KreTek

Nils Kr. Rossing

MICRO:BIT

Radiokommunikasjon – Forslag til undervisningsopplegg

Bit:Bot med Micro:bit

NTNU
Trondheim

Institutt for fysikk

Skolelaboratoriet
for matematikk, naturfag og teknologi

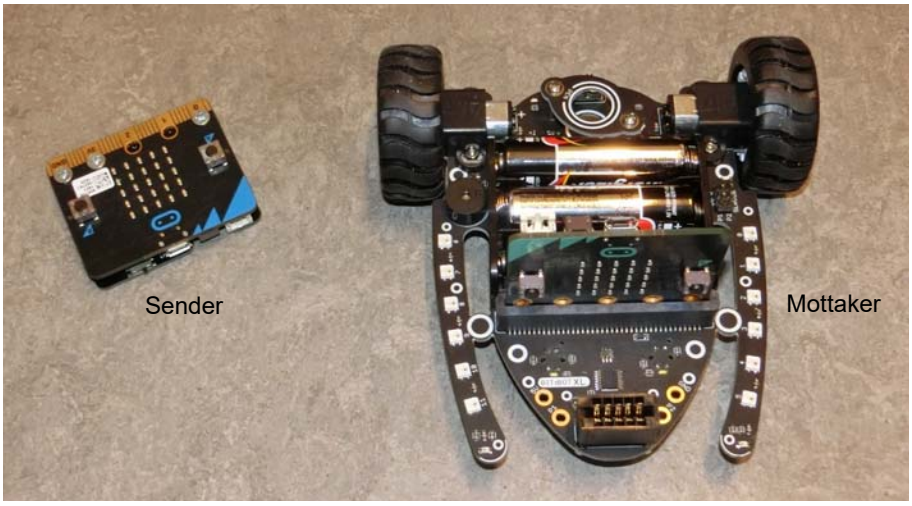
Mars 2020

Denne siden er blank

MICRO:BIT

Radiokommunikasjon –
Forslag til undervisningsopplegg

Nils Kr. Rossing



Micro:bit - Radiokommunikasjon – Forslag til undervisningsopplegg

Trondheim 2020

ISBN

Bidragsytere:
Nils Kr. Rossing, *(nkr@vitensenteret.com)* Skolelaboratoriet i Trondheim

Layout og redigering: Nils Kr. Rossing, Skolelaboratoriet i Trondheim

Tekst og bilder: Nils Kr. Rossing, Skolelaboratoriet i Trondheim

Faglige spørsmål rettes til:
Skolelaboratoriet ved NTNU
v/Nils Kr. Rossing, 73 55 11 91
nils.rossing@ntnu.no

Høgskoleringen 5
7491 Trondheim

Skolelaboratoriet ved NTNU
Telefon: 73 59 61 23
http://www.ntnu.no/Skolelab/

Rev 2.1 – 13.03.20

KreTek (Kreativ teknologi og samskaping på ungdomsstrinnet), et FoU-prosjekt støttet av Norges forskningsråd. Prosjektet er et samarbeid mellom Trondheim kommune og NTNU, hvor Skolelaboratoriet ved NTNU har prosjektledelsen.

Forord

Heftet er skrevet i forbindelse med et innslag på et kollokvium med pilotlærerne knyttet til KreTek-prosjektet 26. feb. 2020. De var alle kjent med micro:bit programmering, men ikke i særlig grad med med bruk av radioen. Det er derfor lagt vesentlig vekt på å formidle hvordan sender-mottaker delen av micro:bit fungerer inntil et visst nivå (se avsnitt 2.3).

Hensikten med kollokviet var både å få en vurdering av undervisningsopplegget og som et tilbud til opplæring av pilotlærerne. Det ble gitt mulighet til både å følge en oppskrift på papir med beskrivelse av oppbyggingen av programmet og forklaringer til valg av blokk-kommandoer, men også et opplegg som Roy Even Aune ved Vitensenteret har utarbeidet og som ligger på Wiki'en til Vitensenteret[1], der han utnytter muligheten til å gi skjulte tips eller avsløre løsningen på de ulike utfordringene. Det er således opp til brukeren å bruke tipsene eller la være. Begge ble vurdert, men denne gangen brukte deltagerne hovedsakelig papirkopien.

I tillegg ble radiodelen av micro:bit presentert omtrent slik den er omtalt i avsnitt 2.3, men i en noe forenklet form. Tilbakemeldingen var at de klarte å følge med og forsto det som ble presentert. Det var enighet om at det var fornuftig å gi elever flest en forenklet versjon, men at den interesserte elev burde kunne følge resonnementet i avsnitt 2.3. I en forenklet utgave kan det være lurt å fokusere på at man kan kommunisere trådløst parallelt innen forskjellige grupper uten at gruppene forstyrrer hverandre. Og at dette ikke handler om ulike kanaler med forskjellig frekvens, men bruk av ulike adresser som gjør at alle micro:bit'ene mottar alle meldinger, men forkaster de meldingene som ikke er adressert til dem.

Undervisningsopplegget prøves ut den 26. feb. 2020, 6. og 16. mars 2020, men er også bruket i en tid ved Vitensenteret i Trondheim.

En takk til Roy Even Aune (ViT) som har lest gjennom og kommentert deler av heftet.

Trondheim
Mars 2020

Nils Kr. Rossing
Skolelaboratoriet ved NTNU

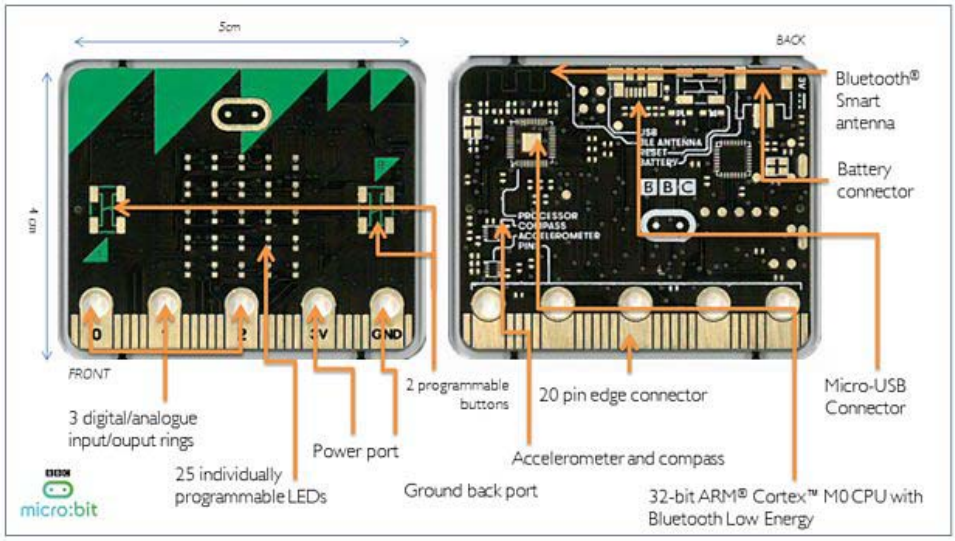
1. http://wiki.trigger.vitensenteret.com/doku.php?id=introduksjon_til_bitbot

