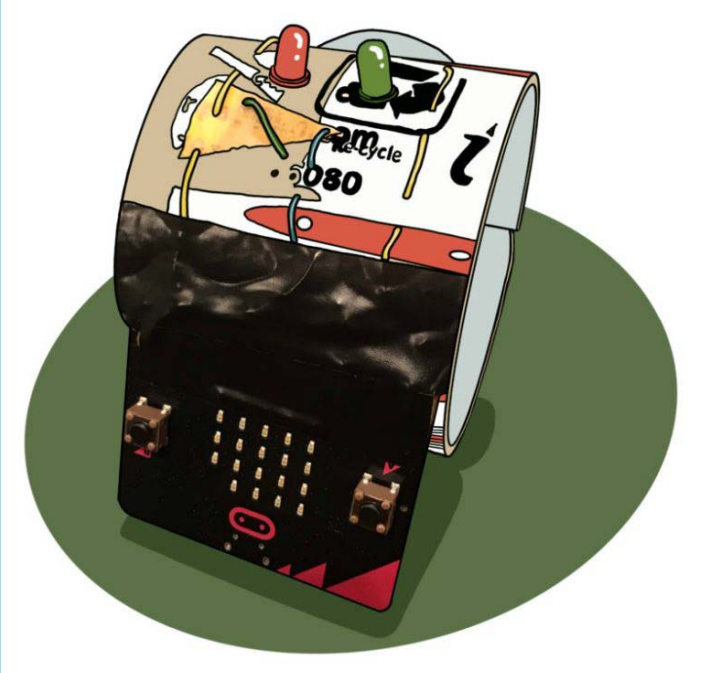
VITENSENTERET
TRONDHEIM

Rev 1.0

Nils Kr. Rossing, Ola Kleiven, Anne-Birgitte Belboe, Rannvei Sæther, Eva H. Hagen

Videregående kurs Micro:bit DeKom



November 2020

Denne siden er blank

Videregående kurs Micro:bit DeKom

Nils Kr. Rossing, Ola Kleiven, Anne-Birgitte Belboe,
Rannvei Sæther, Eva H. Hagen

Grunnkurs Micro:bit – DeKom

Trondheim 2020

ISBN 978-82-92088-

Bidragsytere:

Layout og redigering: *Nils Kr. Rossing, Vitensenteret i Trondheim*

Tekst og bilder: *Nils Kr. Rossing, Vitensenteret i Trondheim*
Rannvei Sæther, Vitensenteret i Trondheim
Ola Kleiven, Vitensenteret i Trondheim
Anne Birgitte Belboe, Vitensenteret i Trondheim
Eva H. Hagen, Vitensenteret i Trondheim

Faglige spørsmål rettes til:
Vitensenteret i Trondheim
v/Nils Kr. Rossing
nkr@vitensenteret.com

Kongensgate 1
7011 Trondheim

Postboks 117
7400 Trondheim

Vitensenteret i Trondheim
Telefon: 72 90 90 07
http://www.vitensenteret.com/

Rev 1.1 – 10.11.20

Forside bilde: Tegning - Skaperskolen

Forord

Heftet er skrevet som en hjelp til gjennomføring av 4. samling av DeKom-tilbudet: *Skapende aktivitet i klasserommet*, som ble gitt til Okstad skole, Tomasskolen og Vikhammer/Vikhammeråsen skole høsten 2020.

Målsetingen med denne tredje og fjerde samlingen er å gi deltakerne en grunnleggende innføring i programmering med micro:bit samtidig som det skal være et eksempel på hvordan en kan bruke programmering for å bygge opp et produkt (prosjekt). Det er lagt vekt på sentrale begreper fra læreplanen og å vise hvordan man kan lage og teste små programbiter som etter hvert kan settes sammen til et større program.

Den 4. samlingen bygger på samling 3 og utforske to viktige egenskaper ved mikto:bit: Trådløs kommunikasjon og det innebygde akselerometeret. Begge disse egenskapene er knyttet til et gjennomgående prosjekt – *Smart armbånd,* hvor en kombinerer den håndverksmessige med det programmeringstekniske som vi anser som særdeles viktig. I tillegg vil vi omtale begrepet variabler som både er sentralt i programmering, men likevel er vanskelig å forstå for mange som møter programmering for første gang.

Heftet er ment som en støtte under arbeidet på kursdagen, men mest som en hjelp i det etterfølgende arbeidet i klasserommet, dog ikke uten videre for utdeling til elevene.

Tilbudet er initiert av Trondheim og Malvik kommune og finansiert av DeKom (Desentralisert Kompetansehevning) midler fra Udir.

Vitensenteret i Trondheim
November 2020

Nils Kr. Rossing
Ola Kleiven
Rannvei Sæther
Anne Birgitte Belboe
Eva H. Hagen

Innhold

1 Innledning

I tillegg til et par Startere knyttet til programmering uten PC så handler denne dagen om å lage og programmere produktet: *Smartarmbånd*

1.1 Smartarmbånd

Dette er en aktivitet som i sin helhet er hentet fra Skaperskolen:
https://skaperskolen.no/smartarmband/

På denne nettsiden finner dere en oversikt over lærings- og kompetansemål og hjelpemidler til å gjennomføre en kreativ prosess. Vi vil ikke gjenta det her, men på denne samlingen konsentrere oss om det praktiske arbeidet med armbåndet og det programmeringstekniske.

Tegning: Skaperskolen

Tiden før lunsj vil gå med til å designe og lage armbåndet og montere elektronikken. Tiden etter lunsj vil bli viet programmeringen av armbåndet.

I stedet for bare å laste opp et ferdig program ønsker vi å utvikle programmet fra bunnen av. Ikke for det, det kan være mye læring i å ta utgangspunkt i et ferdig program for så å endre på parametere. Dette er kanskje det mest realistiske for det store flertallet av elever. I noen tilfeller vil *det* være riktig framgangsmåte.

Her velger vi å gå grundigere til verks og håper derfor å øke forståelsen for programutvikling.

1.2 Organiseringen av arbeidet

Arbeidet denne dagen er delt inn i to hoveddeler:

- Design og framstilling av armbåndet, med vekt på den kreative prosessen
- Programmering av armbåndets funksjon, med vekt på hvordan en kravspesifikasjon påvirker arbeidet med prosjekter i tillegg til å gi økt forståelse for programmering og bruk av sensorer.

Tabellen under viser hvordan dagen er organisert:

Tid	Tema	Kommentar
08:30 – 09:00	Introduksjon og omtale av dagens program. Deltagerne refererer fra egen loggbok	Velkommen og praktisk - Presentasjon av erfaring (fra loggboka) - Spørsmål etter forrige gang - Dagens program - Status utstyrspakker - Status prosjekt i klassen - Behov for veiledning? - Visning av MLTS på skolen?
09:00 – 09:15	**Starter:** Ev. gjennomføring av tradisjonell «Rødt lys»	• Motivasjon ved at spillet Grønt lys spilles på tradisjonell måte
09:15 – 11:30	**Arbeidsøkt 1:** Framstilling av armbånd	• Koble opp micro:bit og laste inn program versjon 1, uten lysdioder • Gå gjennom spillereglene. • Deltakerne prøver ut på egen micro:bit • Ev. prøve å spille spillet • Intro til programmet, • gjennomgang av endelig program • Bygg opp programmet gjennom 7 oppdrag • «Time out» underveis med litt teknisk fordypning
12:00 – 12:30	Lunsj	
12:45 – 15:00	**Arbeidsøkt 2:** Om programmering av smart armbånd – «Grønt lys» Avbrudt av innslag av Teori	• Lage armbåndet • Teste oppkoblingen med testprogrammet ev. rette opp feil
15:00 – 15:30	Uttesting av smart armbånd	• Test armbåndet med program versjon 1 og ev. 2 Oppmuntre til hacking i klasserommet

15:30 – 16:00	Oppsummering og oppgaver til neste gang	Oppgaver til neste gang • Jobbe med programmet for armbåndet • Jobbe med prosjekt i klasserommet • Lesestykke Utdrag av Programmering i skolen (kap 14) • Svare på spørsmål i loggboka

NB! Man tar pauser etter behov underveis

2 Programmering uten PC

Det er mange måter å trene algoritmisk tenkning og metodikken i programmering uten å bruke PC. Her er et par eksempler.

2.1 Tullete vilkår

Denne aktiviteten kan brukes for å innføre begrepet betingelser (vilkår) i programmering. Juster de ulike hendelsene slik at det passer for klassen din. Alle lappene unntatt lappen med START og STOPP deles ut med betingelsen NED.

HVIS læreren sier ordet **START**: Snu lappene foran deg og følg instruksjonene
HVIS du ser ordet **STOPP** på tavlen: Sett deg på plassen din, legg armene i kors og se på læreren (smil!).

Del ut disse til flere elever. Disse betingelsene kan utløses flere ganger. Gå rundt i klasserommet under aktiviteten for å utløse noen av betingelsene.

HVIS læreren sier ordet popkorn: Reis deg og si "Popp!" en gang og sitt ned.
HVIS noen skriver på tavlen med grønn tusj: Reis deg, rør noe grønt i rommet og sett deg tilbake på plassen din.
HVIS noen går forbi plassen din når du sitter der: Knips tre ganger.

HVIS noen skriver noe på tavlen: Reis deg, snurr en runde rundt og sett deg ned igjen.
HVIS noen reiser seg: Klapp tre ganger.

Del ut en av hver av disse:
Det er betingelser som bare utløses en gang og starter avslutningen av aktiviteten.

HVIS læreren plukker opp en bok: Reis deg, skriv bokstaven S på tavlen og sett deg tilbake på plassen din.
HVIS noen skriver bokstaven S på tavlen: Gå bort til døren, åpne og lukk den, sett deg tilbake på plassen din.
HVIS noen åpner og lukker døren: Reis deg, skriv bokstaven T (etter bokstaven S) på tavlen og sett deg tilbake på plassen din.
HVIS noen skriver bokstaven T på tavlen: Reis deg, finn lysbryteren og skru lyset av og på. Sett deg tilbake på plassen din.

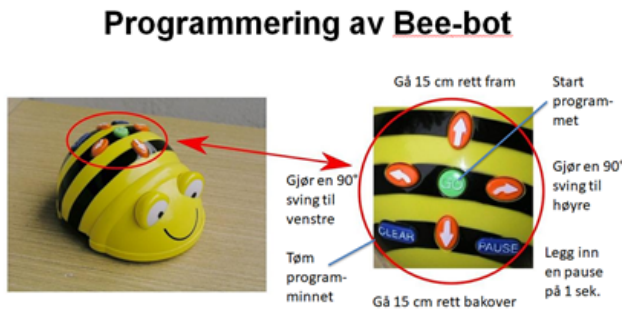
HVIS noen skriver bokstaven O på tavlen: Reis deg, skriv bokstaven P to ganger (etter bokstaven O) på tavlen og sett deg tilbake på plassen din.
HVIS noen skrur lyset av og på: Reis deg, skriv bokstaven O (etter bokstaven T) på tavlen og sett deg tilbake på plassen din.

2.2 Bee:bot

Bee:bot og Blue:bot er roboter som kan programmeres til å følge enkle instruksjoner og er en god første introduksjon til programmering. Her skal vi kort omtale Bee:bot og vise noen eksempler på hvordan man bruke robotene i en undervisningssituasjon.

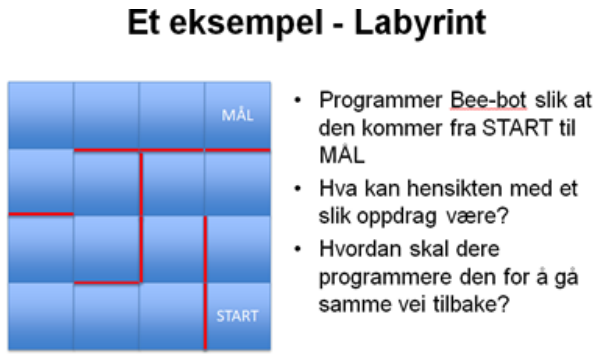
2.2.1 Introduksjon til Bee:bot

Figuren under viser kort hvordan man kan programmere Bee:bot til å bevege seg i rette vinkler i trinn på 15 cm. Sekvensen av instruksjoner (pilene) legges inn før man starter programmet (grønn knapp) og roboten utfører kommandoene. I tillegg kan gamle programmer slettes (CLEAR) i tillegg til at man også kan legge inn korte pauser (PAUSE) i programmet (blå knapper). Noen versjoner gir også mulighet til å legge inn lys som blinker. Ved hjelp av en knapp under kan man slå av og på lyd.



Figur 2.1 Programmering av Bee:bot

En enkel introduksjon kan være å la Bee:bot bevege seg langs en på forhånd utlagt labyrint som vist på figuren under. Siden det følger med en gjennomsiktig plastduk med 16 kvadratiske ruter med målene 15 x 15 cm, så er det lett å legge Post-it lapper under rutene med ulike tall, bokstaver orde eller figurer. Vi har også laget sperringer (laserkuttede 15 cm lange pinner) som kan legges opp som sperringer som vist på figuren under.



Figur 2.2 Labyrint - Eksempel på bruk som introduksjon til Bee-bot

Det følger også med et sett av programmeringskort som elevene kan bruke for å planlegge programmeringssekvensen. Kortene stimulerer også til samtale mellom elevene.

Figur 2.3 Programmeringskortene er fine som en hjelp til å diskutere strategier

Figuren viser en oppgave gitt til en gruppe førskolelærere for å stimulere dem til å komme med eksempler på bruk i barnehage.

Figur 2.4 Oppgavetekst førskolelærere

Du finner flere eksempler både for barn og voksne i vedlegg

3 Programmering av armbåndet

Det hele handler om den gamle leken "Rødt og grønt lys". Tradisjonelt har den for eksempel vært praktisert slik:

"Rødt og grønt lys"

> *Alle stiller opp på linje i enden av gymsalen/feltet. På motsatt side står en person som er "rødt lys". Det "røde lyset" snur ryggen til forsamlingen, og de beveger seg fremover mot vedkommende. Når personen som er "rødt lys", ønsker det, teller han høyt 1-2-3, sier "Rødt lys" og snur seg. Da må ingen være i bevegelse. De som er i bevegelse må returnere til start-linja. Den som kommer først frem til personen som er "rødt lys", overtar denne rollen i neste runde.*

Som "*starter*" for denne teknologiøkta kan man gjerne lek den tradisjonelle versjonen av leken.

Oppdraget går nå ut på å lage en teknologisk variant av denne leken hvor man utnytter micro:biten både for å varsle om overgangen fra grønt til rødt lys, men også til å detektere om deltagerne rører seg i den "røde"-perioden.

3.1 Kravspesifikasjoner

I dette avsnittet skal vi se nærmere på hvordan vi kan arbeide systematisk med å framstille et produkt. Vi kan tenke oss at klassen har fått eller gitt seg selv et oppdrag om å utvikle, framstille og levere et produkt.

Funksjonspesifikasjon

Det første en må gjøre er å finne ut hva oppdraget går ut på. I en hver sammenheng hvor man skal lage et produkt vil det være viktig å bestemme seg for hvilken funksjon produktet skal ha, vi kan kalle det for en *Funksjons- eller kravsspesifikasjon*. Dvs. hvordan brukeren av produktet skal oppfatte at det fungerer, ikke teknisk, men rent bruksmessig. For de fleste brukere er det likegyldig hvilke tekniske løsninger ingeniørene velger bare det har den ønskede funksjonen og ellers tilfredsstiller visse krav til f.eks. pris, holdbarhet, vekt, størrelse, brukerkomfort, dokumentasjon og leveringstid.

Teknisk spesifikasjon

Dernest er det ingeniørenes oppgave å finne ut hvilke *tekniske løsninger* som må velges for å oppfylle funksjonspesifikasjonen. De må bestemme utformingen slik at produktet er behagelig å bruke, de må velge komponenter og framstillingsmetoder slik at produktet oppfyller kravene til pris og holdbarhet, dernest bør det ikke være for vanskelig å reparere. Et produkt som kan skues fra hverandre er lettere å reparere enn et produkt som er limt eller sveiset sammen. Alle disse kravene til de tekniske løsningene kan vi kalle en *teknisk spesifikasjonen*.

3.1.1 Armbåndets funksjonsspesifikasjon

Her skal vi lage et Smart-armbånd som kan fungere som et viktig teknisk hjelpemiddel i leken "Rødt og grønt lys".

Følgende krav er i utgangspunktet gitt til utformingen:

- *Armbåndet må kunne tas av og på.*
- *Armbåndet må være behagelig å ha på.*
- *Armbåndet må sitte stabilt på håndleddet.*
- *Micro:bit og batteri må kunne tas av og på.*
- *Man må kunne se både skjerm og eksterne LED-lamper.*
- *Designet på armbåndet må visuelt kommunisere hvilket lag man tilhører.*

Dernest må vi ha en god forståelse for hvordan det skal fungere som et hjelpemiddel under leken "Rødt og grønt".

Funksjonspesifikasjonen til leken: Rødt og grønt lys

1. Elevene står på linje et stykke fra og like langt fra læreren som leder leken
2. Alle har Smartarmbåndet på armen og påslått.
   Armbåndet til deltakerne skal vise at de må holde seg i ro.
3. Læreren starter leken ved å gi deltakerne trådløs beskjed om at de kan bevege seg
4. Alle går mot læreren i normalt tempo (hva er "normalt tempo" for en elev i denne sammenhengen?)
5. Læreren gir så deltakerne trådløs beskjed om at de må stoppe
6. Armbåndet skal registrere om de beveger seg eller står helt i ro etter at stoppsignalet er gitt.
7. Dersom de beveger seg i "rød periode", så skal armbåndet varsle deltakerne om at de er diskvalifisert og må starte på nytt fra startlinjen
8. Dersom de klarer å holde seg i ro, viser displayet et hjerte og de kan fortsett fra der de stoppet ved neste grønne lys
9. Første elev fram til læreren har vunnet og overtar som leder i neste runde

Det vi legger merke til er at oppdraget er litt mangelfullt spesifisert på hjemmesiden til Skaperskolen. Vi antar at dette kan ha to årsaker:

- *At det er meningen at den ferdige koden bare skal lastes ned på micro:biten og at man kan utforske koden i ettertid for ev. å forbedre den eller gjøre den mer egnet for å vinne konkurransen (hacking)*
- *At det er opp til deltagerne og utforme koden slik at mikro:biten fungerer best mulig til det den er ment for*

Studerer vi koden og den tilhørende forklaringen, finner vi følgende funksjoner:

- Når læreren trykker på knapp A og B samtidig så resettes spillet både for læreren og alle elevene
- Når læreren trykker knapp A alene vises en "G" på lærerens og elevenes display og de kan fritt bevege seg
- Når læreren trykker på knapp B alene vises en "R" på lærerens og elevenes display og de må holde seg i ro
- Hos de elevene som ikke beveger seg i rød periode, vises et *hjerte* på displayet og de kan fortsette spillet
- Hos de elevene som beveger seg i rød periode, vises et *dødningehode* på displayet og de er ute av spillet
- De som er ute kan ikke komme inn igjen før læreren starter spillet på nytt. I denne tilstanden får de ingen nye beskjeder fra læreren.

