Morsesender – spesifikasjon D

Blokkoden i figuren under viser et forslag til løsning av spesifikasjon D.





Det er lagt inn litt lengre forsinkelse på slutten av loopen for å indikere et skille mellom hvert SOS-signal.

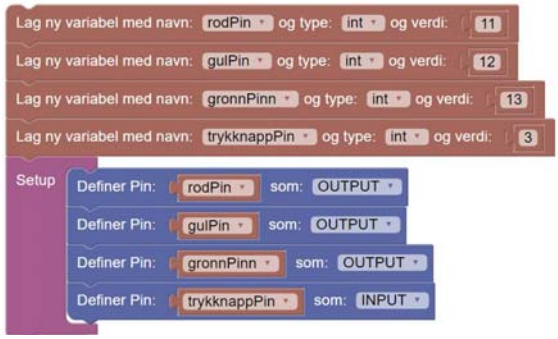
A.2 Trafikklys

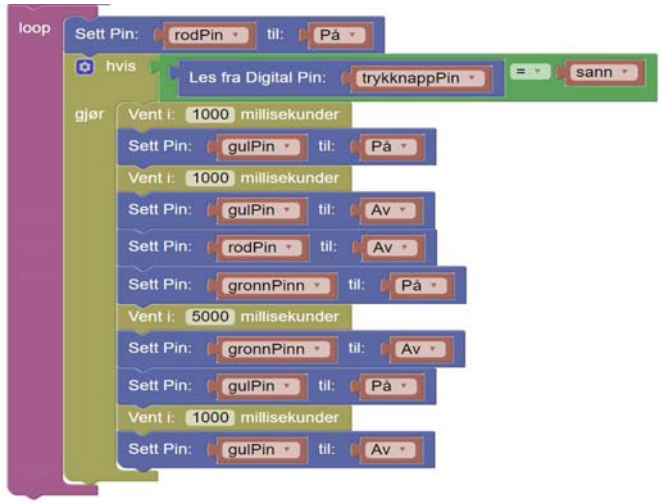
A.2.1 Forslag til løsning: Trafikklys spesifikasjon A

Blokkoden i figuren under viser et forslag til løsning av spesifikasjon A

A.2.2 Forslag til løsning: Trafikklys spesifikasjon B

Figuren under viser forslag til løsning av spesifikasjon B.