Innhold

1 Innledning 7
1.1 Micro:bit og roboter 7
2 Litt teknisk småplukk om Micro:bit 9
2.1 Kantkontakten 9
2.2 Batteripakke for Micro:bit 11
2.3 Radioen i Micro:bit 11
2.3.1 nRF51822 – Nordic Semiconductor 12
2.3.2 Radioen kan operere på to forskjellige måter 12
2.3.3 Peer to peer kommunikasjon 13
2.3.4 Datapakkenes oppbygging 16
2.3.5 Modulasjon 17
2.3.6 Blokker som styrer radiokommunikasjon 17
2.3.7 Overføring av informasjon og styring via radio 20
3 Fjernstyring av Bit:Bot 21
3.1 Grunnleggende programmering av Bit:Bot 22
3.1.1 Programmering av sender-enheten 22
3.1.2 Programmering av mottaker-enheten 23
3.2 Tilleggsoppgaver 28
3.2.1 Lys og lyd hos Bit:Bot 30
4 Referanser 31

1 Innledning

Micro:bit er blitt en svært populær krets for å komme raskt i gang med programmering og å realisere ideer. Årsaken til dette er flere, men de viktigste grunnene er at kretsen er utstyrt med sensorer og et enkelt diodedisplay, den kan kommunisere trådløst med andre micro:bits og den kan programmeres med ulike programmeringsverktøy fra blokkprogrammering til Java og Python. Den kan også monteres i en sokkel (kantkontakt) slik at man kan få tilgang til mange av Micro:bits porter. Den er derfor lett å komme i gang med, men gir samtidig store muligheter.

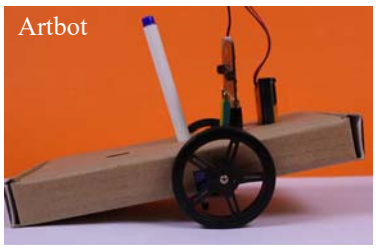


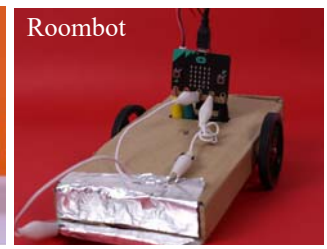
Kretsen har dessuten et 3 akse akselerometer og magnetometer slik at den kan detektere helningen til Micro:bit-kortet. Den kan dessuten fungere som kompass. Videre kan diode-displayet brukes som lysdetektorer.

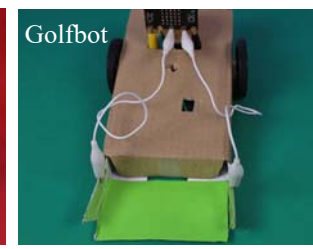
1.1 Micro:bit og roboter

På markedet finnes det en mengde roboter bygget omkring Micro:bit, ikke minst roboter laget av enkle materialer. Her er noen eksempler som er i salg.

Bildet under viser en roboter bygget opp av meget billige materialer.







Artbot kan holde en pen og tegne på et papir mens den beveger seg. Roombot har en kollisjons-

bryter foran laget av aluminiumsfolie som gjør den i stand til å detektere sammenstøt og på den måten snu og styre unna. Golfbot detekterer en ball som ruller inn i skuffen foran. Når ballen legger seg i skuffen sluttes en krets gjennom aluminiumsfolien[2]. Det fine med disse er at barna selv kan utforme og videreutvikle roboten selv med hjelpemidler de finner hjemme. Det forutsettes imidlertid at de har en Micro:bit. Pris for settet: ca. 20£.

Kitronik Micro:bit buggy

Kitronik leverer også en linjefølgende robot: ***Micro:bit buggy***[3]. Denne leveres om byggesett og krever litt lodding. Den er basert på et DC-motor driverkort med sokkel for Micro:bit-kortet. To DC-motorer med utveksling driver hjulene. Tre lyssensorer er festet foran på undersiden og fungerer som sensor for følging av en svart linje.

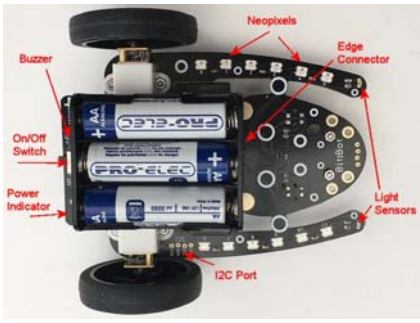
Pris € 32 Micro:bit-kort ikke inkludert.

4:tronix Robo:Bit Buggy[4]

har laget en interessant robotvariant med Micro:bit. Denne anvender akselerometeret for å detektere kollisjoner. Dessuten kan den styres trådløst via en annen Micro:bit. Med tillegg i prisen kan den utstyres med ultralyd avstandssensor og sensorer for å følge en linje.

Pris: £ 22,- hos Rapid electronics

4:tronix Bit:Bot[5]



Denne roboten er basert på standard DC-motorer med utveksling som forsynes med 3 x AA batterier. 12 lysdioder (Neopixel) er montert på armer på hver side av roboten, og to digitale reflektanssensorer for å følge en linje er montert under foran. En lydgiver (buzzer) kan programmeres til å gi lyd. Det følger dessuten med en innretning for montering av en penn. Roboten kan dessuten utstyres med flere sensorer. En ulempe er at pennen er plassert så langt bak, dette gjør det utfordrende å lage programmer for å tegne spesifikke tegninger. Dette er rettet opp i den neste utgaven bit:bot XL hvor pennen er plassert midt mellom hjulene. I tillegg er leveres den med en sokkel for montasje av en ultralyd avstandssensor. Pris bit:bot XL: 519,- + MVA.

Dette er et lite utvalg av roboter som finnes på markedet og som benytter Micro:bit.

Hva kan så vår robot tilføre dette mangfoldet?