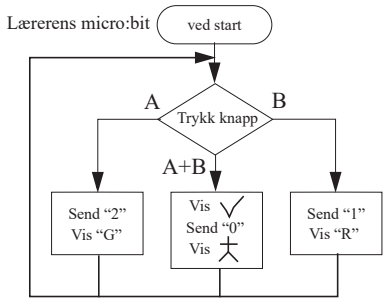
3.1.2 Teknisk spesifikasjon

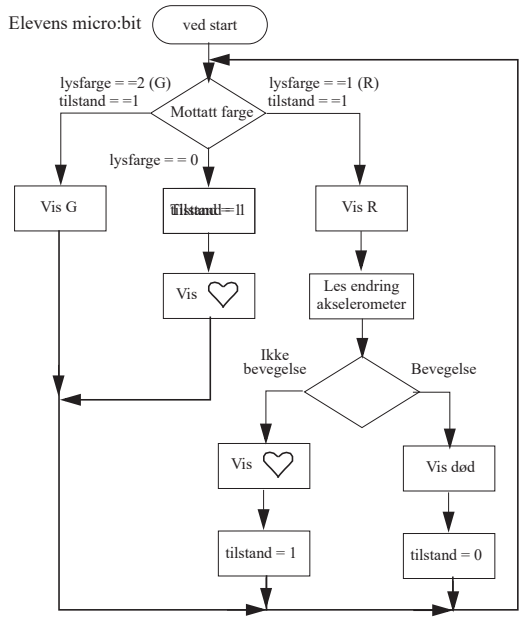
Flytdiagram

Det neste vi bør gjøre er å sette opp et *flytdiagram*. Flytdiagrammet er en mer skjematisk beskrivelse av oppdraget og er gjerne det første skrittet vi tar for å begynne å skrive programmet. Her er noen momenter som er viktige i et enkelt flytdiagram:

- Hvor *starter* programmet?
- Hvilke *handlinger* består programmet av?
- Hvilke *veivalg* finnes i programmet?

Å sette opp gode flytskjemaer kan være vanskelig og krever erfaring. Men et sted må man begynne.





3.1.3 Løsningsmetodikk

Dersom vi skal lage programmet fra bunnen av må vi finne ut hvilke programtekniske komponenter vi trenger for å løse oppdraget og oppfylle kravspesifikasjonen.

Det kan være lurt å sortere ut de som vi kjenner løsningen til og de vi ikke kjenner løsningen til.

Her er noen oppgaver som vi anser som ***lett å besvare*** siden vi kanskje har kunnskap om dem fra før:

1. Hvordan vise ikoner og tekst på displayet?
2. Hvordan lese av bryterne?

Her er noen oppgaver som vi anser som ***vanskeligere å finne svar på***. Det kan dessuten være lurt å prioritere oppgavene etter vanskelighetsgrad, eller i hvilken grad vi tror det kan by på problemer å finne en løsning. Ved å starte med det vanskeligste, så kan vi ev. ganske tidlig i arbeidet finne ut hva som ev. hindrer oss i å finne en endelig løsning og slik spare tid[1]:

1. Hvordan skal vi få overført informasjon fra læreren til alle elevene?

1. Ved å undersøke de mest utfordrende oppgavene først så kan vi ev. terminere prosjektet på et tidlig stadium der som noe viser seg umulig. Ev. en kan tidlig sette igang arbeid for å finne andre løsninger, på den måten kan man spare store penger.

2. Hvordan kan vi detektere bevegelse?

Siden vi alltid vil bevege oss litt så må vi bestemme:

3. Hva er *nok bevegelse* til at vi skal bli diskvalifisert?
4. Hvordan skal vi avgjøre at grensen for *nok bevegelse* er nådd?

Deltakere som har beveget seg ulovlig mye er disket og går ut av konkurransen:

5. Hvordan *skal* micro:biten til diskede elever *"huske"* at den er ute av konkurransen og ...
6. Hvordan kan micro:biten igjen få lov til å starte opp på nytt?

Så la oss begynne med de to mest utfordrende:

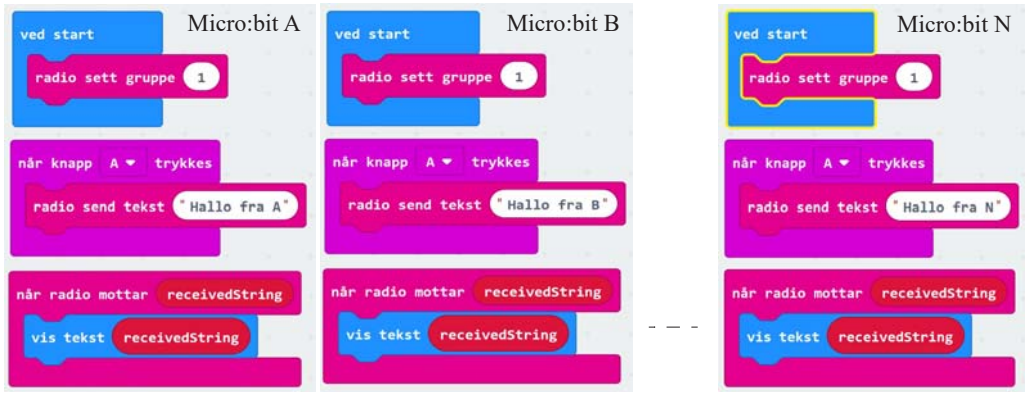
(1) Hvordan skal vi få overført informasjon fra læreren til alle elevene?

For oss som kjenner litt til micro:bit så er trådløs kommunikasjon, eller radio, den mest opplagte løsningen. Så er spørsmålet *hvordan tar vi i bruk radioen for å sende meldinger*?

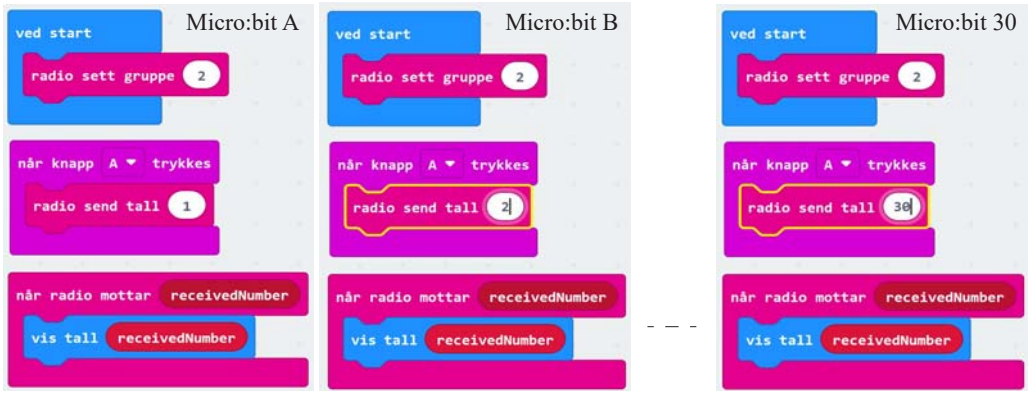
Ved å studere et utvalg av kommandoer som finnes under Radio-menyfanen, så ser vi at:

- ... vi først må velge en felles kanal eller radiogruppe (adresse) for de micro:bitene som skal kommunisere med hverandre
- ... vi kan sende og motta tekst
- ... vi kan sende og motta tall

Figuren under viser hvordan vi kan sette opp en *radiogruppe*, her gruppe 1, sende en setning når vi trykker på knapp A og motta og vise setningen på displayet når den mottas på de andre micro:bitene som har valgt samme kanal (nr. 1). Alle micro:bitene kan både sende og motta, og vil, når de trykker på knapp A, fortelle mottakerne hvem som har sendt meldingen (f.eks. "Hallo fra A"):



I figuren under sendes et tall når vi trykker på knapp A. Tallet kan f.eks. være knyttet til *hvem* som sender tallet (1, 2 eller til 30 eller andre tall). I dette eksempelet har vi valgt radiogruppe 2.

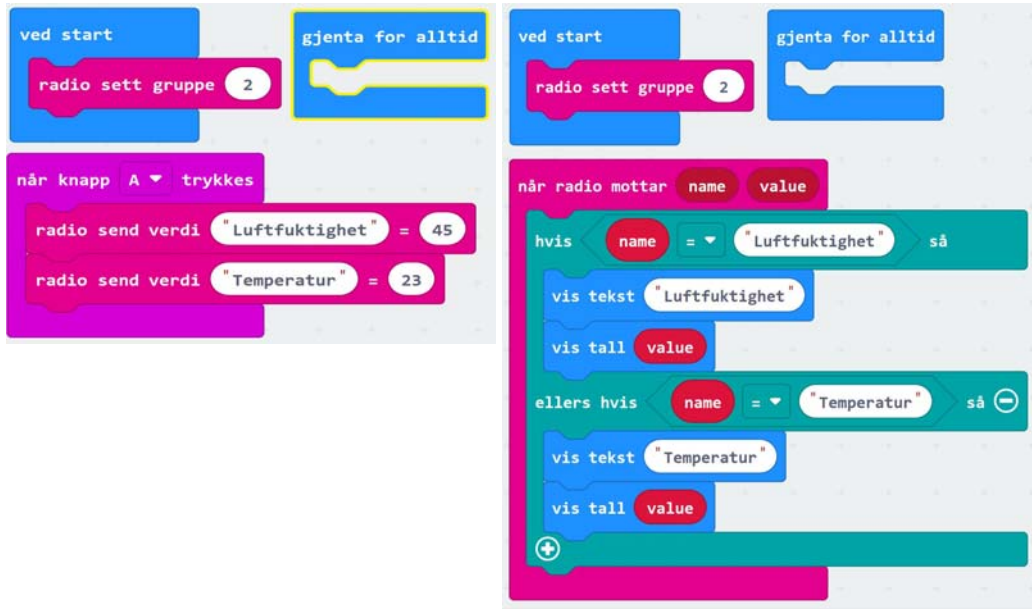


Legg merke til at tallet vi overfører er knyttet til et navn: *receivedNumber*. Dette er en *merkelapp* som vi setter på det tallet vi sender ut, så mottakeren kan vite hva vedkommende mottar, nemlig *receivedNumber*. Denne merkelappen kan vi så bruke når vi ønsker å bruke tallet i programmet vårt, f.eks. skrive tallet til displayet som vist på figuren over.

Dersom vi ønsker å sende over forskjellige tall, så gjør vi det ved å knytte ulike merkelapper til tallene som vist i figuren under. Her ønsker vi å overføre to tall-verdier, den ene er luftfuktighet og den andre er temperatur. Merkelappen forteller oss hvilken av de to vi mottar.

Micro:bit A
sender

Micro:bit B
mottaker



Dermed har vi funnet ut at det er mulig å overføre både tekst og tall. I tillegg kan vi knytte en

merkelapp til tallene slik at vi vet hva vi mottar dersom det kan være tvil om det. Dette er strengt tatt bare nødvendig når vi overfører flere ulike tall. Har vi bare et tall er det ikke nødvendig å sende med merkelapp.

Så er det bare å prøve kommandoene for å bekrefte at det fungerer.

Den som ønsker å vite mer om hvordan radioen fungerer kan gå til vedlegg B.

(2) Hvordan kan vi detektere bevegelse?

Det neste kritiske spørsmålet er om det er mulig å registrere at micro:biten rører på seg eller beveger seg.

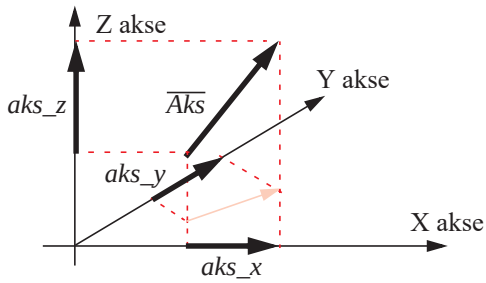
Her kan en tenke seg ulike mulige løsninger:

- ***Bruke lyssensoren***. Undersøke endringer i lysstyrken som treffer micro:bit når den beveger seg.
- ***Bruke magnetsensoren***. Undersøke endringer i målinger av magnetfeltet når micro:bit beveger seg
- **Bruke akselerometeret**. Undersøke endringer i akselerasjonen til micro:bit når den beveger seg.

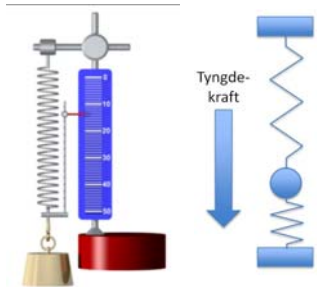
Alle disse tre sensorene kan la seg bruke, men det er nok ***akselerometeret*** som er det mest følsomme. Skal vi kunne ta i bruk dette må vi vite litt om hvordan det fungerer

Akselerometeret

Et akselerometer registrerer akselererende bevegelse, dvs. endring av hastigheten. Vi vet også at både hastighet og akselerasjon har en *retning i rommet*, som vi kan vise ved hjelp av en pil med en retning og en lengde. Her angir lengden av pila størrelsen på akselerasjonen. En slik pil i rommet med retning og lengde, kalles en *vektor.* Her kaller vi vektoren for $\overline{Aks}$ (streken over viser at det er en vektor). Lengden av vektoren kan vi f.eks. kalle *Styrke*.

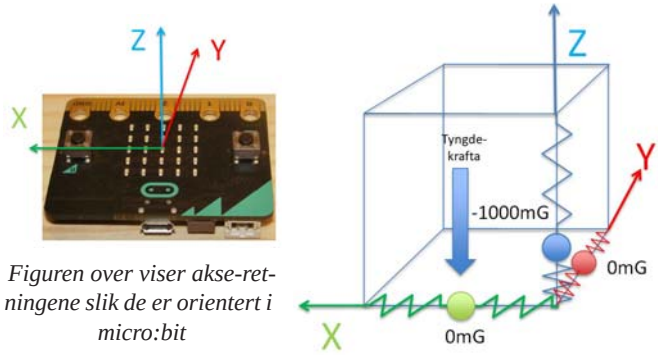


Lengden og retningen kan angis ved hjelp av tre størrelser *aks_x*, *aks_y* og *aks_z*, en størrelse langs hver av de tre aksene i aksekorset, med *X akse, Y akse og Z akse som vist på* figuren over til høyre.



En skulle tro at akselerasjonen er lik 0 langs alle akser når micro:biten holdes helt i ro. Det er den imidlertid ikke. Siden vi befinner oss i tyngdefeltet fra jorda, vil akselerometeret påvirkes av dette. Dette er forståelig dersom vi vet hvordan akselerometeret fungerer. Vi er også kjent med at om vi slipper et lodd så vil det falle mot bakken med økende hastighet, med *tyngdeakselerasjonen*.

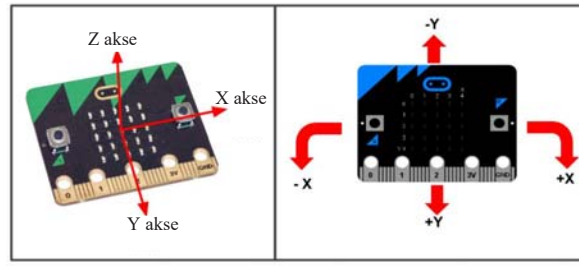
Om vi har et lodd som henger i en fjær som hos en fjærvekt, så vil tyngden av loddet strekke ut fjæra som vist på figuren over til høyre. Akselerometeret til micro:biten har tre slik "fjærvekter" som står loddrett på hverandre inne i den vesle kretsen. En i hver av de tre retningene x, y og z som vist på figuren til høyre.



Figuren over viser akse-retningene slik de er orientert i micro:bit

"Fjærvekter" som ligger horisontalt vil ikke påvirkes av jordas tyngdefelt og vil vise nær 0 mg (milli g), det vil derimot "fjærvekta som står vertkalt (– 1000 mG). En fjærvekt som gradvis går fra horisontal til vertikal stilling, vil måle en tyngdeakselerasjon som gradvis øker i verdi. Dersom vi henter inn informasjon fra alle "fjærvektene" så kan vi bestemme retningen i rommet til tyngdeakselerasjonen. Siden tyngdeakselerasjonen alltid virker vertikalt (nedover), så kan vi ved å lese av de tre verdiene, istedet bestemme orienteringen til micro:biten.

Figuren til høyre viser akselerometerets akseretninger i forhold til micro:bit-kortet. Siden akselerometeret forholder seg til tyngdeakselerasjonen, som er 1 G i vertikal retning, vil avleste verdier i x-, y- og i z-retning bli som i tabellen under avhengig av hvordan vi holder micro:bit'en. Legg merke til at den avleste verdien har benevningen mg (milli G), hvor 1000 milli G er 1 G:



Handling	x-retning	y-retning	z-retning
Flatt på bordet, display opp	0 mg	0 mg	- 1000 mg
Flatt på bordet, display ned	0 mg	0 mg	+ 1000 mg
Tilting mot høyre om y-akse, display opp	økende positiv verdi	0 mg	minkende negativ verdi
Tilting mot venstre om y-akse, display opp	økende negativ verdi	0 mg	minkende negativ verdi
Tilting framover om x-aksen, display opp	0 mg	økende positiv verdi	minkende negativ verdi
Tilting bakover om x-aksen, display opp	0 mg	økende negativ verdi	minkende negativ verdi

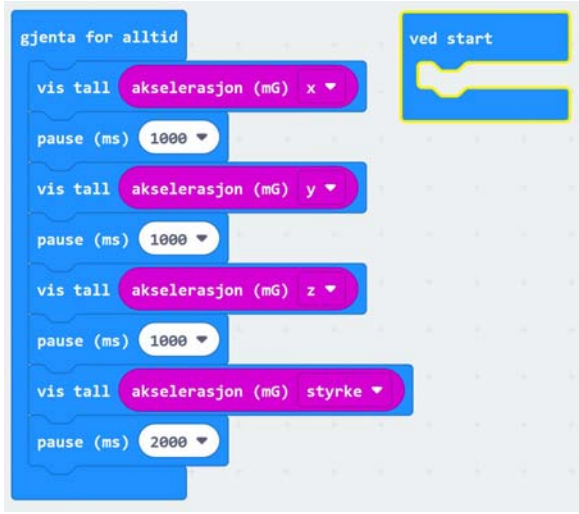
Det er viktig å merke seg at vi kan få alle mellomliggende verdier i x- og y-retning når vi dreier mikro:bit'en fra horisontal til vertikal orientering. Det samme gjelder også i z-retningen.