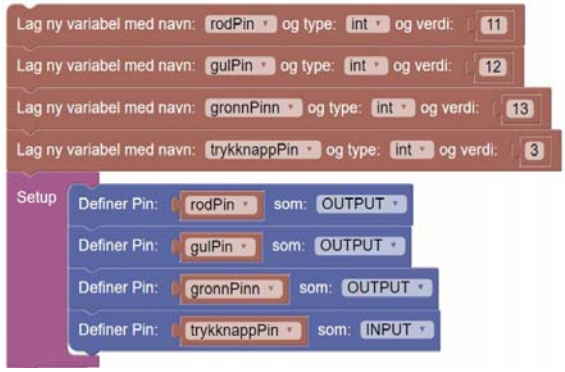


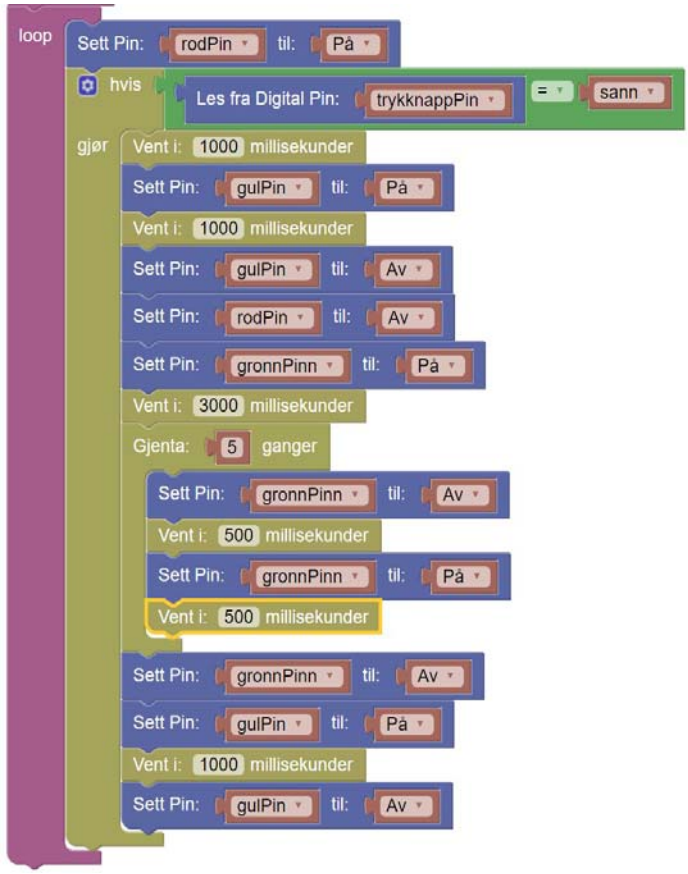


Legg spesielt merke til at tilstanden til knappen leses direkte inn i betingelsen til blokken “hvis”. På denne måten sparer vi en linje i programmet og deklarasjon av en variabel.

A.2.3 Forslag til løsning: Trafikklys spesifikasjon C

Blokkoden i figuren under viser et forslag til løsning av spesifikasjon C.

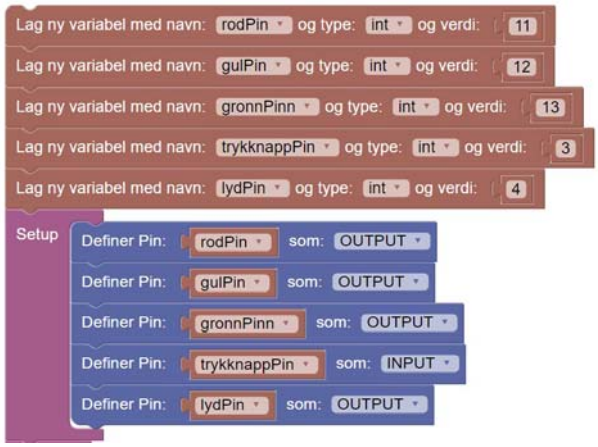




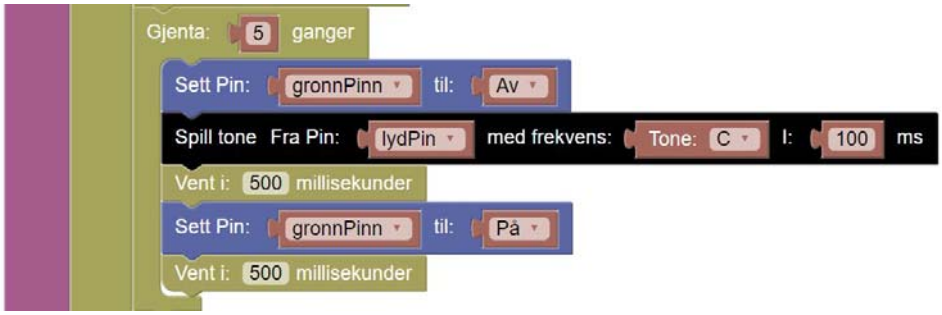
A.2.4 Forslag til løsning: Trafikklys spesifikasjon D

Blokkoden i figuren under viser et forslag til løsning av oppgaven.

Vi må først sørge for å definere en utgang som skal levere lydsignaet til lydkilden.



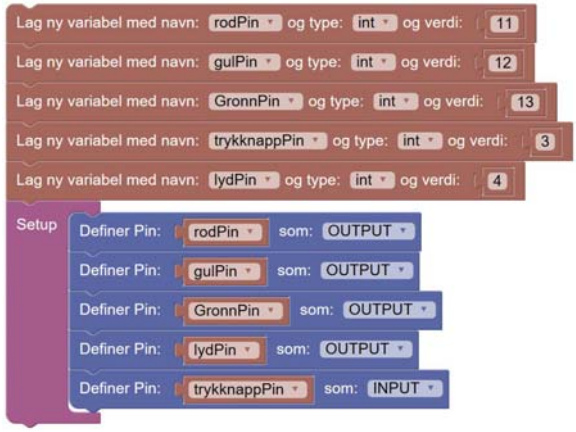
Deretter legges tonen inn idet det blinkende grønne lyset tennes.