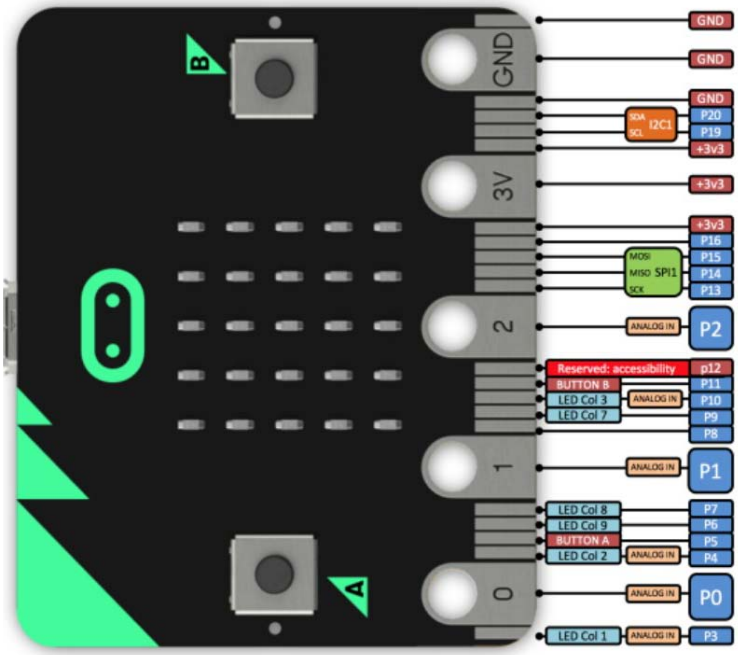
2. https://www.techwillsaveus.com/shop/microbit-microbot-pack/
3. https://www.stuff.tv/features/how-masterthe-bbc-microbit/master-1-microbit-crane?_format=amp
4. https://www.rapidonline.com/4tronix-robo-bit-buggy-for-bbc-micro-bit-75-0123
5. https://www.rapidonline.com/pdf/75-0117_v1.pdf
   https://www.rapidonline.com/4tronix-robo-bit-buggy-for-bbc-micro-bit-75-0123

2 Litt teknisk småplukk om Micro:bit

I dette kapittelet skal vi se nærmere på Micro:bit-kretsen og løfte fram noen aktuelle programmeringsblokker. Det er ikke tanken at dette skal være noen komplett oversikt over blokkoding av Micro:bit.

2.1 Kantkontakten

Ved tilkobling av kantkontakten til Micro:bit får man tilgang til et langt større utvalg av porter, både inn- og utganger og analoge og digitale porter. Figuren under viser en oversikt over kantkontakten.

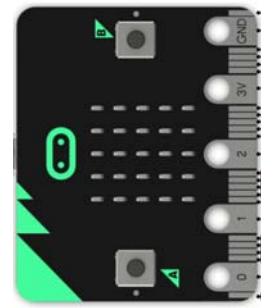


Som det framgår av figuren så har kretsen 20 porter hvorav 6 er analoge inn- eller utganger (P0, 1, 2, 3, 4 og 10), hvorav P0, 1 og 2 også kan brukes som berøringssensorer. De resterende 15 digitale portene (P5, 6, 7, 8, 9, 11, 12, 14, 15, 16, 17, 18, 19, 20) kan settes opp som inn- eller utganger etter behov så fremt de ikke er reservert til spesielle oppgaver som f.eks. P12 som er reservert til annet formål. Dessuten er P3, 4, 6, 7, 9, 10 brukt til å styre displayet, men kan frigjøres til andre formål etter behov. Vi legger også merke til at P3, 4 og 10, kombineres med analoge innganger. Hvilket betyr at lysdiodene også kan fungere som lyssensorer. Knappene A og B er koblet til P5 og P11.

I tillegg ser vi at drivspenningen er spesifisert til 3,3 V, men Micro:bit kan også kjøres på 3 V (dvs. ett CR2032 eller 2 x AA eller 2 AAA). Forsyningsspenningen er tilknyttet tre pinner. Dessuten er tre pinner koblet til jord. Selv om mange sensorer og *break out kort*[6] anvender 3,3 V, så er det også

mange som trenger en arbeidsspenning på 5 V. Den roboten vi skal bygge har en 6 V kilde som kan senkes til 3,3 V om nødvendig.

Pinne P19 (SCL) og P20 (SDA) kan også brukes til kommunikasjon via seriebuss (I^2C). Dette er en særdeles anvendelig seriekommunikasjonslinje for å kommunisere med andre digitale sensorer o.l. Det er også gitt mulighet for trelinje SPI-buss (P13, 14 og 15).



Jord (–)

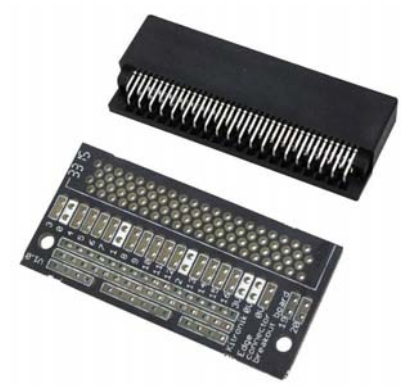
3V (+)

P2

P1

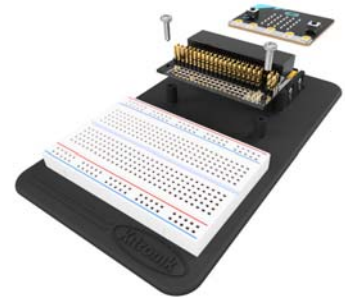
P0

Fem av tilkoblingspunktene er utvidet og gjennomhullet slik at det skal være mulig å koble til bananstikkere eller krokodilleklemmer. Dette gjelder batterispenningen 3V (+ og –) og portene P0, P1 og P2. Dette gjør at en kan komme igang svært fort med et minimum av tilleggsutstyr. Samtidig som det ligger et betydelig utviklingspotensial i kretsen.



Det finnes kantkontakter som passer til Micro:bit og som selges til en overkommelig pris (typ. £4 montert på kretskort og dobbel stiftlist). Figuren over til venstre viser en kantkontakt uten kretskort. Bildet til høyre viser en kantkort og kretskort som kan utstyret med dobbel stiftlist.

For den som ønsker å gå videre og utforske mulighetene til kretsen, kan en anskaffe et Inventors kit som gjør det lett å koble til eksterne komponenter på et koblingsbrett.



I tillegg til byggeplata inneholder settet jumpere og en rekke sensorer og aktuatorer, men selges uten Micro:bit så det må kjøpes separat. Pris i Norge ca. kr. 500,- inkl. mva. Mens Micro:bit koster ca. kr. 250,- inkl. mva.