Så det vi ønsker å måle er om deltakerne har klart å holde micro:biten i samme posisjon så lenge de befinner seg i "rød" periode.

Det interessante er at dersom vi klarer å holde posisjonen fast (vinkelen i forhold til vertikal retning) og beveger oss med *konstant hastighet uten å akselerere*, så skulle vi også kunne bevege oss i "rød" periode uten å bli avslørt. Skjønt det er ikke så lett å få til.

Avlesning av akselerometeret

Vi kan lese av verdien av akselerasjonen i *x*-, *y*- og *z*-retning, i tillegg til at vi kan lese av lengden av vektoren, her kalt *styrke*.



Blokken *akselerasjon (mG) x, y, z og styrke* finner vi i menyfanen *Inndata*. Programmet vist på figuren til høyre leser av akselerasjonen i de tre retningene og *styrke* og skriver resultatene ut til displayet på micro:biten.

Et viktig spørsmål er hvilken av de fire verdiene vi skal bruke for mest effektivt å bestemme om elevene beveger seg?

Dersom vi bruker *x*-, *y*- eller *z*-*verdien* så vil vi registrere om elevene klarer å holde micro:biten i samme orientering i for hold til tyngdefeltet, i tillegg til om de "akselererer", dvs. beveger seg i rykk og napp. Bruker vi størrelsen *styrke* så registrerer vi bare om de beveger seg i rykk og napp.

Vi velger å bruke *styrke* for å avsløre elevenes bevegelse.

(3) Hva er nok bevegelse til at vi blir diskvalifisert?

Akselerometeret er særdeles følsomt. Det skal ikke så mye til før de målte verdiene forandrer seg når man rører på seg. Hvor mye, vet man ikke før man har undersøkt.

Dersom vi velger å bruke *styrke* for å avsløre bevegelse, så vet vi at denne sannsynligvis vil være i nærheten av 1000 mG som er lik tyngdeakslerasjonen. Vi kan lage et enkelt program for å undersøke størrelsen på måleverdien *styrke*. Programmet skriver ut styrken på den målte akselerasjonen, uansett i hvilken retning vektoren peker.

Tester viser at verdien ligger fra 980 – 1020 mg dersom vi holder micro:biten i ro. Dersom vi beveger den i raske bevegelser så variere den mellom 750 – 1250 mg.

Nå er vi bare interessert i endring fra et tidspunkt til et annet. La oss si at vi ønsker å se på endringen i akselerometerverdi i løpet av et halvt sekund. Dvs. at vi må måle styrken av akselerasjonen med f.eks. 0,5 sek. mellomrom. Vi lager oss to variabler og kaller dem henholdsvis *akselerasjonStart* og *akselerasjonSlutt*, som gjør måling før og etter tidsrommet på 0,5 sek. Forskjellen mellom de to finner vi ved å trekke den ene fra den andre, f.eks.

endring = akselerasjonSlutt – akselerasjonStart

Tester viser at variasjonen i differansen er fra –15 til +15 mg når den ligger helt i ro, altså en spredning på ca. 30 mg. Håndholdt er spredningen større. Ikke overraskende så er variasjonen både positiv og negativ.

For ikke å gjøre oppgaven for elevene for vanskelig, så setter vi kravet for å være i ro til mindre enn ±50 mg.

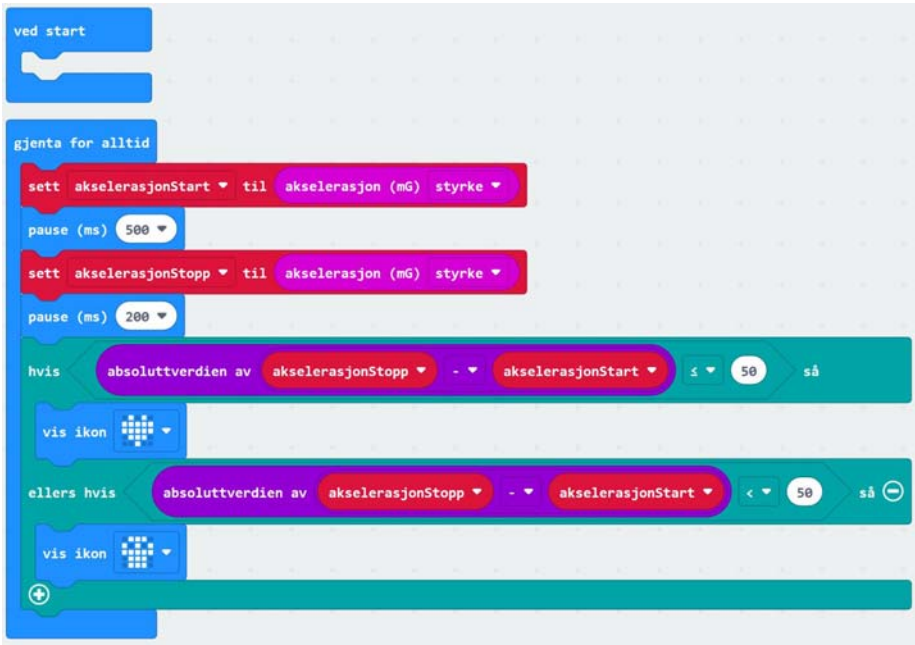
Tips: Det er mulig å måle styrken på det utsendte signalet, dvs. det er mulig å få en indikasjon på hvor langt det er mellom sender og mottaker. Ved å måle styrken til signalet fra senderen til læreren, så kan en la dette redusere terskelverdien slik at når signalstyrken blir større blir terskelverdien redusert og spillet blir mer krevende når man nærmer seg læreren.

(4) Hvordan skal vi avgjør at grensen for bevegelse er overskredet?

Vi skal nå teste om bevegelsen er akseptabel – dvs. at vi får lov til å fortsette spillet, eller uakseptabel – at vi må gå ut av spillet. For å få til dette bruker vi en *Hvis-ellers blokk*.

For å unngå å måtte teste for både positive og negative verdier, så er det enklere å først ta *absoluttverdien* av differansen. Dvs. at man sløyfer fortegnet og ser kun på verdien, til dette finnes det en egen kommando i meny-fanen *Matematikk*.

Vi kan nå teste om absoluttverdien av endringen i akselerasjonen er større eller mindre enn grenseverdien på f.eks. 50.



Programmet viser et hjerte om endringen i akselerasjon er mindre enn 50 og et dødningehode dersom den er større enn 50.

(5) Hvordan skal micro:biten huske at den er ute av konkurransen?

En måte å gjøre dette på er å bruke et "*flagg*" (huskelapp). Dersom flagget er "heist", så har noe skjedd, dersom det ikke er "heist" så har det ikke skjedd. Begrepet flagg stammer kanskje fra amerikanske postkasser som gjerne har et lite flagg som kan reises opp når det er kommet post.

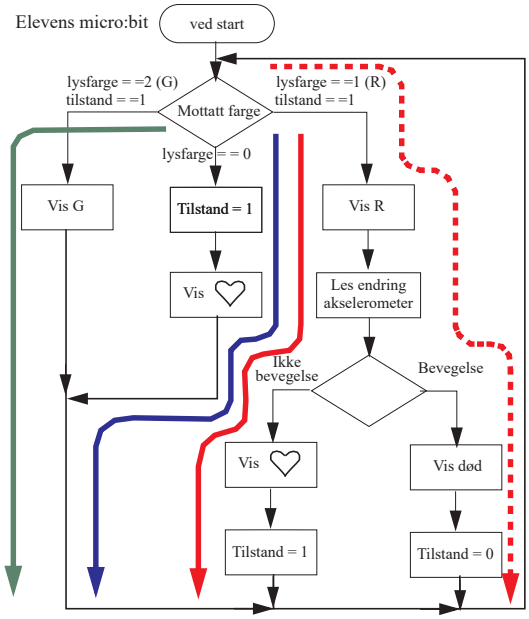
Hendingen i vårt prosjekt er at man har rørt på seg i utide og må gå ut av spillet. Flagget vårt er en variabel som vi har kalt *tilstand*. Om tilstanden er satt lik "1" så er vi inne i spillet, når tilstanden er "0" så er vi ut av spillet. Om man synes det er mer logisk, kan man definere det omvendt.

(6) Hvordan kan vi igjen få lov til å starte opp på nytt?

Det er viktige at man husker å "fire" flaggene når virkningen av hendingen er opphevet. I vårt tilfelle skjer det når læreren starter spillet på nytt, se blå linje på flytskjema til høyre.

Dersom vi ser på flytdiagrammet vårt, ser vi at så lenge *tilstand* er lik "1" så er vi inne i spillet og mottar alle meldingene fra læreren (rød (variabelen "lysfarge = 1") og grønn heltrukket linje).

Så snart vi er ute av spillet så slutter vi å få meldinger fra læreren (stiplet rød linje). Når så læreren sender *lysfarge* "0", så får igjen alle lov til å delta. Dette gjøres ved å sette *tilstand* lik "1" (blå linje), enten *tilstand* var "0" eller "1" på forhånd. Dermed er alle med i spillet igjen.

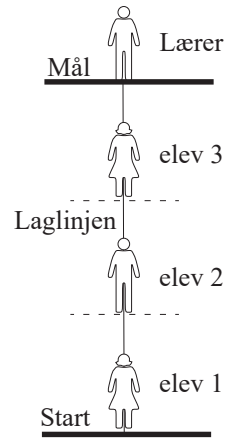


3.1.4 Andre måter å spille

Det finnes flere måter å spille dette spillet på, en annen er ...

Stafett

- Elev 1 på hvert lag starter på linje et stykke unna lærer. Både elev 2 og 3 plasserer seg på laglinjen med like stor avstand mellom hverandre.
- Læreren styrer spillet ved å trykke på knapp A på micro:bit for grønt lys og på B for rødt lys.
- Elev 1 starter å gå i normalt tempo mot elev 2 når «G» lyser på skjermen, mens elev 2 og elev 3 må stå helt i ro.
- Elevene stopper og står stille når «R» lyser på skjermen.
- Beveger eleven seg med «R» på skjermen vil det vises et dødninghode på skjermen og eleven må rykke tilbake til start.
- Når elev 1 når fram til elev 2, blir elev 1 stående helt i ro, mens elev 2 begynner å gå mot elev 3.
- Det laget hvor elev 3 først når fram til læreren har vunnet.



3.2 Definisjon og bruk av variabler

Hva variabler er og hvorfor vi skal bruke dem er for mange ganske uforståelig. Vi vil derfor her forklare hva variabler er og hvorfor vi bruker dem.

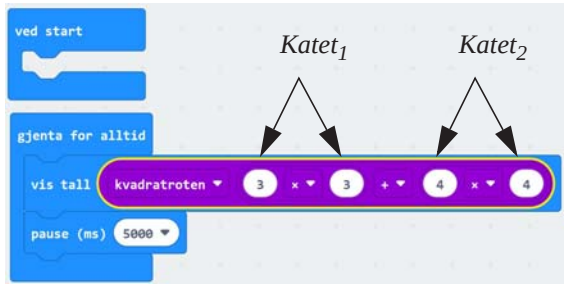
3.2.1 Et eksempel på bruk av variabler

Vi skal vise et eksempel for å argumentere for hvorfor vi bruker variabler.

Vi ønsker å lage et program som regner ut hypotenusen i en rettvinklet trekant når vi kjenner lengden til de to katetene. Vi antar at katet 1 er 3 meter og katet 2 er 4 meter. Vi bruker da Pytagoras setning som sier:

$$Hypotenusen = \sqrt{3^2 + 4^2} \tag{3.1}$$

Vi lager oss følgende program som gjør denne utregningen:



I dette programmet setter vi verdien for katetene rett inn i ligningen og skrive resultatet direkte ut til displayet. Vi registrerer følgende:

- Vi må skrive lengden av hver av katetene to ganger
  Dette er kanskje ikke så galt her. Dersom vi har mange utregninger i programmet vårt hvor lengden av katetene brukes flere ganger, så blir det straks mer komplisert.
- Det er problematisk å huske hva 3 og 4 står for, de har ikke noe navn
  Dette vil være et alvorlig problem da vi lett kan ta feil og sette inn verdiene for de to katetene på feil sted i en komplisert utregning.
- Vi kan ikke ta vare på resultatet av utregningen, dvs. lengden til hypotenusen
  Det er kanskje greit nok så lenge vi bare har ett resultatet å holde rede på, nemlig hypotenusen. Det blir imidlertid nesten umulig om vi ønsker å bruke resultatet av utregningen i videre utregninger.

Så, ...

Hva er en variabel og hva oppnår vi med å bruke variabler?

Vi velger å sette et ***navn*** på de to katetene, og hva er vel mer naturlig enn å kalle dem for *Katet_1* og *Katet_2*, eller om vi vil *K_1* og *K_2*. Legg merke til at vi bruke understrekning (underline) for å unngå oppdelte ord.

I programmering er ikke *Katet_1* og *Katet_2* bare navn, de er *oppbevaringssteder* for tallene som uttrykker lengdene av katetene, i dette tilfellet 3 og 4 meter. Vi kan tenke på det som skuffer i en kommode, skuffer det er satt navn på. Disse oppbevaringsstedene kaller vi *variabler* siden innholdet (verdien) kan variere, mens navnet er uforandret. Det er viktig å velge fornuftige navn på variablene slik at vi som programmerere, husker hva som er hva. Dermed er *Katet_1* og *Katet_2*

bedre navn enn *K_1* og *K_2*. *K_1* og *K_2* er til gjengjeld mye kortere, så her må vi bruke skjønn.

Det er ofte også viktig å ta vare på resultatet, her hypotenusen. Dermed lager vi en variabel med navnet *Hypotenusen*, ev. bare *H*.

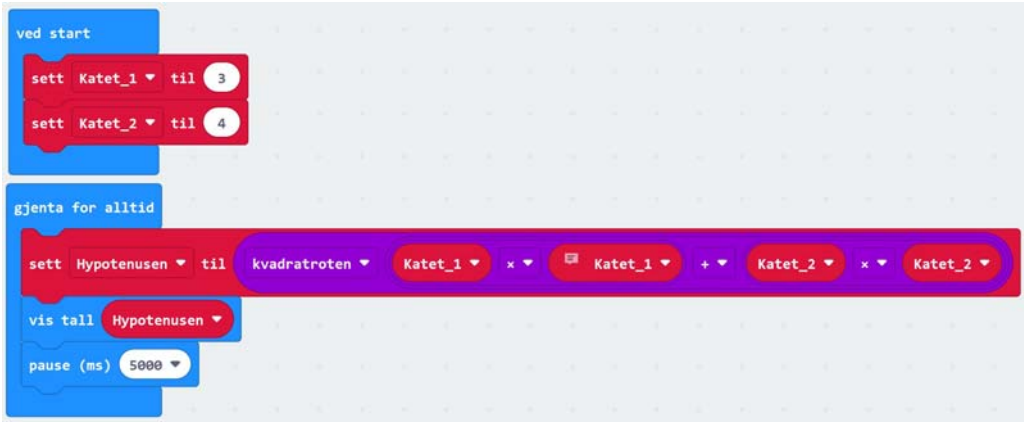
Dermed oppnår vi følgende:

- Vi kan nøye oss med å skrive inn verdien til variablene bare en gang selv om vi bruker dem mange ganger i utregningen.
- Det blir mye lettere å huske hva som er hva, vi ser av navnet hva som er katet 1 og 2
- Vi kan ta vare på resultatet, hypotenusen, og bruke den senere i regnestykket

Og det aller viktigste ...

- Vi kan sette opp hele regnestykket uten at vi kjenner de eksakte verdiene. Disse kan vi sette inn senere

Programmet blir da som følger:



Som vi ser blir kommandoene lengre side de skal inneholde variabelnavnene, men de blir lettere å lese og forstå.

Hvordan lager vi variabler?

Menyen har en egen fane som heter ***Variabler***. Velger vi den får vi opp en oversikt over de variablene vi har laget og spørsmål om vi vil lage nye variabler som vist i figuren til høyre.

Vi skriver da inn navnet på variabelen vi ønsker å opprette og trykker OK.

Vi gjør tilsvarende for *Katet_2* og *Hypotenusen*.

Vi legger også merke til at vi har fått opp to nye kommandoer, nemlig:

sett Katet_2 til
Her gir vi variabelen en ønsket verdi

endre Katet_2 med
Her kan vi *legge til en verdi* til den verdien variabelen alt har

Dette gjelder selvfølgelig ikke bare *Katet_2*, men alle de definerte variablene i lista. Vi endrer variabel ved hjelp av den vesle pila til høyre hos kommandoene.

Variabler blir mye brukt så det er like greit å lære det først som sist.

4 Programmering av armbåndet – Oppdragene

I denne delen skal vi gjennomfør mange mindre oppgaver som til sammen vil bli sender og mottaker programmet som installeres henholdsvis hos læreren og hos elevene. Vi nøyer oss med ganske kort å beskrive oppgaven slik at en ser strukturen i oppbygningen av forståelsen. For nærmere beskrivelse se oppdragskortene.

4.1 Oppdrag 1 – Melding via radio

Deltagerne skal i dette oppdraget sette opp radioen og sende og motta meldinger fra hverandre. Meldingene skal skrives på displayet som rulletekst. Sentralt i dette oppdraget er å:

- Lære å sette opp en radiogruppe
- Lære å sende og motta tekststrenger
- Lære å bruke blokken "når radio mottar receivedString"
- Lære å bruke variabelen "receivedString"

4.2 Oppdrag 2 – Lage program til lederen av leken "Rødt lys"

I dette oppdraget skal deltakerne tenke gjennom gangen i leken. Dessuten skal de lage programmet til lederen av leken hvor de skal sende over tallene 1 ved "Stopp" og 2 ved "Gå". Dessuten skal programmet gi lederen beskjed om hva som sendes ut: Ved "Gå" skal det vises en "G" (Grønt) på displayet, og ved "Stopp" skal det vises en "R" (Rødt) på displayet.