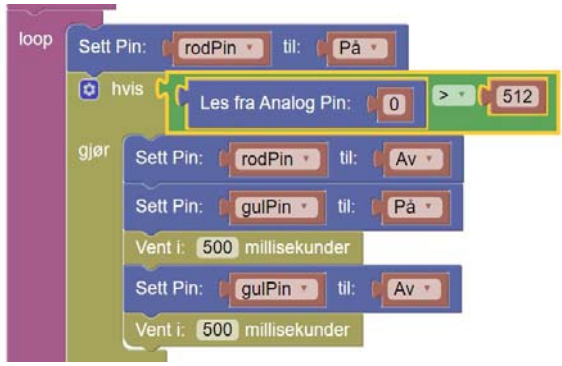
Figuren over viser kun det utsnittet av blokkoden hvor lyden er satt inn. Blokken "Spill tone" er en funksjon hvor vi overfører noen parameterer til funksjonen. Dette gjør vi med å fortelle hvilken pinne lyden skal sendes til ("lydPin" – D4), hvilken tone denne skal spille av ("Tone C") og hvor lang tonen skal være ("100 ms"). Koden gir dermed et 100 ms pip for hver gang lysblinket slås på. Merk at funksjonen "Spill tone" ikke stopper resten av koden, men at programmet går videre, mens tonen spilles av.

A.2.5 Forslag til løsning: Trafikklys spesifikasjon E

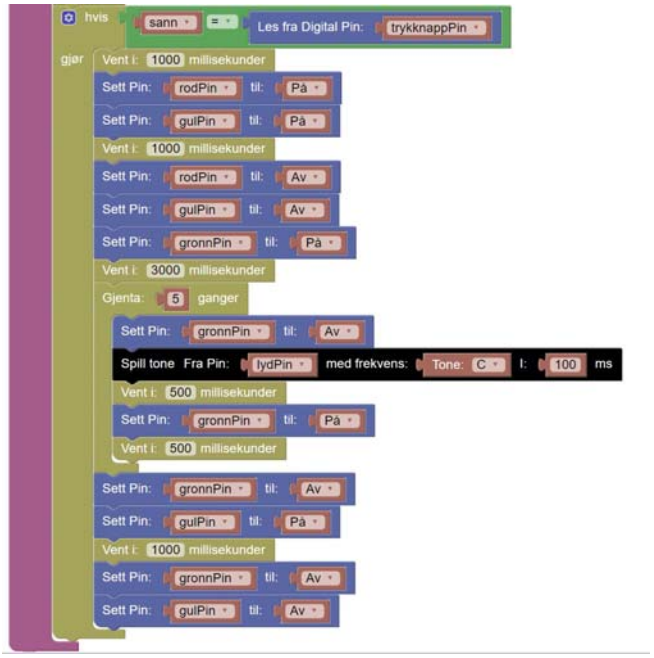
Blokkoden i figuren under viser et forslag til løsning av oppgaven.



Figuren over viser deklarasjon av variabler og setup () funksjonen som definerer de aktuelle digitale pinnene som utganger. Legg merke til at vi trenger ikke knytte en variabel til den analoge inngangen A0 fordi den automatisk ansees som en inngang. Vi kan dessuten legge lesefunksjonen direkte inn i betingelsen til "hvis-setningen". Vi har valgt å bruke to "hvis-setninger" i stedet for en "hvis-gjør-ellers-hvis-gjør-ellers-setning". Det ene er like bra som det andre.



Blokkoden i figuren over viser øverste del av loop() funksjonen. Først slås den røde lysdioden på, deretter brukes en "hvis" blokk for å sjekke hvor mørkt det er nær LDR'en. Dersom det er mørkt nok, dvs. verdien på A0 er høy nok (> 512), går programmet inn i "hvis" funksjonen og starter blinkingen.