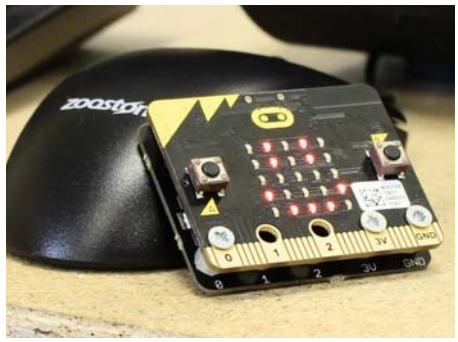
6. Små kretskort med sensorer eller andre elektriske komponenter, påmontert kontakter for tilkobling til andre elektroniske kretser

Nylig er det blitt utviklet et kort med kantkontakt av Joachim Hagen Skeie som driver Kodegenet AS. Kortet har fått navnet Micro:bit Servo:bot, og er spesielt laget for å kunne brukes for å bygge små roboter. Kortet kan fås som byggesett eller ferdig bygget.

Det er forberedt for å kunne koble til 4 servoer, avstandssensor og to Neopixler (fargete LED). Vi skal bruke kortet til vår robot.

2.2 Batteripakke for Micro:bit

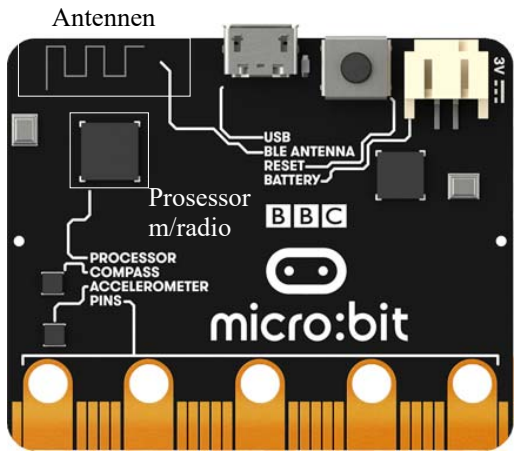
Dersom Micro:bit skal brukes uten å være tilkoblet PC, er det praktisk å montere den på et batteridekk (Power Back) som inneholder batteri (CR2032 - 3V), lydgiver (piezo elektrisk) og bryter. Dekket monteres til Micro:bit med tre maskinskruer med mutter og avstandsstykker.



2.3 Radioen i Micro:bit

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

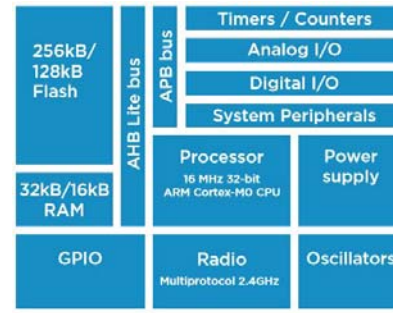
Radioen er uten tvil den enkeltegenskapen som gjør Micro:bit'en til et så kraftfullt mikrokontrollerkort. Radioen gjør det mulig å kommunisere trådløst mellom to mikro:bits eller grupper av mikro:bits, og til andre enheter som f.eks. en PC. Radioen omtales gjerne som en BLE hvilket betyr Bluetooth Low 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[7] 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.

2.3.1 nRF51822 – Nordic Semiconductor[8]



Blokkdiagram for nRF51822

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.

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.

2.3.2 Radioen kan operere på to forskjellige måter

Radioen kan operere på to forskjellige måter[9]:

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.

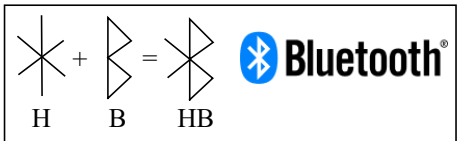
   Som all annen digital kommunikasjon overføres dataene i "pakker", dvs. et visst antall bit

7. Mikrobølgeovner opererer normalt på 2,45GHz Mikrobølgeovener med lekkasje av mikrobølger kan derfor lett forstyrre kommunikasjonen mellom micro:bits.
8. https://www.nordicsemi.com/-/media/Software-and-other-downloads/Product-Briefs/nRF51822-product-brief.pdf
9. https://www.littlebird.com.au/a/how-to/112/bluetooth-with-micro-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[10]-båndet er fra 2.4 – 2,5 GHz, en båndbredde på 100 MHz[11], 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"[12].



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.

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[13], men nok for kortdistansekommunikasjon (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[14].

2.3.3 Peer to peer kommunikasjon[15]

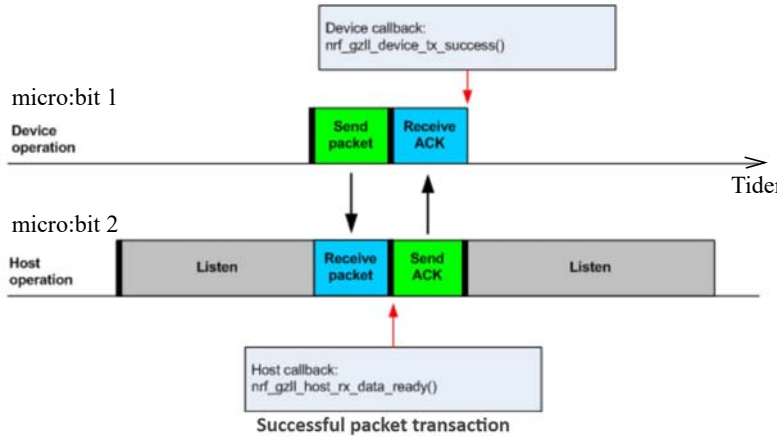
I dette avsnittet skal vi se nærmere på hvordan kommunikasjonen mellom to micro:bits foregår.

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

10. ISM - "Industrial, scientific and medical"
11. https://en.wikipedia.org/wiki/ISM_band
12. https://lancaster-university.github.io/microbit-docs/ble/profile/
13. En vanlig mobiltelefon kan sende på effekter opp til 2Watt, riktignok i kort pulser.
14. kbps - kilo bit pr. sekund. Mbps - Mega bit pr. sekund
15. 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 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.