Det sentrale i dette oppdraget er å:

- Lære å tenke strukturert gjennom oppdraget – Gangen i leken
- Trene på å sende tall og vise symboler på displayet

4.3 Oppdrag 3 – Program til deltaker

I dette oppdraget skal deltakerne lage den første utgaven av programmet for å motta instrukser fra lederen av leken. Mottar de et 1-tall skal displayet vise "R" (Rødt) og de må stoppe, mottar de et 2-tall skal displayet vise "G" (Grønt" og de kan gå framover.

Det viktig i dette oppdraget er å:

- Trene på å bruke variabelen receivedNumber
- Trene på å bruke hvis-blokker
- Lære å sette opp betingelser
- Trene på å bruke variabelen "recievedNumber" i en betingelse

4.4 Oppdrag 4 – Bevegelse

I dette oppdraget skal deltakerne bli kjent med akselerometeret. De skal lage et frittstående program som viser verdien av avlest akselerasjon i *x*-, *y*- og *z-retning* i tillegg til *styrke* som er vektorsummen av de tre målingene langs hver akse. Det viktig er å:

- Erfare hvordan akselerometeret fungerer og hva verdien betyr
- Lære å bruke variabelen akselerasjon (mG)
- Lære å forstå hvordan et akselerometer fungerer

Det siste er viktig for å kunne tolke målingene

4.5 Oppdrag 5 – Vis endring i akselerasjon

I dette oppdraget skal deltakerne lage et program som tester ut endringer i den målte verdien av akselerasjonens "styrke" over tid. For testen brukes et tidsintervall på 2 sekunder. Vi er bare interessert i størrelsen på endringen ikke om den er positiv eller negativ, vi bruker derfor absoluttverdien av "styrke".

Deltakerne skal:

- Trene på å bruke variabler
- Lære å finne absoluttverdien av endring over tid
- Få en dypere forståelse av hvordan akselerometeret fungerer

4.6 Oppdrag 6 – Bevegelse hos deltaker

I dette oppdraget bygges hele programmet for deltakerne opp. De må ta imot tall fra lederen av leken ("1" – Stopp, "2" – Gå, "0" – Nullstill leken og klargjør for omstart) og vise riktig respons på displayet. Dersom de får beskjed om å holde seg i ro, vil målte endringer i akselerasjonens styrke bestemme om de må gå ut av spillet eller kan fortsette, i fall de må gå ut vises et dødningehode og variabelen "tilstand" settes til 0. Dersom lederen sender en "0" skal spillet resettes og de kan starte på nytt igjen.

Det mest krevende i dette oppdraget er å klare å kombinere alle betingelsen på en korrekt måte. De skal:

- Motta tallet lysFarge
- Ta konsekvensen av verdien av tallet
- Vise riktig symbol på displayet
- Sjekke akselerometeret for bevegelse
- Ta konsekvensen av måling av akselerasjon
- Ta vare på resultatet av målingen i variabelen "tilstand"
- Ta konsekvensen av "tilstanden" og

- Resette “tilstanden” når lederen ønsker dem velkommen inn i spillet igjen.

En slik kombinasjon av mange betingelser kan være en krevende øvelse. En variant kan derfor være at de studerer den ferdige koden og ser om de kan forstå den ut fra oppdragene 1 – 6.

4.7 Oppdrag 7 – Lysdiode på armbåndet

I dette oppdraget skal deltakerne inkludere tenning av den røde og den grønne lysdioden slik at det skal være lettere å se når de skal stå stille og når de kan bevege seg. Rød lysdiode lyser når de skal stå stille (lysfarge = “1”), og grønn lysdiode skal lyse når de kan gå (lysfarge = “2”).

Det som kan være krevende her er:

- Å vite hvor i programmet hvor tenning og slukking av LED må gjøres
- Å sørge for at diodene slukkes til rett tid.

5 Framstilling av et smartarmbånd

I dette kapittelet skal vi ganske kort se på et par alternative måter å lage et armbånd for å holde mikro:bit med tilhørende lysdioder og batteri.

Hele hensikten er at micro:biten skal kunne festes til armen slik at det skal være enklere å bruke den under leken "Rødt og grønt lys".

5.1 Materialer

Armbåndet kan lages av mange forskjellige materialer, men følgende ting hører normalt med:

Utstyr:

- En micro:bit
- En batteriholder med bryter[2]
- Batteri, 3V – CR2032[3]
- Ledninger i ulike farger[4]
- En grønn lysdiode[5]
- En rød lysdiode[6]
- Noe for å feste utstyret til armen





Verkstøy

- Saks, sideavbiter, spisstang, pinsett, skrutrekker, stor broderinål

Vi anbefaler Skaperskolens introduksjonvideo og organisering av prosjektet:

https://skaperskolen.no/undervisningsopplegg-8-10/smartarmband/

Framstillingen består av to deler:

1. Elektronikken og oppkoblingen – Denne er bestemt og her gis det i utgangspunktet ikke rom for så mye kreativitet
2. Armbåndet og monteringen – Her er det rike muligheter til å være kreativ og dra nytte av de kreative prosessene man har lært.

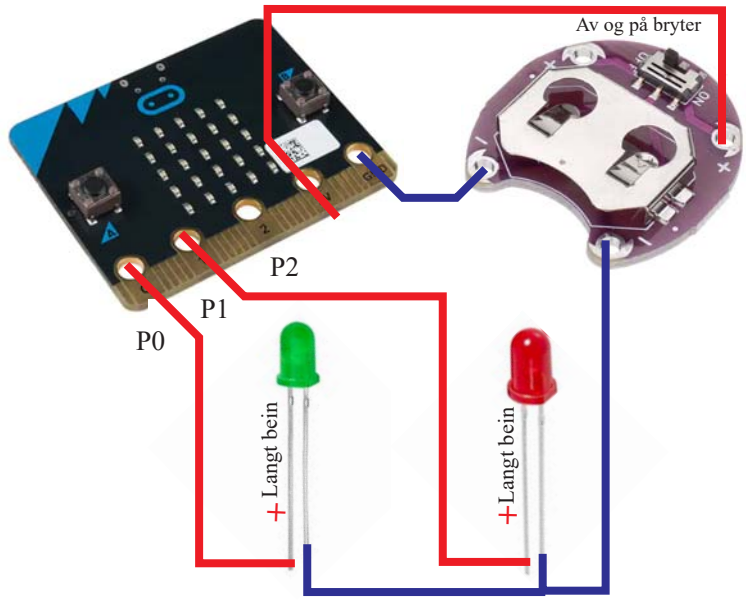
---

2. BangGood: https://www.banggood.com/no/LilyPad-Coin-Cell-Battery-Holder-CR2032-Battery-Mount-Module-p-1596232.html
3. Batteri CR2032 kan kjøpes nesten overalt, men IKEA er billig.
4. Egnede ledninger (0,14mm^2) i ulike farger kan bestilles fra ELFA, Grønn 300-10-16, Sort 300-10-155, Hvit 300-80-577, Rød 300-10-158, Blå 300-10-156 ELFA: https://www.elfadistrelec.no/
5. Grønn lysdiode ELFA: 175-10-192 ELFA: https://www.elfadistrelec.no/
6. Rød lysdiode ELFA: 175-06-664 ELFA: https://www.elfadistrelec.no/

La oss først se på koblingsskjemaet før vi ser på et par alternativer på utformingen av armbåndet. Det er viktig at en begrenser bruk av ferdige modeller da dette lett fører elevene inn i et spor. Det samme gjelder i og for seg oss som er lærere. Selv om vi har foreslått to ganske så forskjellige framgangsmåter så finnes det svært mange flere som sikkert også er bedre.

5.2 Koblingsskjema

La oss først se litt på hvordan vi skal koble opp kretsen, vi velger å ikke bruke lodding.Figuren under viser hvordan vi må koble opp de elektroniske delene.



Bruk av ledninger med forskjellige farger, gjør oversikten lettere, men er elektrisk uten betydning.

5.3 Montering av armbånd laget av tøy[7]

Her er det rike muligheter til å eksperimentere og bruke ulike materialer. På eksempelet under har vi valgt å bruke strielignende tøy av typen Aida. Dette er kraftig vevd stoff som kan egne seg til å tre ledninger gjennom. Ved hjelp av påmonterte trykknapper kan en feste stoffet til armen.

Utstyr:

- En stor broderinål
- Et stykke Aida-stoff (6 x 25 cm)
- Ev. trykknapper

Her er noen tips for å lette monteringen.

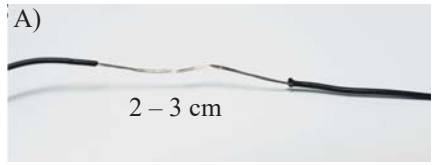
7. Av Rannvei Sæther

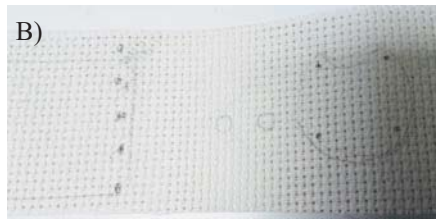
5.3.1 Tips til fremstilling av armbånd med Aida stoff

Utgangspunktet for denne beskrivelsen er en remse av Aida stoff. Stoffet må være langt nok slik at det går rundt håndleddet og vel så det, og bredt nok slik at micro:biten kan ligge på tvers.

Lag gjerne en skisse av tøystykket gjør følgende:

- Bestem hvor de ulike elektroniske delene skal være på tøystykket. Kanskje det er mulig å tegne svakt på stoffet.
- Tegn koblingsskjemaet med rett farge på lysdiodene og ledningene
- Bruk lange ledninger og avisoler den en enden i en lengde på ca. 2–3 cm for tilkobling (A). La en del av isolasjonen bli igjen på enden av ledningen for å lettere å kunne sy med ledningen i stoffet
- Merk av på stoffet med blyant hvor hullene til micro:biten, batteriholderen og lysdiodene vil komme (B).
- Bruk en broderinål til å gjøre hullene i Aida-stoffet litt større.
- Tre ledningen på nåla slik at den ligger i klem (C).





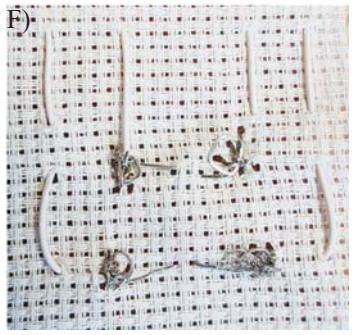
- Start med ledningen ved pin0 (P0) på micro:biten og sy deg mot den grønne lysdioden. La det være igjen rikelig med ledning både ved micro:biten (P0) og ved lysdioden.
- Avisoler enden ved lysdioden slik at det er rikelig ledning til å feste ledningen til diodens lengste ben (+)
- Merk gjerne det lengste beinet til dioden med tusj slik at det er lettere å se forskjell på de to beina.
- Bruk en spisstang, en krokodilleklemme eller lignende og lag ei løkke på beinet som vist på figur D og E til høyre

- Fest den avisolerte delen av ledningen til dioden ved å sy noen runder gjennom løkka, pass på at ledningen presser hardt mot beinet til lysdioden (F).
- Gjør likedan for å koble opp det andre beinet på dioden. Dette skal til minus (GND) på micro:biten.
- Gjenta prosessen for den røde dioden hvor det lange beinet skal kobles til pin1 (P1) på mikro:biten og det korte beinet til minus (GND) på micro:biten.

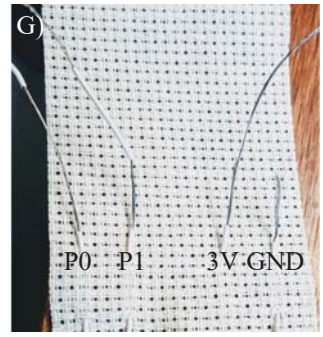
Montering av micro:bit

Når alle diodene er festet skal det være fire ledninger som stikkes gjennom stoffet og opp på forsiden av tøystykket. Avstanden skal være slik at de akkurat passer til hullene i micro:biten (P0, P1 3V og GND).

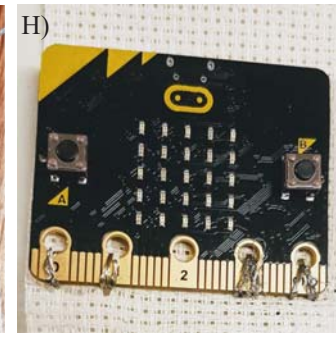
- Avisoler ledningen som omtalt over, gjerne i en lengde på 3 cm eller mer.
- Fest alle de løse ledningene til micro:biten ved å tre ledning i nåla og dra den igjennom stoffet og opp gjennom hullet i micro:biten. Tre nåla på nytt og dra igjennom stoffet. Fortsett til det er blitt godt festet i alle hull de fire hullene



Sett fra baksiden



Sett fra forsiden



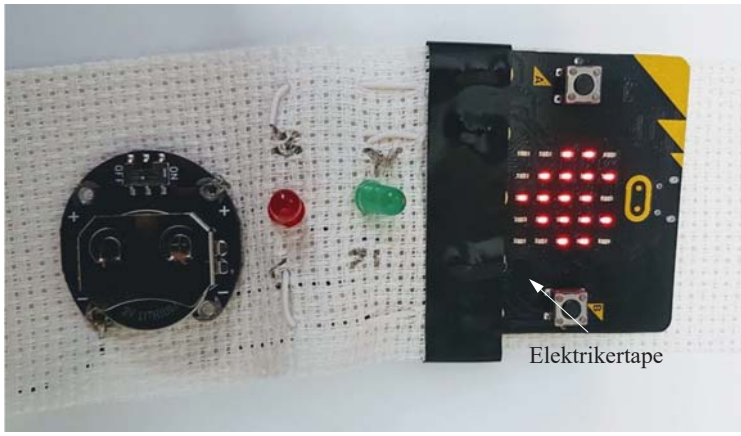
Sett fra forsiden

Montering av batteriholderen

- Fest batteriholderen til Aida-stoffet med avisolert ledning på samme måte som diodene.
- Husk å koble en ledning (svart) fra batteriholderens minus til pin GND på micro:biten og en ledning (rød) fra plussiden på batteriholderen til pin 3V på micro:biten. Legg merke til at det er to plasser på batteriholderen som er merket +. Ved å bruke den til høyre så vil betegnelsen på bryteren bli riktig ("on" betyr på)

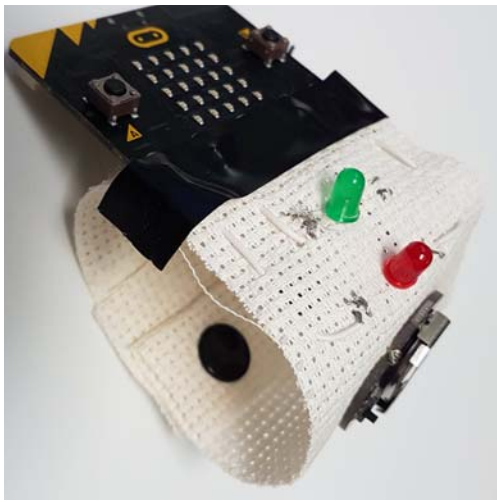
Monter til slutt en trykknapp som holder armbåndet på plass rundt håndleddet.

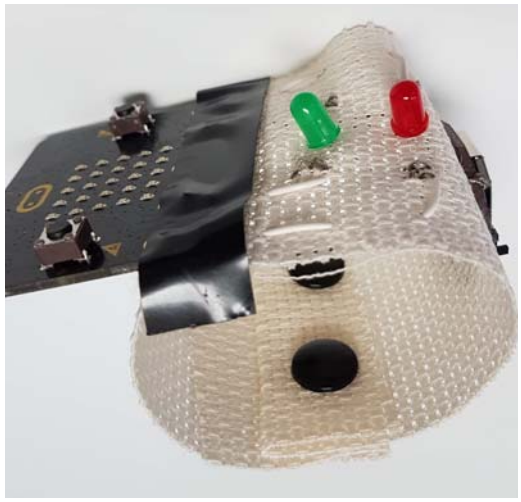
- Om ønskelig fest diodene til ledningen ved å forsegle med klar neglelakk. Neglelakk kan også virke isolerende så lodding er absolutt det beste om man har det.
- Ved behov, ta en bit tape over micro:biten for bedre feste

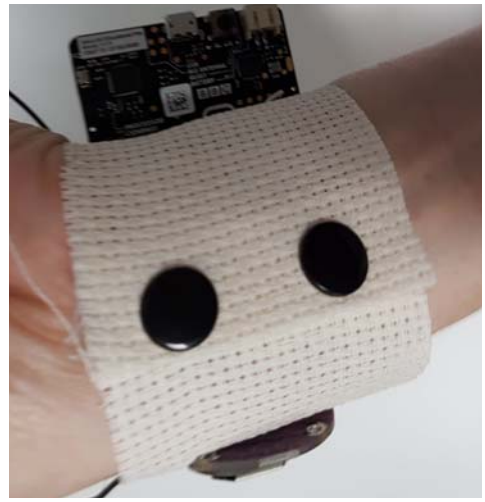


5.3.2 Flere bilder av det ferdige armbåndet

Her er flere bilder av eksempelet på et ferdig armbånd.





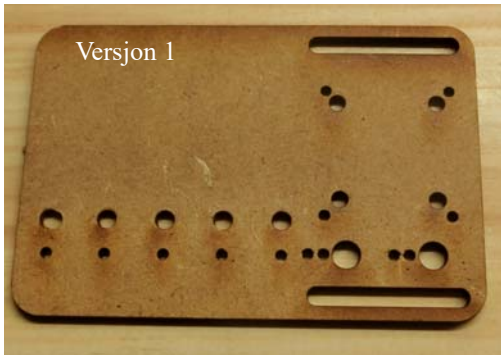


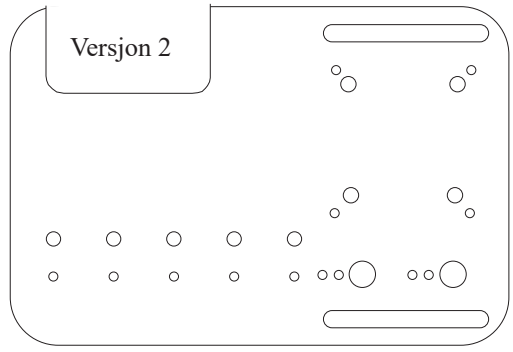
5.4 Montering av armbåndet på en MDF-plate

Utstyr:

- 1 stk. 2 mm MDF-plate
- 5 stk. skruer med muttere og skifer M 2,5 (Ø 2,5 mm)
- 1 stk. skireim
- Tilgang til laserkutter

Koblingskjemaet er det samme som omtalt i avsnitt 5.2. I denne varianten skal vi forsøke å montere hele kretsen på en stiv plate skåret ut i 2 mm MDF. Fordelen med en slik løsning er at hele konstruksjonen blir stødig.