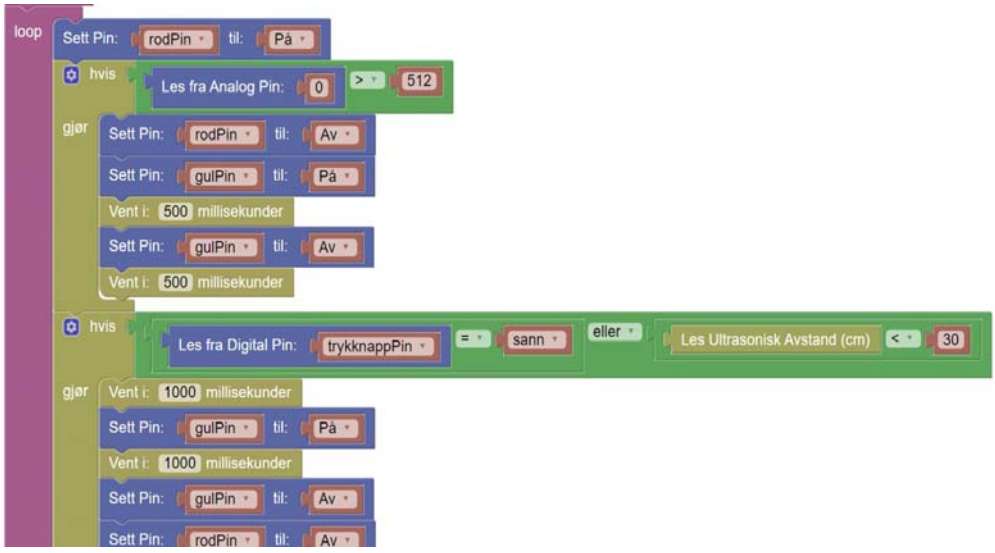


Blokkoden i figuren over viser siste del av loop() funksjonen. Her testes om knappen er trykket inn. Om det er tilfelle, går programmet inn i “hvis” funksjonen og gjennomgår en syklus med grønt lys, inkludert lyd.

A.2.6 Forslag til løsning: Trafikklys spesifikasjon F

Blokkoden i figuren under viser et forslag til løsning av spesifikasjon F.

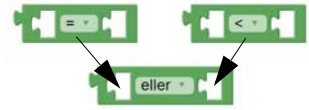
I figuren over ser vi hvordan funksjonen "Initialiser Ultrasonic Sensor" er lagt inn før funksjonen "Setup", her finner vi bl.a. bibliotekene som kreves for å lese avstandssensoren. Også funksjonen "Ultrasonic setup" er lagt inn i "Setup"-funksjonen.



Ellers er hele endringen bakt inn i betingelsen til “hvis”-setningen ved at vi først benytter en logisk “eller” operator for deretter å utstyre denne med to komparative operatorer, “=” for trykknappen og “<“, for den ultrasoniske sensoren. Dermed vil programmet gå inn i “hvis” funksjonen dersom enten knappen er trykket eller noen er kommet nær nok den ultrasoniske sensoren.

I dag er noen fotgjengeroverganger nettopp utstyrt på denne måten, ved at gule lys begynner å blinke når en fotgjenger kommer nær lystene ved fotgjengerovergangen.

To komparative operatorer



En logisk operatorer

Sparkfun Inventor's kit V3.3

Heftet er ment som et ressurshefte for den som ønsker å anvende blokkprogrammering av Arduino. Heftet vil gradvis bli utviklet dersom man ser at det er et marked for denne typen opplæring.

Nils Kr. Rossing
Førstelektor ved Skolelaboratoriet og
E-post: nils.rossing@ntnu.no
Prosjektleder ved Vitensenteret
E-post: nkr@vitensenteret.com

NTNU
Trondheim

Institutt for lærerutdanning

Skolelaboratoriet
for matematikk, naturfag og teknologi

Tlf. 73 55 11 43
Faks 73 55 11 40
http://www.ntnu.no/skolelab

Trykk: NTNU-trykk