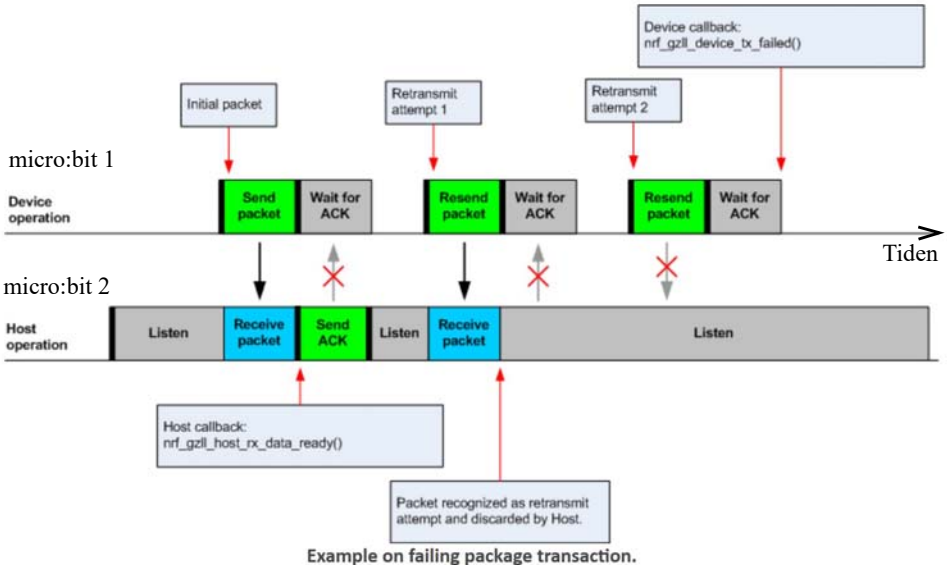
Figuren under viser hvordan en kan tenke seg at selve kommunikasjonen skjer.



Successful packet transaction

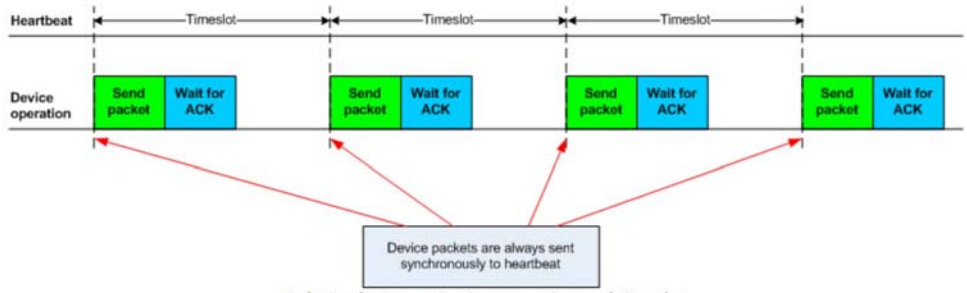
Vi tenker oss at micro:bit 1 (Device) "ønsker" å sende en melding til micro:bit 2 (Host). Micro:bit 1 sender en "pakke" med bit. Denne mottas av flere micro:bits deriblant micro:bit 2. Denne sjekker adressen (gruppe) for å se om pakken er ment for den. Når alle bitene i pakken er mottatt, sendes det 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.

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.



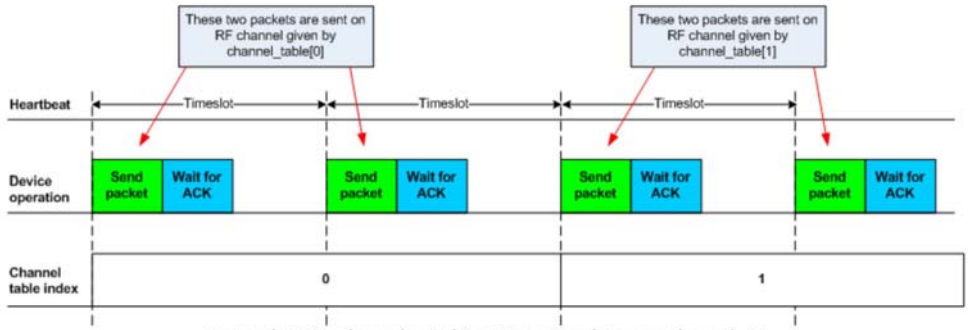
Relation between Device operation and timeslot

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[16]:

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

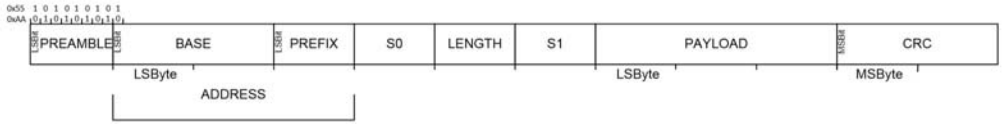
Når senderen skifter frekvens, så må også mottakeren skifte samtidig, ellers mister den pakken.

16.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

Både sender og mottaker må derfor bli enige om å bruke samme tabell over hvordan frekvenshoppingen 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 inneholder frekvensinformasjonen for å vite hvilke frekvenser de skal hoppe til, i tillegg til at frekvenshoppingen skjer synkront.

2.3.4 Datapakkenes oppbygging[17]

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



"Preamble" (innledning)

Dette er en sekvens av 01010101 (0x55) eller 10101010 (0xAA)[18]. 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.

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

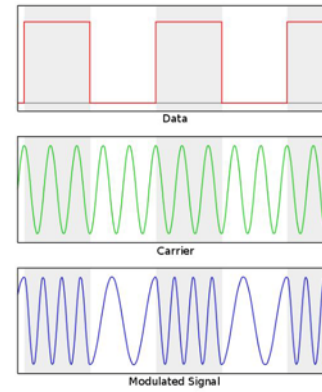
18.0xAA er tall uttrykt i det hexadesimale tallsystemet f.eks. 1010 1010 = AA hvor man benytter grunntall 16, tallrekken blir da 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Fire bit kan da uttrykkes med en bokstav 1010 = A. Dette skrives på formen (0xA)

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.

2.3.5 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*.

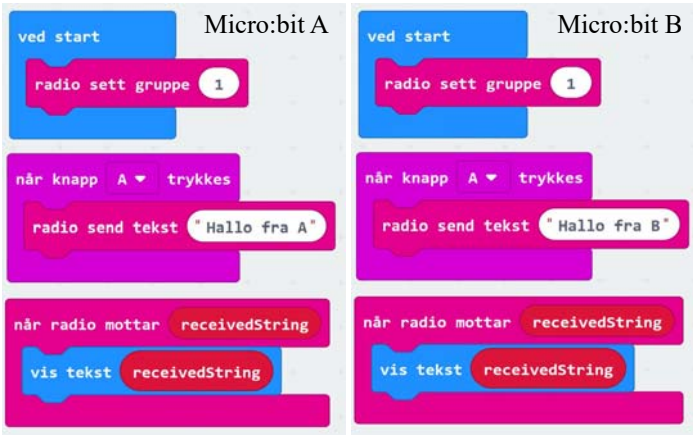
2.3.6 Blokker som styrer radiokommunikasjon[19]

Micro:bit er tilrettelagt for å sende og motta beskjeder:

- ... mellom to micro:bits
- ... fra en til en gruppe micro:bits
- ... og å skille mellom ulike grupper som skal kommunisere internt, men ikke bli forstyrret av kommunikasjon i andre grupper.