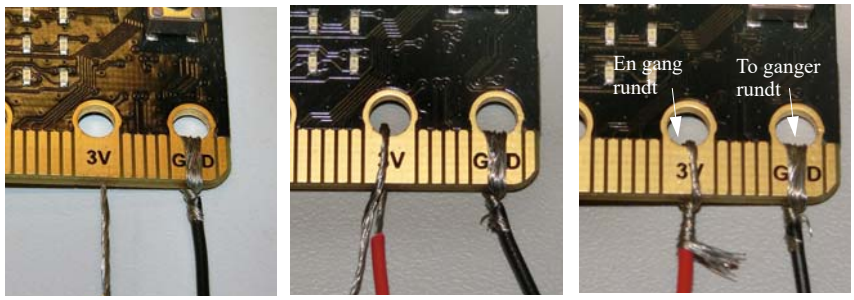


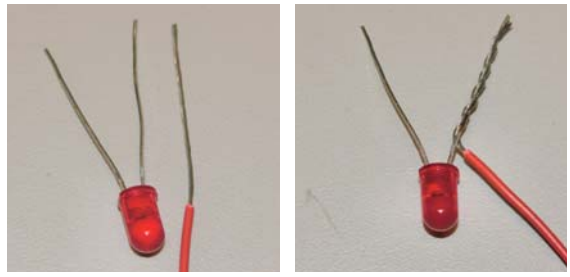


Vi ser at tegningen til venstre har fått en åpning i bakkant. Dette var et resultat av at det sitter en batterikontakt, en bryter og en USB-kontakten som trenger litt plass. Derfor ble det i versjon 2 gjort plass til disse.

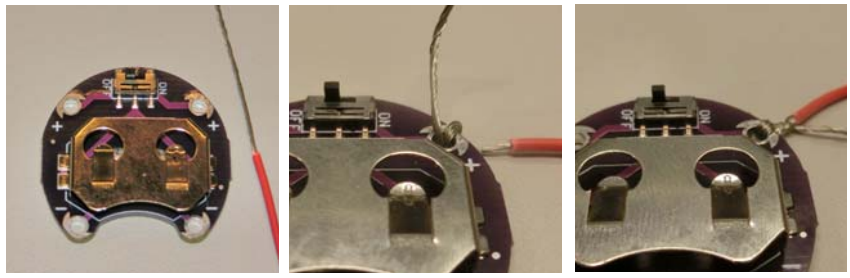
5.4.1 Oppkobling av ledninger

Siden vi ikke vil lodde ledningene til micro:biten og lysdiodene kan vi lett få dårlig kontakt. Ledningene vi bruker består av mange små tynne ledninger (kordeller), disse lar seg lett feste til kontaktpunktene til micro:biten eller vikles rundt beinet på lysdioden som vist på figuren under.





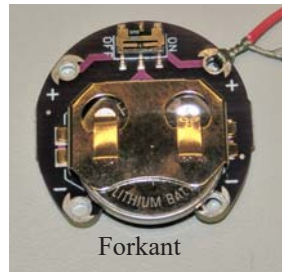
Dersom du legger ledningen *to gangen* gjennom kontaktpunktene (hullene) til micro:biten og batteriholderen, for så å stramme godt til, burde det gå bra en stund.



Dersom du har flere farger kan man gjerne bruke rød ledning til 3V, sort ledning til GND (–) og grønn og rød til de to lysdiodene. På den måten er det lettere å ha oversikt.

Batteriet

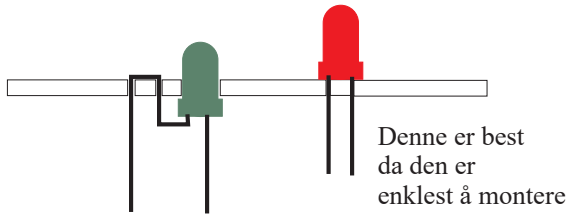
Det viser seg at batteriet ikke får god kontakt med tilkoblingen på undersiden med mindre det plasseres litt i forkant av holderen som vist på bildene under til høyre. Pass på at den siden som er merket med + kommer opp.



Ved å bruke + terminalen på batteriholderen som er til høyre virker bryteren i henhold til merkingen ("on", "off".

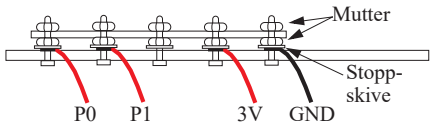
Montering av lysdiodene

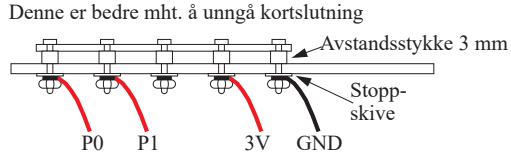
Lysdiodene er montert på en litt spesiell måte for ikke å falle ut. I ettertid viser det seg at det sannsynligvis ville vært bedre og vesentlig enklere å montere dem som vist til høyre på figuren under.



Montering av micro:bit

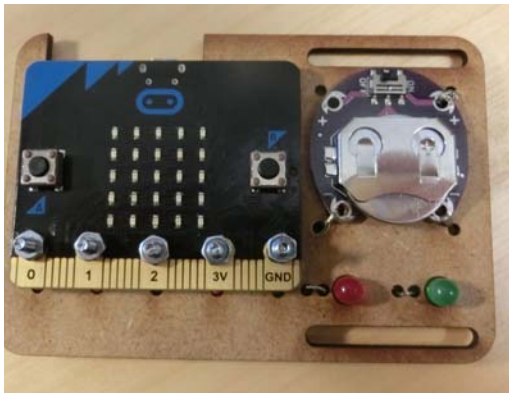
Siden ett av kravene er at micro:biten skal være lett å demontere, helst uten å ødelegge oppkoblingen på armbåndet, så har vi valgt å bruke skruer M2,5 (Ø 2,5mm). Vi har valgt dimensjonen så liten for at mutrene ikke skal berøre naboportene. Dette kan uansett være en utfordring, noe som kan løses ved å la hodet på skruen være på oversiden og ved bruk av plast avstandsstykker (f.eks. 3 mm) mellom plata og micro:biten, som vist til høyre på figuren under.

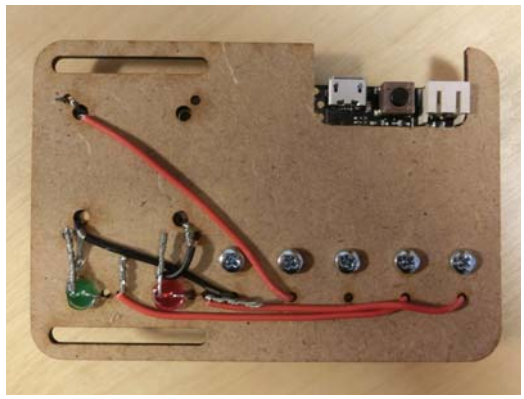




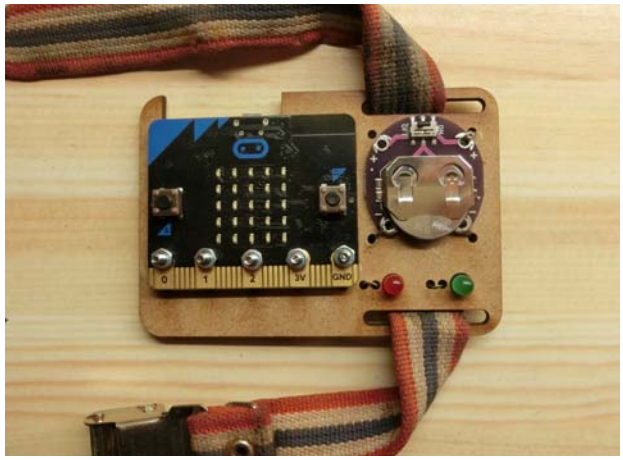
5.4.2 Montering.

Siden det er gjort plass på plata til alle komponentene er det bare og sette alle komponentene på plass og koble til ledningene på den måten som er antydet foran. Det kan være lurt å plassere en elektriker tape på undersiden over koblingspunktene slik at de ikke forskyver seg og kortslutter..





På bilde under ser vi den ferdige kretsen. Ei skireim kan fungere som feste rundt håndleddet, enkelte slik har også borrelås.



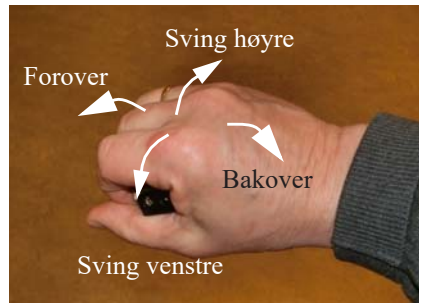
6 Fjernstyring av Bit:Bot

I dette kapittelet skal vi beskrive hvordan vi kan fjernstyre en robot av typen Bit:Bot XL ved hjelp av en håndholdt micro:bit. Vi ønsker å bruke Bit:Bot sitt bibliotek som gjør det lettere å programmere den.

Vi har tatt utgangspunkt i et undervisningsopplegg utviklet av Roy Even Aune ved Vitensenteret i Trondheim. En nettbasert utgave av opplegget finnes på wiki-siden til Vitensenteret: http://wiki.trigger.vitensenteret.com/doku.php?id=introduksjon_til_bitbot

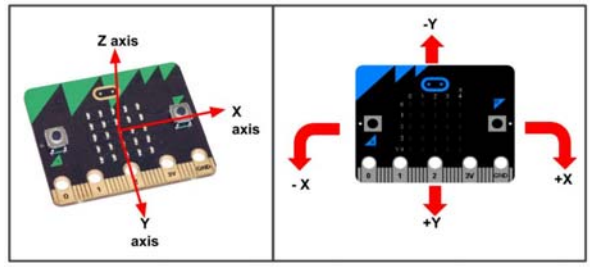
Vi ønsker å fjernstyre roboten ved hjelp av enkle håndbevegelser. For å utføre dette oppdraget trenger vi derfor to micro:bit'er, en batteripakke og en Bit:Bot XL. Den ene micro:bit'en, *sender-enheten*, bruker vi til å sende kommandoer til micro:bit'en som er montert på Bit:Bot'en, *mottaker-enheten*.

Vi ønsker at oppdragets fokus skal være *radiokommunikasjon* og bruk av sender- og mottaker-enhetene hos micro:bit'ene.



For å kunne styre og regulere farten til roboten, skal vi bruke *akselerometeret* i sender-enheten. Siden det kan være mest praktisk å holde kortet på tvers inne i hånda, så velger vi å øke hastigheten framover ved å bøye hånden framover (ned), og tilsvarende øker vi hastigheten bakover ved å bøye hånden bakover (opp). I tillegg ønsker vi å kunne svinge roboten til høyre og venstre, ved å dreie hånden mot henholdsvis høyre og venstre. Dette er mulig når vi vet hvordan akselerometeret i micro:bit'en fungerer.

Figuren til høyre viser akselerometerets akseretninger i forhold til micro:bit-kortet. Siden akselerometeret forholder seg til tyngdeakselerasjonen, som er 1 g i vertikal retning, vil avleste verdier i x- og y-retning bli som i tabellen under avhengig av hvordan vi holder micro:bit'en. Legg merke til at den avleste verdien har benevningen mg (milli g):



Handling	x-retning	y-retning	z-retning
Flatt på bordet, display opp	0 mg	0 mg	- 1000 mg
Flatt på bordet, display ned	0 mg	0 mg	+ 1000 mg
Tilting mot høyre om y-akse, display opp	økende positiv verdi	0 mg	minkende negativ verdi
Tilting mot venstre om y-akse, display opp	økende negativ verdi	0 mg	minkende negativ verdi
Tilting framover om x-aksen, display opp	0 mg	økende positiv verdi	minkende negativ verdi

Handling	x-retning	y-retning	z-retning
Tilting bakover om x-aksen, display opp	0 mg	økende negativ verdi	minkende negativ verdi

Det er viktig å merke seg at vi får alle mellomliggende verdier i x- og y-retning når vi dreier mikro:bit'en fra horisontal til vertikal orientering.

6.1 Grunnleggende programmering av Bit:Bot

La oss begynne med å programmere senderenheten som er den enheten som holdes i hånda. Vi bruker kommandoblokker fra radio-menyen (rød) (se figuren til høyre)

6.1.1 Programmering av sender-enheten

1. Velg radiokanal

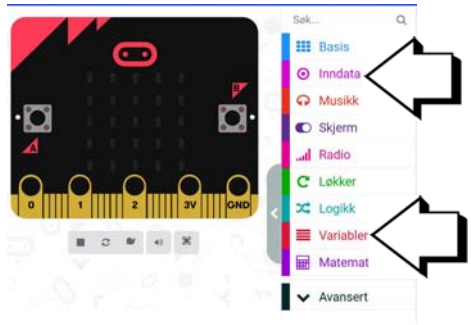
Startblokken hentes fra *Basis-menyen* (blå) og *radio sett gruppe* hentes fra Radio-menyen (rød). Velg en radiogruppe som skiller seg fra de andre om det er flere i rommet som gjør det samme.

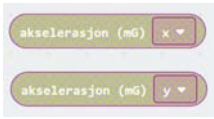
I vårt eksempel har vi valgt *gruppe 10*. For at vår sender-enhet skal kunne kommunisere med vår mottaker-enhet hos roboten, må også den settes til samme gruppe.

2. Les av akselerometrene

Vi skal lese av akselerometeret i x- og y-retning og sende verdiene til mottaker-enheten. Verdiene for *g* leses av i milli *g* (her betegnet mG). Vi finner kommandoblokken for å lese av akselerometeret i menygruppen *Inndata* (fiolett).

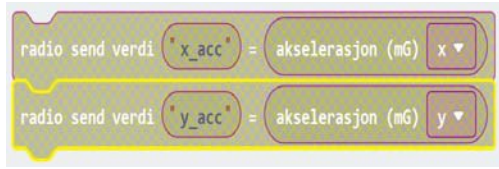
Siden vi skal lese av akselerometeret i både x- og y-retning, må vi gjøre to slike avlesninger, en for hver retning, x og y (se figuren under). Retning velges med den vesle pila til høyre i blokken.





Dernest må vi knytte avlesningene til *identfikatorer* slik at mottakeren vet hvilken måleverdi den mottar. Vi velger å kalle disse for *x_acc* og *y_acc*. De to identifikatorene lager vi ved å skrive dem inn mellom hermetegnene i *radio send verdi*-blokken samtidig som vi legger inn akselerometerverdiene til høyre i blokkene som vist på figuren under. Da får vi knyttet sammen identifikatorer og avlest verdi.

Senderkommandoen *radio send verdi*-blokken finner vi i Radio-menyen (rød):



Blokkene vist i figuren over sender ut identifikatorene og akselerometerverdiene til de andre i gruppen, hos oss bare vår Bit:Bot.

3. Legg inn i loop

Vi gjentar sendingen hele tiden, gjerne med en liten tidsforsinkelse (*Pause*) mellom hver sending. *Pause*-kommandoen finnes i menyen *Basis (blå)*. Vi velger å legge inn 50 millisekunder mellom sending av de to verdiene x*_acc* og *y_acc*.

Da er programmet for sender-enheten ferdig. Før vi sender programmet over til micro:bit'en gir vi programmet et navn og lagrer det. Bruk menylinjen for lagring nederst. Vi har kalt programmet for *Bit-bot Sender 1*

Man velger selv om man vil lagre i skyen (default) eller på egen PC.

Programmet legges over til sender-enhetens micro:bit, ved f.eks. å dra fila over til micro:bit'en som kobles til PC'en med en USB-kabel.

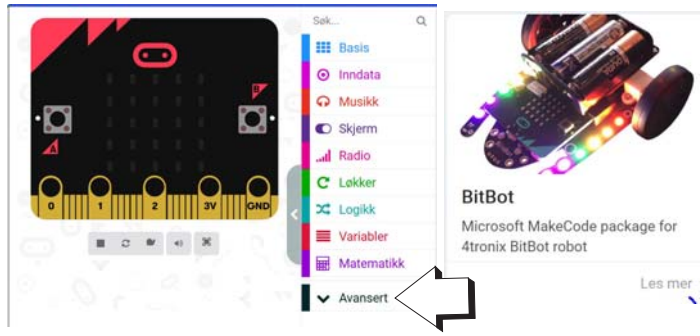
6.1.2 Programmering av mottaker-enheten

Mottaker-enheten er den micro:bit'en som er plugget i Bit:Bot'en og som mottar signalene fra den håndholdte sender-enheten.

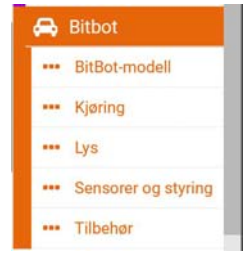
Pass på at du fjerner programmet som har med sender-enheten og skriver inn navnet på programmet til mottakeren. F.eks.: *Bit-bot Mottaker 1*

4. Installer bibliotek for styring av Bit:Bot

For at det skal være lettere å programmere Bit:Bot'en, har noen laget et bibliotek med et ekstra sett av kommandoer. For å kunne ta i bruk disse kommandoene må vi installere biblioteket til Bit:Bot. Biblioteket installeres på følgende måte:



- *Velg menyen Avansert og deretter Utvidelser. Du kommer da til en meny hvor du finner en rekke utvidelser deriblant Bit:Bot. Velg Bit-Bot ved å trykke på rubrikken hvor Bit:Bot er avbildet (figur over til høyre).*
- *Det finnes to versjoner av Bit:Bot: Classic og XL. Sjekk hva slag Bit:Bot du har og velg riktig robot (Skolelaboratoriet har Bit:Bot XL). Figuren til høyre viser en rekke tilleggsmenyer med kommandoer spesiallaget for Bit:Bot.*



5. Velg radiokanal og modell

Hent fra funksjonen ved start fra Basis-menyen (blå). De blokkene som legges i funksjonsgapet til denne utføres bare når programmet startes opp.