19. Mye av dette stoffet er hentet fra Halfacree Gareth, The Official BBC Micro:bit User Guide, Wiley 2018, kapittel 8

Å sette opp kommunikasjon mellom to micro:bits A og B (peer-to-peer)



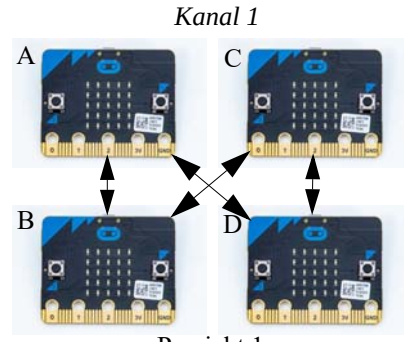
Figuren til venstre viser koden for hvordan man setter opp micro:bit A til å sende og motta innen gruppe 1: *ved start radio sett gruppe 1*. Dernest skal micro:bit A sende "Hallo fra A" når man trykker knapp A. Dersom micro:bit A mottar en tekst så skal den vises på displayet som en tekst som ruller forbi.

Micro:bit B programmeres på samme måte, men denne sender en litt annen tekst: "Hallo fra B" når det trykkes på knapp B.

Her programmeres begge micro:bits til å tilhøre gruppe 1. Det betyr både at de opererer med samme frekvensskjema (samme "kanal"), og at de opererer med samme adresse i adressefeltet i datapakken. I alt kan det settes opp 256 individuelle grupper (0 – 255) som kan operere uavhengig av hverandre til tross for at de opererer etter samme frekvenstabell. Dvs. at alle mottar alle meldinger, men at de siler ut de meldingene som ikke er merket med deres gruppeadresse. Dette kan bety at dersom mange sender samtidig så kan to meldinger forstyrre hverandre (kollidere) og må sendes på nytt.

Prosjekt 1: *Dørklokke og døråpner*

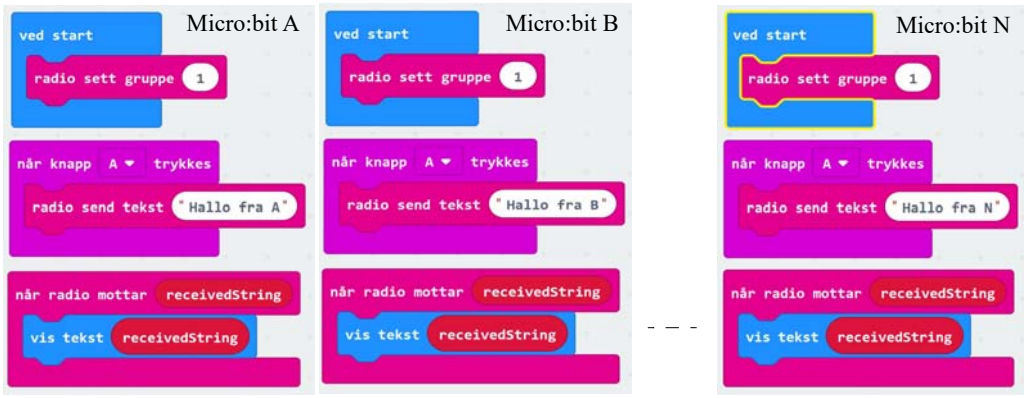
- To micro:bits kan brukes som en toveis dørklokke. Når det ringer på sendes det en beskjed opp til kontorpulten din som varsler om at det står noen ved døren og vil inn. Du kan da svare at du er på vei for å åpne.
- Den mest nærliggende utvidelsen er å inkludere en ringelyd slik at du blir oppmerksom på at det er noen som vil inn, uten at du må se på displayet.
- Dernest vil det være en ide å ikke bare gi en beskjed om at du kommer å åpner ved å trykke knapp A, men rett og slett låse opp døra ved å trykke på knapp A.



Prosjekt 1

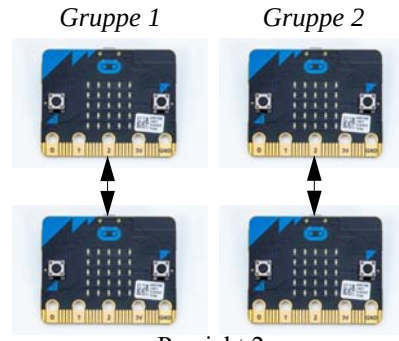
Å sette opp kommunikasjon mellom mange micro:bits (fra en til mange)

På samme måte kan man programmere flere til å operere i samme gruppe og dermed vil alle kunne kommunisere med mange som vist på figuren under.



I dette tilfellet vil det når knapp A trykkes hos micro:bit A gå ut melding til samtlige micro:bit som er koblet opp i nettverket. Det samme gjelder også for de andre. Dvs. at alle kan sende meldinger til alle.

Et slikt nettverk kalles *desentralisert* siden det ikke styres av *en sentral enhet*. Et slikt nettverk vil fungerer like godt til tross for at en eller ev. flere av micro:bitene er frakoblet. Eksempelvis kan A kommunisere med B selv om C er koblet fra. A kan kommunisere med C selv om B er koblet fra, og til sist B kan kommunisere med C selv om A er koblet fra.



Prosjekt 2

Prosjekt 2: *Dørklokke og døråpner til ulike beboere*

- *En kan også koble opp flere micro:bits som opererer innen ulike grupper. Dette kan være aktuelt dersom man har flere beboere i huset kanskje med en felles inngang. Da kan det være aktuelt med flere ringeknapper ved døra som kunne kommunisere med én bestemt beboer.*

Å sette opp gruppekommunikasjon

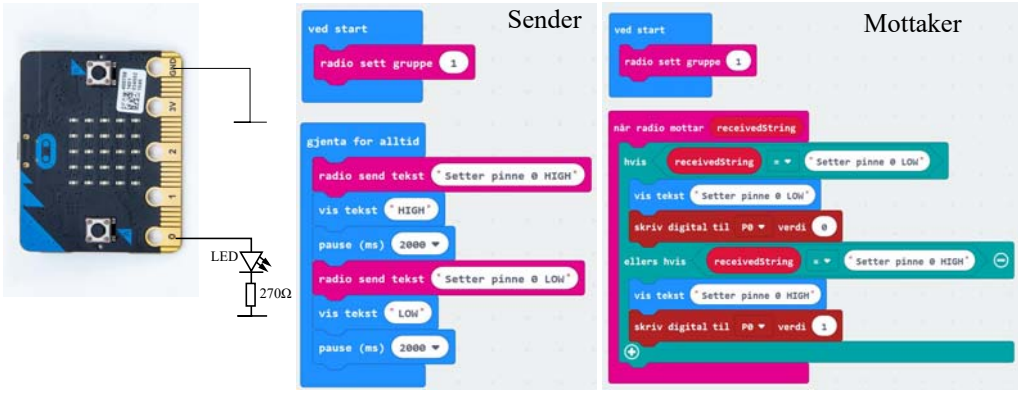
Dette gir mulighet for en micro:bit til å kommunisere med ulike grupper, og til og med skifte gruppe under veis om det er ønskelig:

Vi legger nå inn et skifte av gruppe når vi trykker på knapp B. For å teste ut denne egenskapen så trengs tre mikro:bits, A, B og C.

I utgangspunktet starter alle med å tilhøre gruppe 1. Når vi trykker på knapp A hos en av micro:bitene så mottar de to andre meldingen "Hallo fra A, B eller C". Dersom vi derimot trykker på knapp B, hos micro:bit A så vil det ikke skje noen ting, hos de to andre. B og C mottar ingen melding. Årsaken er at senderen hos A har skiftet til gruppe 2, mens de to andre, B og C, fortsatt befinner seg i gruppe 1. Trykker vi på knapp B hos micro:bit B, vil vi motta en melding hos A, men ikke hos C. Årsaken er at både A og B nå er i gruppe 2, mens C befinner seg forsatt i gruppe 1. Trykker vi på knapp B hos micro:bit C, så vil de to andre motta meldingen fra C. Nå er alle i samme gruppe igjen.

2.3.7 Overføring av informasjon og styring via radio

Det er mange måter å overføre informasjon mellom ulike micro:bits. I dette eksempelet skal vi se hvordan vi kan slå av og på en LED hos en micro:bit ved hjelp av en annen micro:bit.



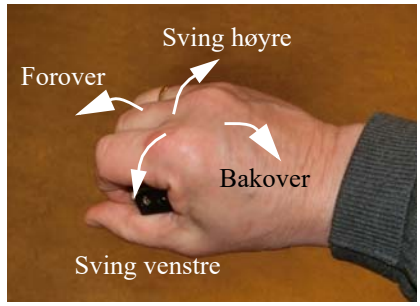
Koden i figuren over bruker en tekststreng til å overføre informasjon i tillegg til at den slår av og på en lysdiode. En enkel hvis-blokk (if-setning) sjekker om teksten sender en kommando om å slå på ("Setter pinne 0 HIGH") eller slå av lyset ("Setter pinne 0 LOW"). Vi har valgt å bruke de engelske ordene for høy og lav, siden vi i denne sammenhengen ikke kan bruke norske bokstaver.

3 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

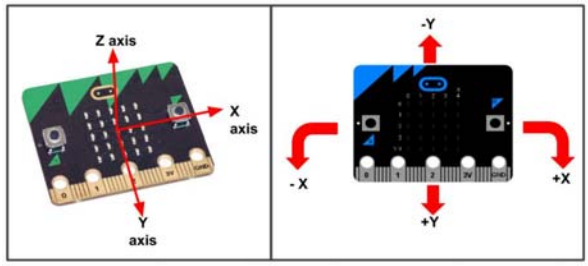
Vi ønsker å fjernstyre roboten ved hjelp av enkle håndbevegelser. For å utføre dette oppdraget trenger vi derfor to micro:bit'er, en batteripakke og en Bit:Bot XL. Den ene micro:bit'en, *sender-enheten*, bruker vi til å sende kommandoer til micro:bit'en som er montert på Bit:Bot'en, *mottaker-enheten*.



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



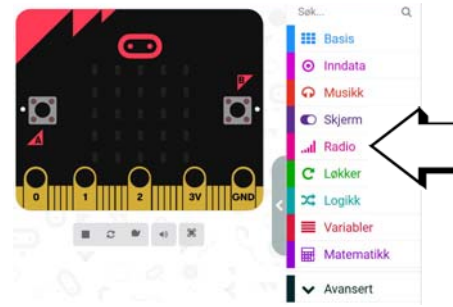
avleste verdien har benevningen mg (milli g):

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
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.

3.1 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)



3.1.1 Programmering av sender-enheten

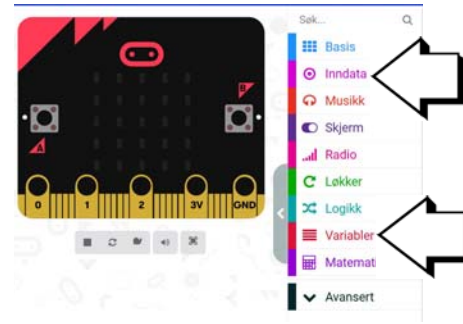
1. Velg radiokanal

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.

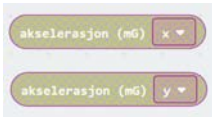
I vårt eksempel har vi valgt *gruppe 10*. For at vår sender-enhet skal kunne kommunisere med vår mottaker-enhet hos roboten, må også den settes til samme gruppe.

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).

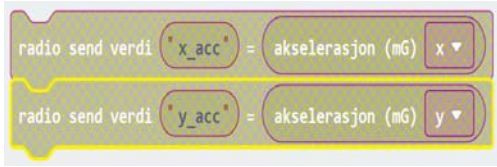


Siden vi skal lese av akselerometeret i både x- og y-retning, må vi gjøre to slike avlesninger, en for hver retning, x og y (se figuren under). Retning velges med den vesle pila til høyre i blokken.



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.

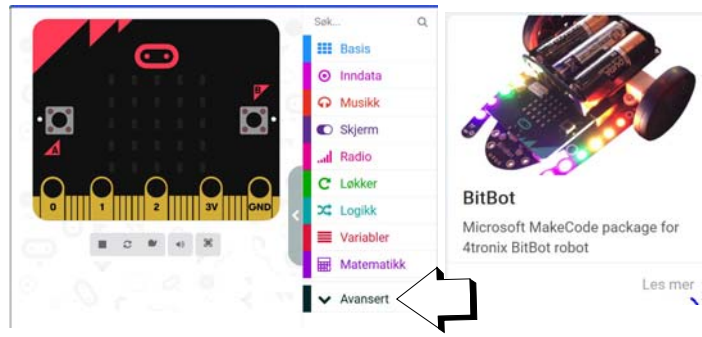
3.1.2 Programmering av mottaker-enheten

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

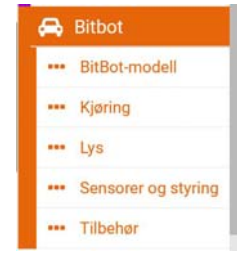
Pass på at du fjerner programmet som har med sender-enheten og skriver inn navnet på programmet til mottakeren. F.eks.: *Bit-bot Mottaker 1*