Her velger vi samme radiokanal (gruppe) som senderen. Dessuten velger vi *BitBot-modell* fra Bit:Bot-menyen (oransje), og deretter *BitBot modell* ved hjelp av den vesle pila til høyre i blokken, *classic* eller *XL*. Ved å velge *Auto* så vil Bit:Bot'en selv velge riktig bibliotek, *dersom micro:bit'en er plugget inn i roboten når programmet lastes opp*. Det er viktig å velge riktig modell siden det er brukt litt forskjellige porter hos micro:bit'en for å styre de to typene robot. Skolelaboratoriets Bit:Bot'er er som sagt av typen XL.

6. Motta akselerometerverdiene og legg dem i to variabler, X og Y

Vi oppretter de to variablene *X* og *Y*, som skal holde de mottatte verdiene fra *x_acc* og *y_acc*.

Først oppretter vi de to variablene. Dette gjør vi i menyen *Variabler (rød)* og skriver inn de to variablene i innboksen *Lag ny variabel*.

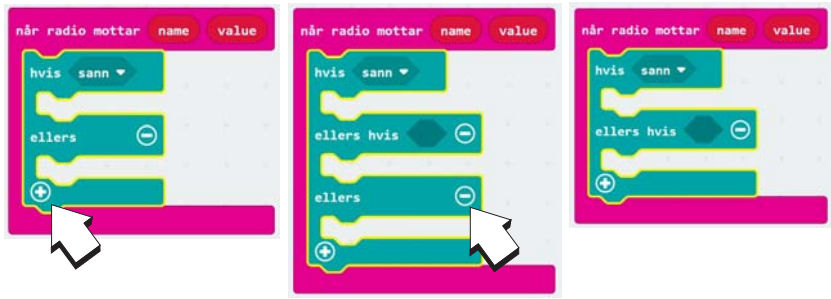
Ved oppstart velger vi å sette de variablene *X* og *Y* til 0 ved å bruke blokken *sett ... til ...* som vi finner i variabel-menyen. Blokken *ved start* kjører, som tidligere nevnt, bare når programmet starter opp.

7. **Lytt etter meldinger på radio**

Dernest skal vi lytte etter radiomeldinger som sender på vår radiogruppe (kanal). Til dette bruker vi kommandoblokken: *når radio mottar "name" "value"* fra Radio-menyen (fiolett). Her kan man enten bruke default navnene *"name"* og *"value"*, eller man kan definere sine egne navn. Vi velger default verdier.

Denne blokka sjekker om det mottas informasjon innen den aktuelle radiogruppen (10). Om så er tilfelle utføres de kommandoene som legges inn i "funksjons-gapet". Når det mottas informasjon, vil *name* og *value* inneholde henholdsvis navnet på den overførte parameteren (*x_acc* eller *y_acc*) og verdien (0 – 1023).

Avhengig av om vi mottar *x_acc* (dvs. name er lik "*x_acc*") eller *y_acc* (dvs. name er lik "*y_acc*") skal vi legge verdien i henholdsvis *X* og *Y* variabelen. For å få til det må vi bruke en *hvis-funksjon* (grønn blokk som vist under).

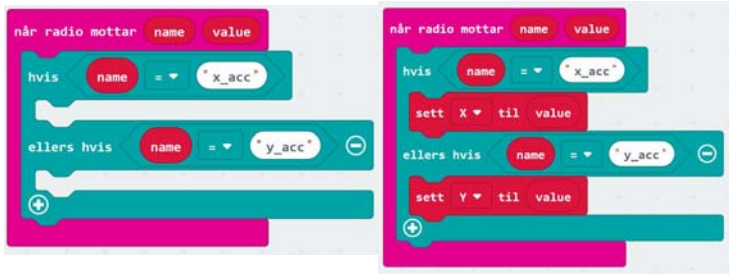


Hent *hvis-ellers-blokken* fra *Logikk-menyen* (grønn). Ved å trykke på + nederst i venstre hjørne av blokken, får vi opp et ekstra *ellers-hvis-felt* som vi ønsker i å bruke her. Vi ønsker imidlertid ikke *ellers-feltet* så denne fjerner vi ved å trykke – til høyre for *ellers*.

En *hvis-ellers-blokk* fungerer slik at ulike kommandoer kan utføres avhengig av hvilken *betingelse* som er oppfylt. Betingelsen setter vi inn i den sekskantede "åpningen".

Dernest legger vi inn betingelsene ved å hente en sammenlignings-blokk fra Logikk-menyen (lyseblå). Vi velger den som sammenligner tekst (med hermetegn, se figuren til høyre).

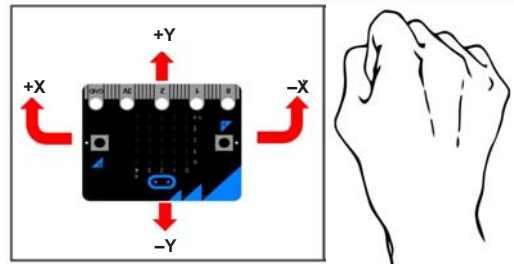
Så legger vi inn *name* på venstre side av betingelsene og skriver inn henholdsvis *x_acc* og *y_acc* mellom hermetegnene til høyre i betingelsen for å sjekke hvilken variabel som er oversendt. Variabelen *name* kan vi kopiere fra den ytre ramma ved kun å dra den over.



Dersom name er lik *x_acc* så skal vi *sette X til value,* dvs. verdien i *value* legges inn i variabelen *X*. Tilsvarende legges *value* inn i variabelen *Y* dersom *name* er lik *y_acc*.

Nå har vi verdiene for *x_acc* og *y_acc* liggende i henholdsvis variablene *X* og *Y.*

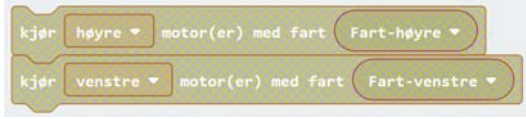
8. Styring av motorene



Vi skal bruke akselerometerverdiene mottatt fra sender-enheten til å styre roboten. Som vi har omtalt tidligere så mottar vi positive og negative verdier fra *x_acc* (*X*) og *y_acc* (*Y*) i henhold til figuren til høyre. Neven lengst til høyre viser i hvilken retning den holder micro:bit'en.

Vi skal nå bruke kommandoer fra Bit:Bot-menyen (oransje) som vi installerte for litt siden.

Roboten har to motorer, en motor til hvert av hjulene. Ved å kjøre de to hjulene med forskjellig fart, vil roboten svinge. For å kjøre motorene med en gitt fart bruker vi kommandoblokkene *kjør høyre/venstre motor(er) med fart ....* Det er viktig at vi kan styre de to motorene uavhengig av hverandre.



I tillegg kan vi definere to variabler *Fart-høyre* og *Fart-venstre* som vi gjør i variabel-menyen (rød).

Vi vet at *X*-verdien som kan være både positiv og negativ, skal kontrollere forskjellen mellom hastigheten til motorene, og *Y*-verdien, som også kan være positiv eller negativ, skal styre framdriften. Positive verdier vil drive roboten framover, og negative verdier vil drive roboten bakover. La oss sette opp noen ligninger som beskriver robotens bevegelser framover og bakover:

$$\text{Fart-høyre} = Y \tag{6.1}$$

Fart-venstre = Y (6.2)

Dersom roboten skal svinge til høyre, må farten på venstre hjul øke og farten på høyre hjul avta, og omvendt om den skal svinge til venstre. Denne variasjonen styres av X-verdien. Vi kan da modifisere formlene våre slik:

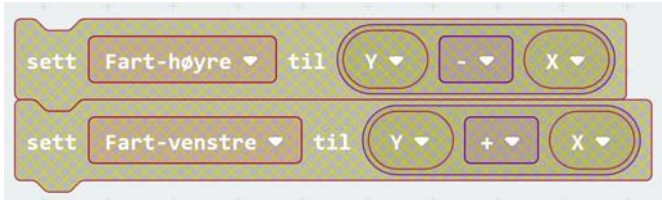
Fart-høyre = $Y - X$ (6.3)

Fart-venstre = $Y + X$ (6.4)

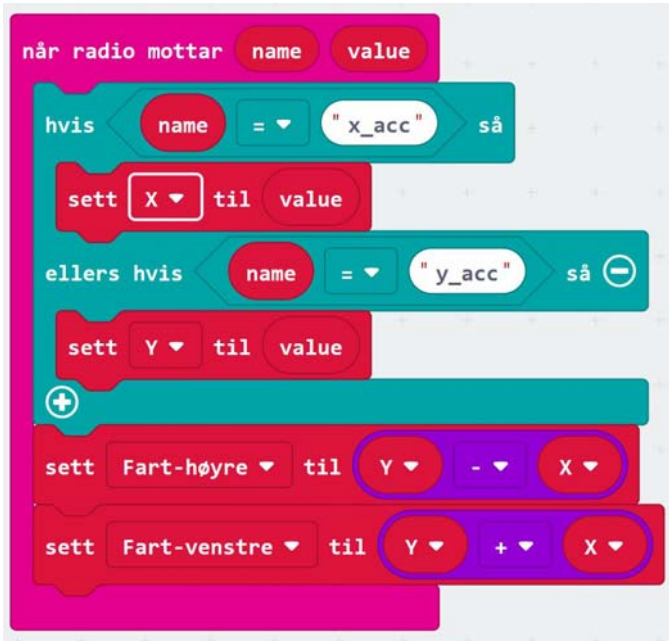
Forsøk å resonner dere fram til hvorfor vi har valgt fortegnene slik som vist. Om vi har valgt rett får dere svar på når dere tester programmet i roboten.

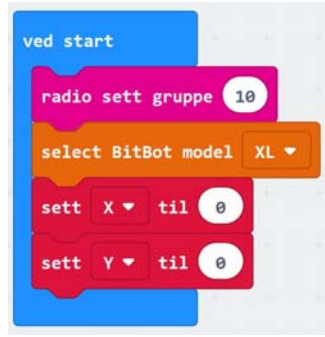
9. Beregn farten på hver av motorene

For å beregne farten bruker vi kommando-blokken *sett variabel til*.



Om vi setter inn disse blokkene etter at verdiene er hentet fra radiomottakeren, vil programmet kunne se slik ut:

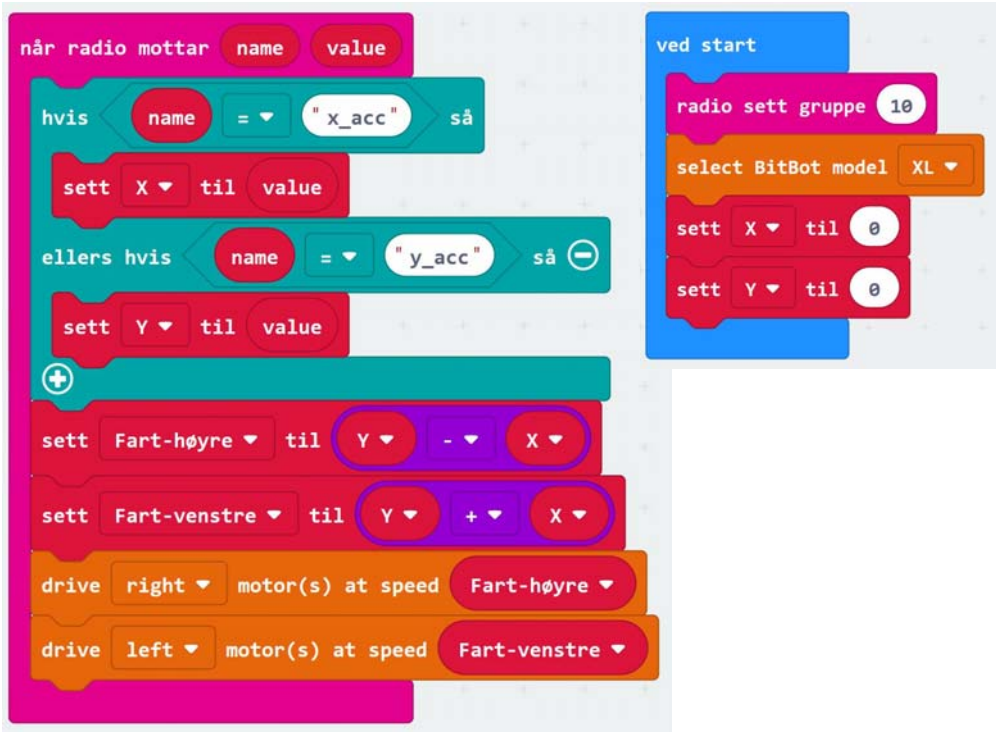




Nå har vi beregnet farten til de to motorene som er et tall og tallet er lagt i de to variablene *Fart-høyre* og *Fart-venstre*.

10. Sett fart på motorene

For å sette fart på motorene bruker vi to kommandoer fra Bit:Bot-menyen.



Legg merke til at *ved start*-blokken inkluderer variablene *X* og *Y* som ved oppstart er satt til 0, dvs. at vi alltid starter med å stå i ro før den første verdien kommer fra den håndholdte senderen.

11. Overføring av programmet til micro:bit

Dernest kan programmet flyttes over til micro:bit'en[8].

12. Forenkling

Ser du en måte som gjør at programmet kan forenkles med færre blokker?

8. NB! Pass på at Bit:Bot løftes fra bordet når den slås på. Dersom den står på bordet vil lys-sensorene (refleksjonssensorene) på undersiden av roboten vanligvis registrere mørke og gå inn i parringsmodus for bluetooth, hvilket vi ikke ønsker i denne omgangen. Det er mulig at dette kun er nødvendig for Classic versjonen av roboten, men greit å være klar over.

Inntil videre hopper vi over punkt 8 og 9. Uttestinger viser at denne ekstra finessen ikke er nødvendig. Dere kan ev. legge den inn senere.

6.2 Tilleggsoppgaver

Dette avsnittet viser programmering av noen ekstra egenskaper ved Bit:Bot'en

1. Endre på følsomheten på styrefunksjonen

Vi ønsker også å kunne justere *følsomheten* til styrefunksjonen. Dette kan vi gjøre ved å multiplisere *Y* og *X* med to følsomhetsfaktorer, *fx* og *fy*. Disse kan f.eks. ha verdier fra 0 til 2. Verdier mellom 0 og 1 vil redusere følsomheten, mens verdier mellom 1 og 2 vil øke følsomheten i forhold til verdien 1 som ikke gir noen justering av følsomheten.

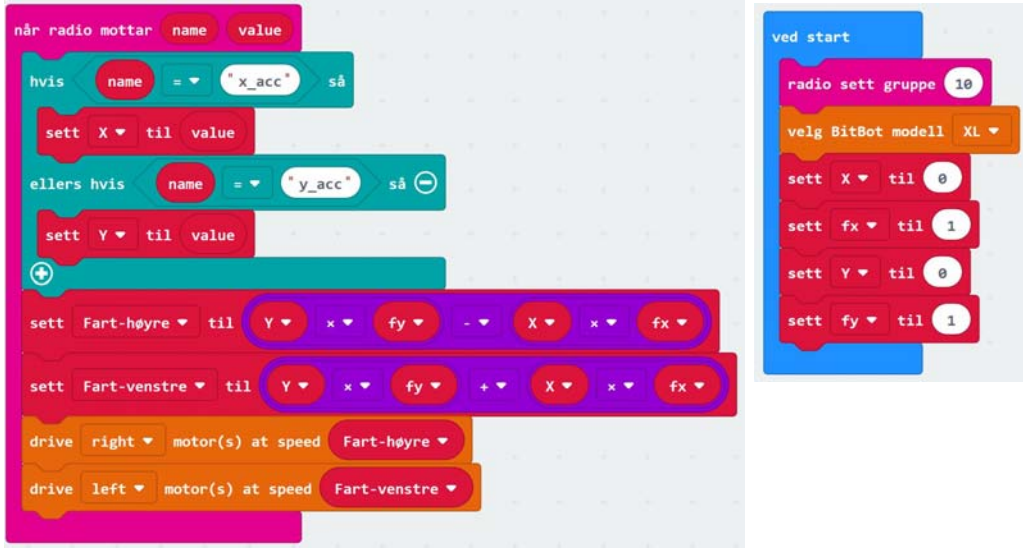
Økt følsomhet betyr at håndbevegelsen får større konsekvenser for farten, mens redusert følsomhet betyr at håndbevegelsen for mindre konsekvenser for farten.

$$\text{Fart-høyre} = Y*fy - X*fx \tag{6.5}$$

$$\text{Fart-venstre} = Y*fy + X*fx \tag{6.6}$$

I tillegg kan vi legge inn startverdier for *fx* og *fy* i oppstartsrutinen *ved start*. Husk og definer variablene *fx* og *fy* under menyen *Variabler*.

I utgangspunktet har vi gitt *fx* og *fy* verdien 1 hvilket betyr at vi hverken har redusert eller økt følsomheten.

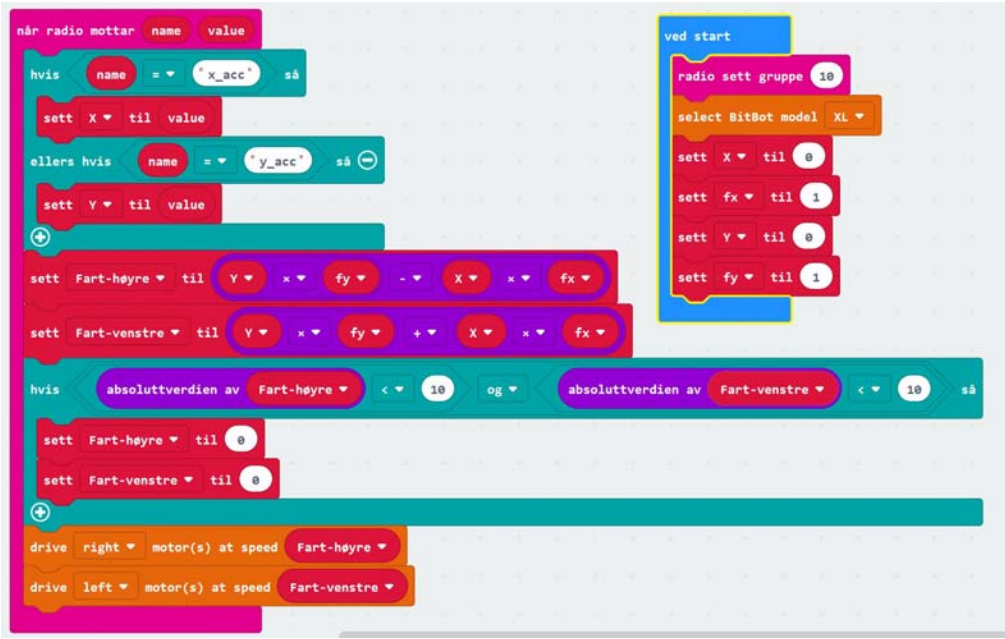


2. Sett opp et stoppvindu

Noen ganger kan det være vanskelig å få roboten til å stå helt stille. For å gjøre det lettere å holde roboten i ro når vi ønsker det, så kan vi sette *Fart-høyre* = 0 og *Fart-venstre* = 0 dersom verdien av de to variablene er under en viss *terskelverdi*, f.eks. 10. Siden dette gjelder både framover (+ verdi) og bakover (– verdi), kan vi bruke den matematiske funksjonen *absoluttverdi* når vi skal undersøke om verdiene er under terskelverdien.

For å teste om farten er mindre enn 10 bruker vi en *hvis-blokk som vi finner under menyen Logikk* (grønn).

Dermed skulle det ferdige programmet bli som vist i figuren under.



3. Overføring av programmet til micro:bit på BitBot

Dernest kan programmet flyttes over til micro:bit'en[9].

6.2.1 Lys og lyd hos Bit:Bot

En kan se for seg en rekke forskjellige utvidelser av funksjonen til roboten. Her er noen forslag:

1. La roboten lage lyd når du trykker på knapp A.
2. Slå på lys ved å trykke på knapp B.

9. NB! Pass på at Bit:Bot løftes fra bordet når den slås på. Dersom den står på bordet vil lys-sensorene (reflektanssensorene) på undersiden av roboten vanligvis registrere mørke og gå inn i parringsmodus for bluetooth, hvilket vi ikke ønsker i denne omgangen. Det er mulig at dette kun er nødvendig for Classic versjonen av roboten, men greit å være klar over.

3. Skift mellom ulike farger på lyset når du trykker gjentatte ganger på knapp B. I løpet av sekvensen skal lysene være avslått.
4. Skriv et program som gjør at roboten følger en svart linje på gulvet.
5. La roboten skifte mellom å følge en linje og bli fjernstyrt når knapp A trykkes.

7 Referanser

[1] Rossing N.K., Micro:bit - Forslag til undervisningsopplegg, Rev. 5.0 21.02.2020, Vitensenteret, Trondheim

[2] Nordic Gazell protokoll
https://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.sdk51.v9.0.0%2Fgzll_02_user_guide.html&cp=4_0_7_5_0_2

[3] Micro:bit radio
https://infocenter.nordicsemi.com/pdf/nRF51_RM_v3.0.1.pdf?cp=5_2_0

Vedlegg A Flere eksempler på oppgaver til Bee:bot

Intro til småskolelærerne

Lærerne i småskolen fikk dette som en første introduksjon til Bee:bot

Figur A.1 Eksempel som kan passe i småskolen som stavetrening

Tanken med denne oppgaven er at deltakerne skal programmere Bee:bot slik at den beveger seg mellom rutene med bokstaver slik at de danner et navn. Bee:boten kan ev. programmeres til å ta en kort pause i de valgte rutene for å markere sitt valg. Da oppgaven ble gitt til lærere i småskolen var tanken at de skulle ende opp med navnet ARNE. Imidlertid var ikke ARNE det foretrukne navnet, men ERNA (siden det var på valgdagen 11. sept. 2017)

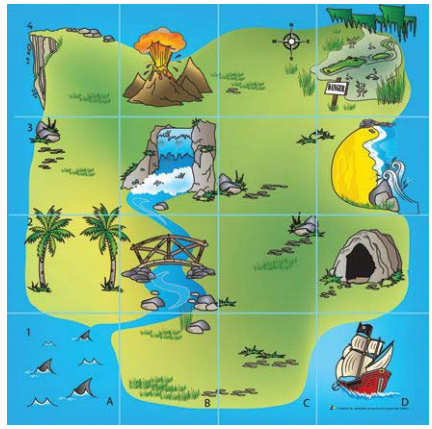
Dernest fikk lærerne fra småskolen følgende utfordring:

Figur A.2 Oppgavetekst til småskolelærere

Under er vist noen eksempler som kom opp under diskusjonen blant lærerne i småskolen:

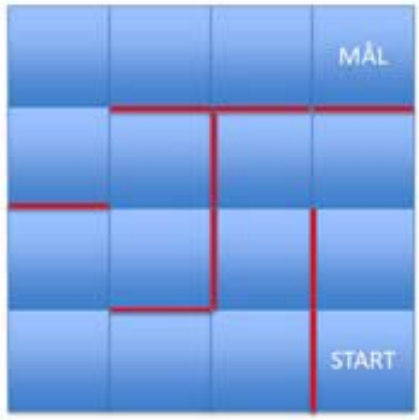
Figur A.3 Forslag fra en gruppe lærere

Andre foreslo at de skulle lage et kart over nærområdet ved barnehagen og la Bee-bot'en bevege seg på kartet, ev. at en av lærerne kunne angi en sekvens som barna skulle realisere på kartet for så å finne ut hvor i barnehagen eller skolegården de skulle lete etter en skatt eller nye instruksjoner.



Figur A.4 Forslag til skattekart som egner seg til å utforskes med Bee-boten

Det ble også foreslått at Bee-bot'en burde egne seg godt til å trene høyre/venstre forståelse med barna. Dette kan være en utfordring for mange, spesielt dyslektikere. Å gå en labyrint fram og tilbake samme vei er god trening for disse. En slik utfordring krever at man tenker omvendt når man skal rygge tilbake. Alternativt kan man dreie roboten helt rundt og gå løypa motsatt vei.



Figur A.5 Undersøk hva som skjer med høyre og venstre når man går fram og tilbake

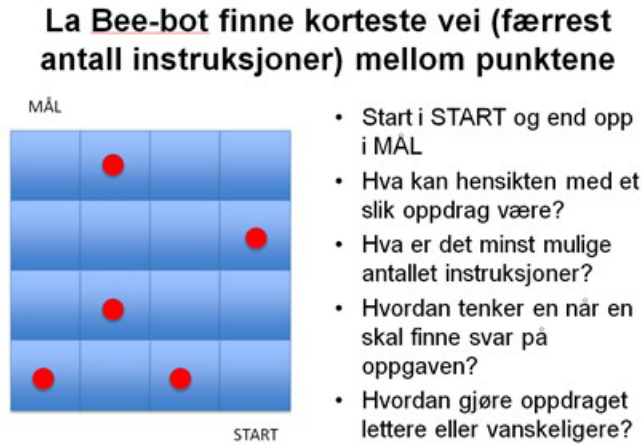
Andre foreslo å fylle brettet med tall og regnesymboler slik at elevene kunne sette opp bevegelsesekvenser som ga gyldige regnestykker:

Figur A.6 Lag regnestykker som er gyldige med tall og tegn slik at stykket går opp

Her er noen flere forslag til bruk av Bee:bot i undervisningen og som passer både for barn og voksne. I eksempelet under er det foreslått å bevege seg mellom former med ulike antall kanter.

Figur A.7 Beveg deg mellom ruter med ulik antall sidekanter

I eksempelet under kan en for eksempel bevege seg mellom punktene på kortest mulig antall forflytninger. En kan for eksempel sette som krav at en ikke skal være innom en rute med rødt punkt mer enn en gang.



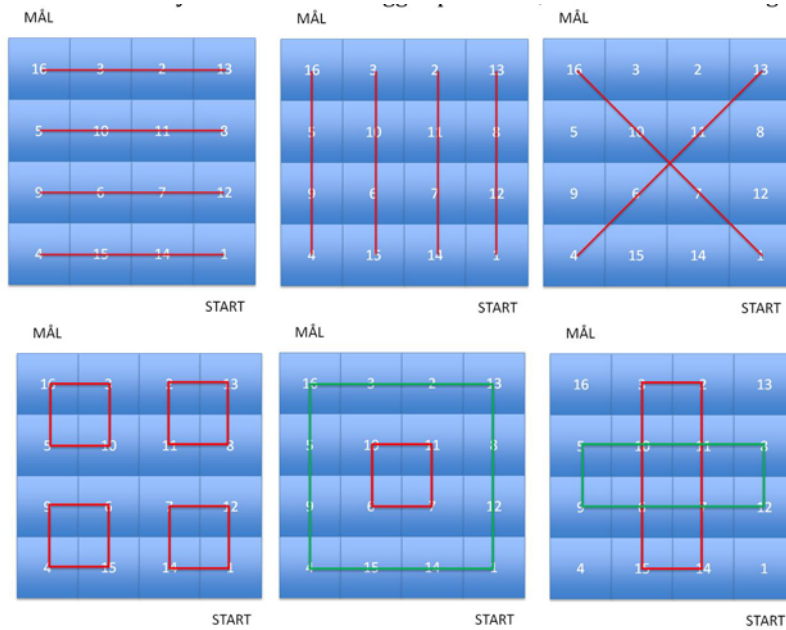
Figur A.8 Gå fra start til mål og vær innom hver rute med punkt bare en gang. Bruk færrest mulig trekk

I den neste har vi plassert tallene 1 til 16 i de 16 rutene. Oppgaven går ut på gå innom fire ruter med tall som til sammen gir verdien 34 når en beveger seg fra start til mål. En markerer tallet ved å stopp litt opp i rutene med de aktuelle tallene. Nestemann i rekke må finne en annen tallkombi-

nasjon som gir 34. Det er ikke lov å bruke alle fire tallene om igjen mellom hver runde. Den siste som klarer å finne en ny tallkombinasjon har vunnet.

Figur A.9 Gå innom fire tall som til sammen gir summen 34

Denne oppgaven har flere løsninger enn en først skulle tro. Under er vist noen forskjellige muligheter. Dette trenger en ikke fortelle deltagerne, men la dem oppdage det under veis. De kan få et kvadrat med tallene og tegne inn rutene etter som de blir valgt. I figuren under er vist alle varianter som er symmetriske eller ligger på en rad, kolonne eller en diagonal.



Figur 7.1 Her ser vi de 18 symmetriske løsningene med fire tall som til sammen gir 34

Det kan imidlertid finnes usymmetriske løsninger som også gir 34. Kan du finne noen av disse?

I den neste oppgaven skal deltagerne lage flest mulig forskjellige navn. Navnene skal være kjente og man har ikke lov til å gjenta samme navnet to ganger i samme omgang.

Figur A.10 Lag flest mulig forskjellige gutte- og jentenavn. Hvem holder ut lengst?

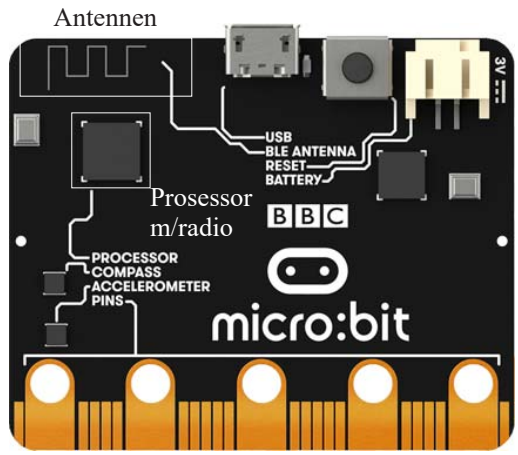
Slik kan man fortsette med å lage ulike oppgaver tilpasset til alderstrinn og de ferdighetene man ønsker å øve på.

Dette er er fine øvelser for å trene programmering og sekvensiell tenkning slik de fleste dataprogrammer er bygget opp. Man lager seg en rekkefølge av instruksjoner som til sist løser en oppgave.

Vedlegg B Slik virker radioen til micro:bit

La oss se litt nærmere på radiodelen av Micro:bit'en.

Radioen er uten tvil den enkeltegenskapen som gjør Micro:bit'en til et så kraftfullt mikrokontrollerkort. Radioen gjør det mulig å kommunisere trådløst mellom to mikro:bits eller grupper av mikro:bits, og til andre enheter som f.eks. en PC. Radioen omtales gjerne som en BLE hvilket betyr Bluetooth Low Energi radio. Antennen er også integrert på kortet og kan sees som en sik-sak-bord øverst i venstre hjørne på baksiden. Selve radiomodulen er integrert i prosessoren som er den svarte kvadratiske kretsen under antenna.

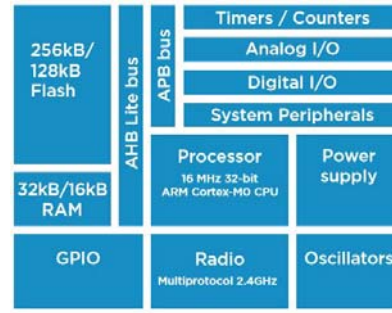


Sendefrekvens og antennelengde

Senderfrekvensene er lagt til 2,4 GHz området som er et lite frekvensbånd som brukes til mange ulike ting. Her finner vi bl.a. microbølgeovner[10] og andre kortdistanse sendere. Bølgelengden i dette frekvensbåndet er ca. 12,5 cm. Siden en vanligvis bruker antenner som har en lengde på en halv eller kvart bølgelengde, blir de ganske små. Antennelengden på mikro:bit'en er ca. 3 cm hvilket er ca. 1/4 bølgelengde. Dersom antennen er plassert på et kretskort, vil også lengden av antenne bli noe kortere enn om den hadde vært strekt ut i rommet.

B.1 nRF51822 – Nordic Semiconductor[11]

Det norske firmaet *Nordic Semiconductor* med hovedkontor i Trondheim, har lagt et gullegg med den kombinerte mikroprosessoren og radioenheten nRF51822. Selve prosessoren er en 32-bit ARM-prosessor, med klokkefrekvens 16 MHz, som anvender en såkalt SoC teknologi (System on Chip). Dvs. at de fleste funksjonene i en kraftig mikrokontroller er plassert på samme brikke (substrat), gjerne også med en radioenhet (sender og mottaker - *transceiver*) på chip'en, som også er tilfelle for nRF51822. Fordelen med en slik løsning er at systemet kan gjøres ekstremt strømbesparende, da det ofte er effektkrevende å føre signaler fra en chip (krets) over til en annen.



Blokkdiagram for nRF51822

10. Mikrobølgeovner opererer normalt på 2,45GHz Mikrobølgeovener med lekkasje av mikrobølger kan derfor lett forstyrre kommunikasjonen mellom micro:bits.
11. https://www.nordicsemi.com/-/media/Software-and-other-downloads/Product-Briefs/nRF51822-product-brief.pdf

Mikrokontrolleren nRF51822 har 31 generelle inn/utganger (GPIO). En AD-konverter på 10 bit gjør det mulig også å koble til analoge signaler. Kretsen inneholder i tillegg en intern temperatursensor.

B.2 Radioen kan operere på to forskjellige måter

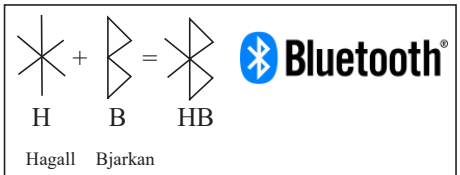
Radioen kan operere på to forskjellige måter[12]:

1. Den første omtales som en "peer to peer connectivity". Dvs. at kommunikasjonen skjer mellom to "likemenn" (peers). Et slikt system er uten "sjefer" som har kontroll over sendingene. Det gjøres heller ingen "avtaler" mellom sender og mottaker. En krets sender ut en melding og "håper" at noen fanger opp signalet. Det er denne varianten vi bruker når vi skal sette opp en enkel kommunikasjon mellom to micro:bits, eller mellom en micro:bit og flere andre. I tillegg kan flere grupper av micro:bits kommunisere med hverandre innen samme *gruppe* uten å forstyrre andre grupper. Micro:bits som skal kommunisere med hverandre må befinne seg innen samme gruppe som bestemmes av den som programmerer kretsene.

   Som all annen digital kommunikasjon overføres dataene i "pakker", dvs. et visst antall bit som er organisert på en bestemt måte, dvs. en "pakke". Skal større mengder data overføres så sendes flere pakker etter hverandre.

2. Den andre er en ordinær *bluetooth radiokanal*. Bluetooth er en kortholds radiostandard utviklet første halvdel av 90-tallet hos Ericsson Telecommunication for bl.a. å kunne kommunisere trådløst mellom elektroniske enheter som f.eks. mobiltelefoner. Bluetooth opererer vanligvis i frekvensområdet 2,400 GHz – 2,485 GHz (hele ISM[13]-båndet er fra 2.4 – 2,5 GHz, en båndbredde på 100 MHz[14], eller 50 *kanaler* á 2 MHz, hvorav normalt kun 40 er for generell bruk). Micro:bit bruker en variant av bluetooth som kalles Bluetooth Low Energy (BLE). Den eneste forskjellen er at den sender med mindre effekt og er billigere å lage og å bruke. Den har derfor noe kortere rekkevidde enn tradisjonell bluetooth.

   Bluetooth standarden brukt hos micro:bits kan normalt kobles opp mot mobiltelefoner. For å oppnå kontakt mellom en micro:bit og en telefon utføres en "pairing". Dette kan gjøres på flere ulike måter som er godt beskrevet i "BBC micro:bit Bluetooth Profile"[15].



Hagall Bjarkan

Det fortelles at da de tre industrigigantene Intel, Nokia og Ericsson møttes i 1996 for bestemme seg for en kortdistanse radio, så foreslo ***Jim Kardach*** fra Intel og gi systemet arbeidstittelen Bluetooth etter den danske kongen Harald Blåtann siden han lyktes med å samle Skandinavia, slik de ønsket å forene PC og mobilverdenen. Dermed ble det slik. Symbolet for Bluetooth er også en sammensmelting av runebokstavene for H (Hagall) og B (Bjarkan), som er initialene til kongen.

12. https://www.littlebird.com.au/a/how-to/112/bluetooth-with-micro-bit
13. ISM - "Industrial, scientific and medical"
14. https://en.wikipedia.org/wiki/ISM_band
15. https://lancaster-university.github.io/microbit-docs/ble/profile/

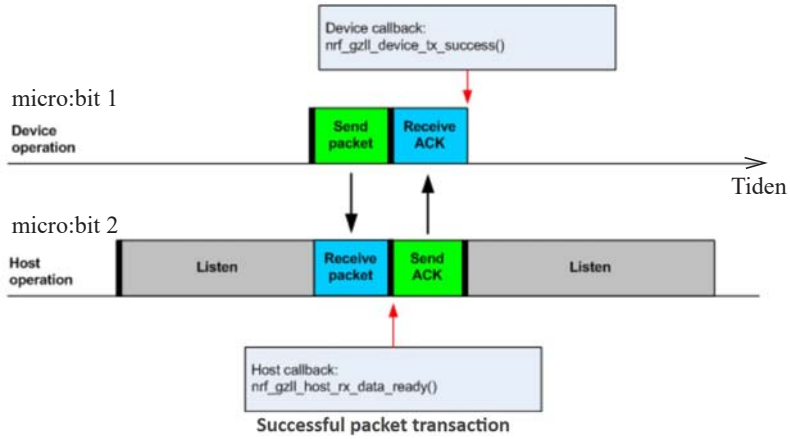
Senderdelen av radioen kan programmeres til å sende med forskjellige effekter fra -20dBm til +4dBm Hvilket betyr fra ca. 0,01 mW til 2-3 mW som er svært små effekter[16], men nok for kort-distansekommunikasjon (opp til ca. 70 m med fri sikt). Kretsen kan operere fra 1,8 – 3.6 V, dvs. på svært lave spenninger som også betyr lave effekter. Mottakerfølsomheten varierer fra -93 dBm til - 85 dBm, som er rimelig bra. Dataraten i radiokommunikasjonen kan settes til fra 250 kbps – 2 Mbps[17].

B.3 Peer to peer kommunikasjon[18]

I dette avsnittet skal vi se nærmere på hvordan kommunikasjonen mellom to micro:bits foregår.

Den pakken som brukes i dette tilfellet er spesiell for Nordic Semiconductor (og kan omtales som proprietær) og går under navnet Nordic Gazell. I denne protokollen er hver pakke merket med en gruppekode ("group code") som inneholder en adresse og annen informasjon som er nødvendig for at dataene skal komme fram til adressaten på rett måte. Det er denne adressen som settes når vi velger "gruppe" når vi skal kommunisere med micro:bits.

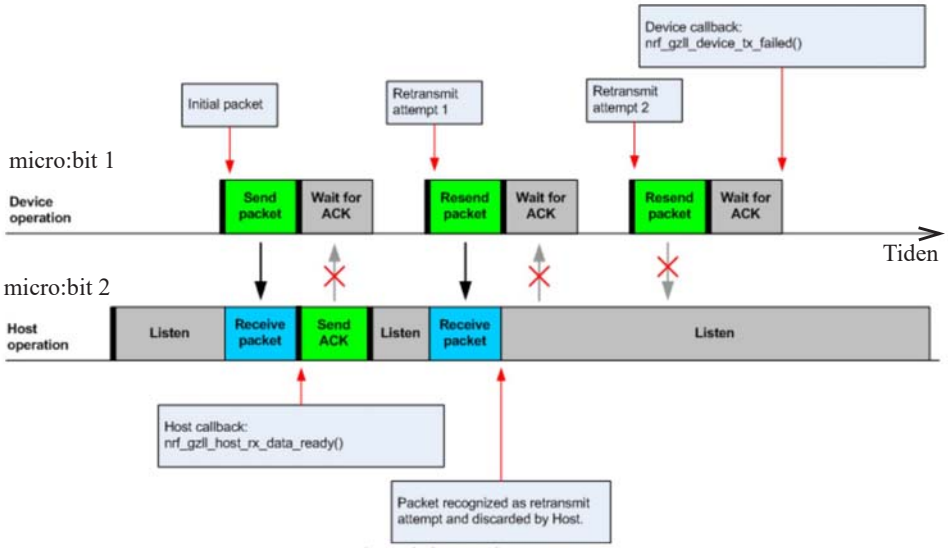
Figuren under viser hvordan en kan tenke seg at selve kommunikasjonen skjer.



Vi tenker oss at micro:bit 1 (Device) "ønsker" å sende en melding til micro:bit 2 (Host). Micro:bit 1 sender en "pakke" med bit. Denne mottas av flere micro:bits deriblant micro:bit 2. Denne sjekker adressen (gruppe) for å se om pakken er ment for den. Når alle bitene i pakken er mottatt, sendes det et svar tilbake til micro:bit 1 om at datapakken er korrekt mottatt (ACK – acknowledge) og at den er klar til å motta en ny datapakke.

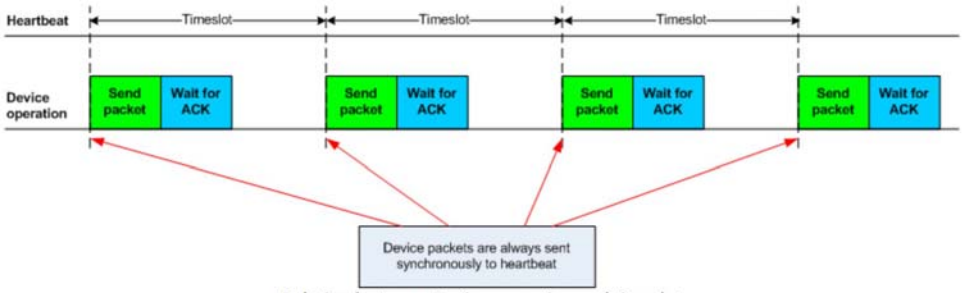
16. En vanlig mobiltelefon kan sende på effekter opp til 2Watt, riktignok i kort pulser.
17. kbps - kilo bit pr. sekund. Mbps - Mega bit pr. sekund
18. https://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.sdk51.v9.0.0%2Fgzll_02_user_guide.html&cp=4_0_7_5_0_2

Dersom micro:bit 2 “merker” at det er feil i dataene som den mottar, varsler den om at pakken ikke er mottatt korrekt og ber om at micro:bit 1 sender den på nytt. Ev. kan micro:bit 1 “erfare” at den ikke mottar noen kvittering (ACK), og sende pakken på nytt. En pakke vil kunne bli sendt om igjen flere ganger (3 ganger) før senderen gir opp.



Example on failing package transaction.

For at overføringen av data skal skje i ordnede former så sendes og mottas dataene i *tidsvinduer (Timeslot)* eller såkalte “hjerteslag” som vist i figuren under.



Relation between Device operation and timeslot

Dvs. at tidsrommet for hver ny pakke som sendes er hele tiden det samme (*Timeslot*). Dette krever at både sender og mottaker er synkronisert i forhold til hverandre. Lengden av et “ timeslot” varierer med datahastigheten[19]:

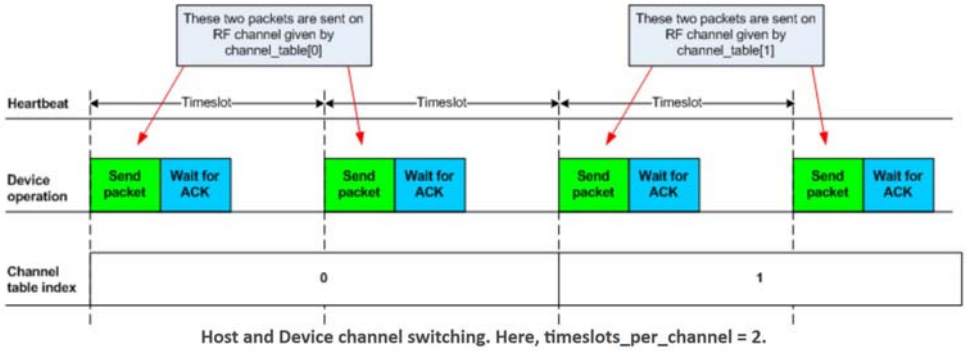
- For 2 MBit/sek er timeslot perioden >= 600 µs.
- For 1 MBit/sek er timeslot perioden >= 900 µs.

19.https://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.sdk51.v9.0.0%2Fgzll_02_user_guide.html&cp=4_0_7_5_0_2

- For 250 kBit/sek er timeslot perioden >= 2700 µs.

Frekvenshopping

Siden det kan være mye støy på de frekvensene som brukes gjør man bruk av *frekvenshopping*. Det vil si at med jevne mellomrom endres sendefrekvensen. Dersom det er støy på én frekvens så kan man håpe på at det er støyfritt på neste slik at en ev. gjentatt pakke vil komme fram til tross for at den mislyktes med første sending. Figuren under viser hvordan senderen skifter kanal etter bare å ha sendt to pakker

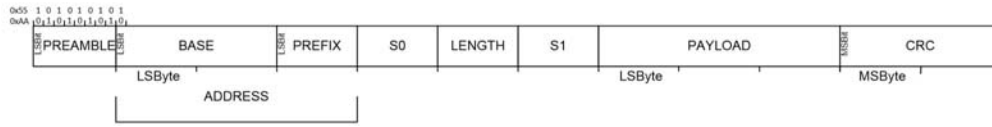


Host and Device channel switching. Here, timeslots_per_channel = 2.

Når senderen skifter frekvens, så må også mottakeren skifte samtidig, ellers mister den pakken. Både sender og mottaker må derfor bli enige om å bruke samme tabell over hvordan frekvens-hoppingen skal utføres. Denne informasjonen må derfor overføres helt i starten av sendingene i det og kalles "Channel tabel index". Både sender og mottakere må slå opp tabellen som inne-holder frekvensinformasjonen for å vite hvilke frekvenser de skal hoppe til, i tillegg til at frekvenshoppingen skjer synkront.

B.4 Datapakkenes oppbygging[20]

Figuren under viser et eksempel på hvordan en pakke er satt sammen av mange deler med ulik funksjon.



20.https://infocenter.nordicsemi.com/pdf/nRF51_RM_v3.0.1.pdf?cp=5_2_0 side 82

“Preamble” (innledning)

Dette er en sekvens av 01010101 (0x55) eller 10101010 (0xAA)[21]. Hvilken som velges avhenger av den etterfølgende adressen. Dersom første bit i adressen er 0, så velger man 01010101. Tilsvarende velger man 10101010 dersom første bit i adressen er 1. Årsaken er at man ikke ønsker to like bit i overgangen mellom preamble og adresse.

“Adressen”

Adressen består av to deler BASE (2 byte) og PREFIX (1 byte). Adressen angir hvilken enhet eller enheter man ønsker å kommunisere med. Dersom den utsendte adressen stemmer med adressen hos mottakeren, så tas nytteinformasjonen (PAYLOAD) vare på i mottakeren, ellers forkastes den. Det er adressen som bestemmer hvilken gruppe micro:bit’ene skal tilhøre.

“S0” og “S1”

S0 og *S1* er to byte (2x8 bit) hvor enkeltbitene gir mottakeren informasjoner om hvordan meldingen skal tolkes.

“LENGTH”

Angir lengden på den nyttige informasjon som skal overføres (PAYLOAD). Maksimal lengde på S0 + LENGTH + S1 + PAYLOAD er 254 byte, eller sagt på en annen måte ca. 250 tall og bokstaver.

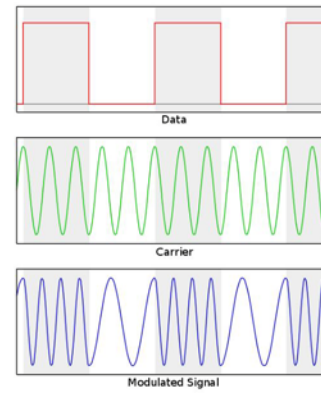
CRC (Cyclic Redundancy Check)

Dette er 2 byte (2x8 bit) som brukes til å sjekke om de mottatte dataene inneholder feil. Dersom noen få bit er feil så vil informasjonen i de to CRC-bytene til en viss grad kunne brukes til å rette feilene slik at meldingen blir riktig. Blir antallet feil for stort, klarer ikke disse to bytene å rette opp feilen men klarer å påvise *at den er feil* og ber om at meldingen sendes på nytt.

21. 0xAA er tall uttrykt i det hexadesimale tallsystemet f.eks. 1010 1010 = AA hvor man benytter grunntall 16, tallrekken blir da 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Fire bit kan da uttrykkes med en bokstav 1010 = A. Dette skrives på formen (0xA)

B.5 Modulasjon

Når dataene skal overføres så gjøres det ved å endre frekvensen på det utsendte signalet ørlite grann, slik at en digital 1'er og en digital 0'er har litt forskjellig sendefrekvens. Vi sier at signalet er *FSK modulert (Frequency Shift Keying)*. På figuren til høyre ser vi øverst det binære datasignalet med 0'ere og 1'ere. Midt i figuren ser vi *bærefrekvensen* som i vårt tilfelle er frekvensen på ca. 2,4 GHz. Nederst ser vi hvordan bærefrekvensen endres i takt med dataenes 0 og 1. Det må bemerkes at frekvensendringen er sterkt overdrevet i figuren.



Bærefrekvensen er den frekvensen som er valgt for overføringen av signalet og kan i prinsippet velges hvor det er plass eller hvor myndighetene har bestemt at denne typen sendinger skal skje.

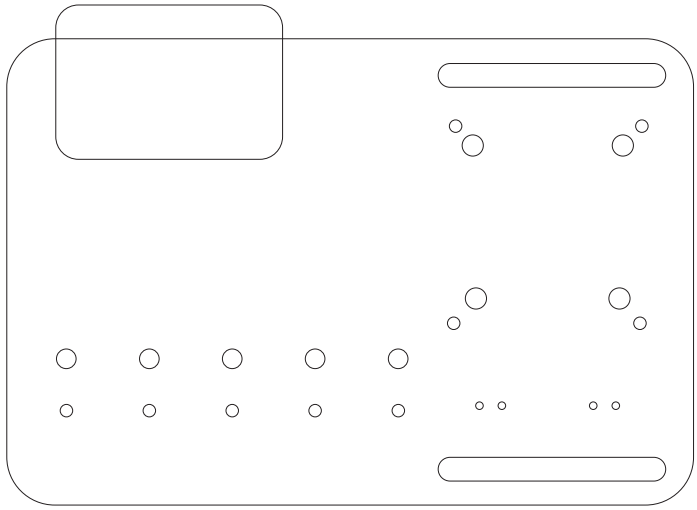
I dette eksempelet skifter frekvensen brått som følge av endringen i det digitale signalet. En slik endring kalles for *Binært FSK* eller *BFSK*.

I mikro:bit brukes en modulasjonstype som kalles *GFSK*, eller *Gaussisk FSK*. Hvilket betyr at skiftet mellom frekvensene skjer langsommere og ikke brått som i BFSK. En langsommere endring gjør at det utsendte signalet ikke tar så mye plass i frekvensbåndet. Dvs. man kan overføre en større datamengde i en kanal med en gitt båndbredde.

På mottakersiden må en detektere disse små endringene i frekvens og gjøre dem om til et digitalt signal med 0'er og 1'ere. Helst med så få feil som mulig. Dette kalles å *demodulere* signalet og den komponenten som gjør det kalles en *demodulator*.

Vedlegg C Mal for armbånd

C.1 Mal til laserkuttet armbånd på MDF



Vedlegg D Bruk av micro:bit og nettbrett

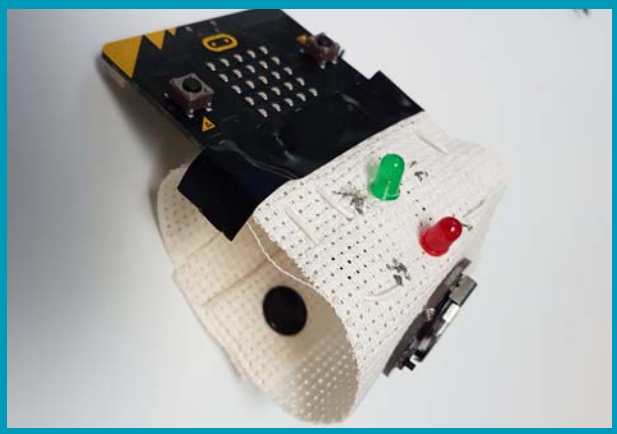
Det er også mulig å bruke nettbrett istedet for PC eller Mac. Videoen på denne nettsiden beskriver hvordan dette lar seg gjøre:

https://www.nrk.no/skole/?page=search&q=super%3Abit+ipad

Her finner du to videoer som forteller hvordan du skal koble mikrobit opp mot iPad og hvilke utfordringer du kan møte:

super:bit - slik løser du vanlige iPad-problemer

super:bit - slik bruker du micro:bit med iPad



Heftet er skrevet som en hjelp til gjennomføring av fjerde samling av DeKom-tilbudet: Skapende aktivitet i klasserommet, som ble gitt til Okstad skole, Tomasskolen og Vikhammer/Vikhammeråsen skole høsten 2020.

Målsetingen med denne tredj og fjerde samlingen er å gi deltakerne en grunnleggende innføring i programmering med micro:bit samtidig som det skal være et eksempel på hvordan en kan bruke programmering for å bygge opp et produkt (prosjekt). Det er lagt vekt på sentrale begreper fra læreplanen og å vise hvordan man kan lage og teste små programbiter som etter hvert kan settes sammen til et større program.

Den fjerde samlingen bygger på samling tre og utforsker to viktige egenskaper ved mikto:bit: Trådløs kommunikasjon og det innebygde akselerometeret. Begge disse egenskapene er knyttet til et gjennomgående prosjekt – *Smart armbånd*, hvor en kombinerer den håndverksmessige med det programmeringstekniske som vi anser som særdeles viktig.

Nils Kr. Rossing (nkr@vitensenteret.com)
Dosent i naturfagdidaktikk ved NTNU og prosjektleder ved Vitensenteret i Trondheim.

Ola Kleiven (ola@vitensenteret.com)
Lærer og prosjektleder for Super:bit-prosjektet ved Vitensenteret i Trondheim

Rannvei Sæther (rannvei@vitensenteret.com)
Pedagog ved Vitensenteret i Trondheim

Anne Birgitte Belboe (annebirgitte@vitensenteret.com)
Lærer og skaperlærer ved Vitensenteret i Tr.heim

Eva H. Hagen (eva@vitensenteret.com)
Leder formidleravdelingen Vitensenteret i Tr.heim

ISBN 978-82-92088-
Rev. 1.1 – 10.11.20