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:

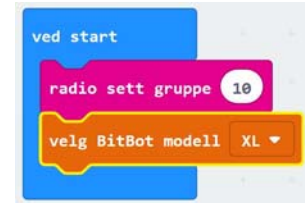


- *Velg menyen Avansert og deretter Utvidelser. Du kommer da til en meny hvor du finner en rekke utvidelser deriblant Bit:Bot. Velg Bit-Bot ved å trykke på rubrikken hvor Bit:Bot er avbildet (figur over til høyre).*
- *Det finnes to versjoner av Bit:Bot: Classic og XL. Sjekk hva slag Bit:Bot du har og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

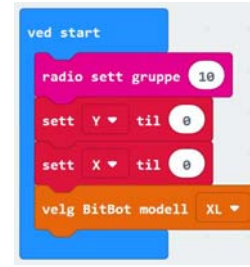
Hent fra funksjonen ved start fra Basis-menyen (blå). De blokkene som legges i funksjonsgapet til denne utføres bare når programmet startes opp.



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*.

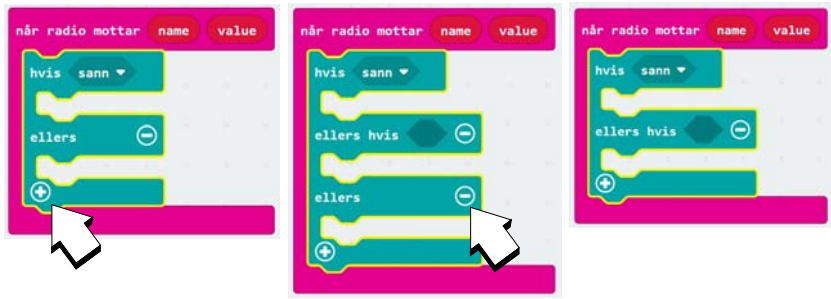
Ved oppstart velger vi å sette de variablene *X* og *Y* til 0 ved å bruke blokken *sett ... til ...* som vi finner i variabel-menyen. Blokken *ved start* kjører, som tidligere nevnt, bare når programmet starter opp.

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).

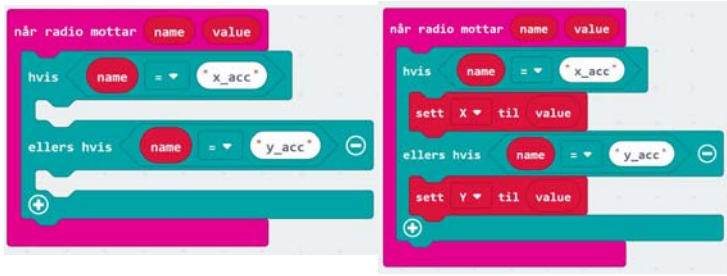


Hent *hvis-ellers-blokken* fra *Logikk-menyen* (grønn). Ved å trykke på + nederst i venstre hjørne av blokken, får vi opp et ekstra *ellers-hvis-felt* som vi ønsker i å bruke her. Vi ønsker imidlertid ikke *ellers-feltet* så denne fjerner vi ved å trykke – til høyre for *ellers*.

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).

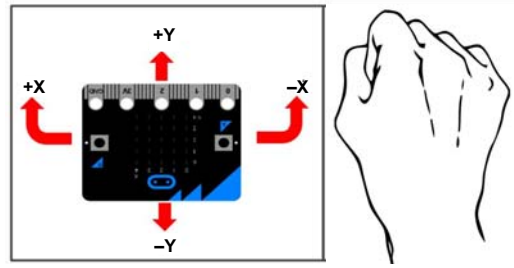
Så legger vi inn *name* på venstre side av betingelsene og skriver inn henholdsvis *x_acc* og *y_acc* mellom hermetegnene til høyre i betingelsen for å sjekke hvilken variabel som er oversendt. Variabelen *name* kan vi kopiere fra den ytre ramma ved kun å dra den over.



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*.

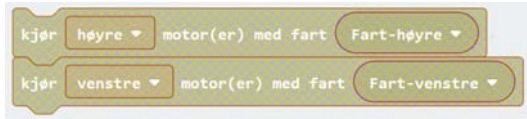
8. Styring av motorene



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:

$$\text{Fart-høyre} = Y \tag{3.1}$$

$$\text{Fart-venstre} = Y \tag{3.2}$$

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:

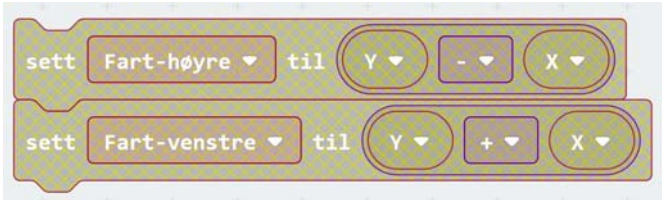
$$\text{Fart-høyre} = Y - X \tag{3.3}$$

$$\text{Fart-venstre} = Y + X \tag{3.4}$$

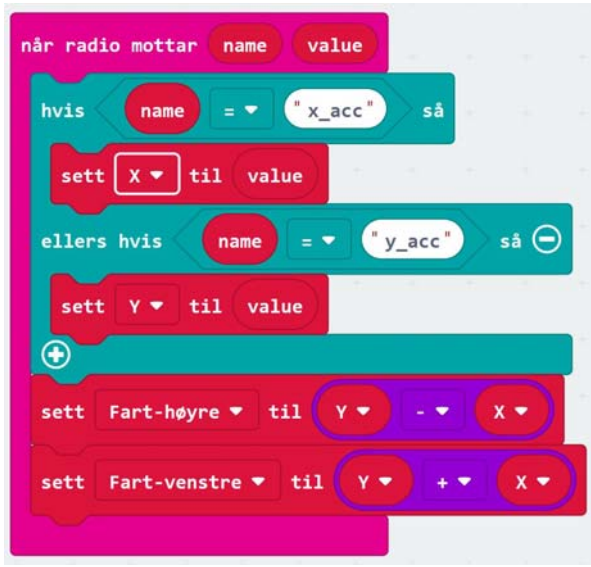
Forsøk å resonner dere fram til hvorfor vi har valgt fortegnene slik som vist. Om vi har valgt rett får dere svar på når dere tester programmet i roboten.

9. Beregn farten på hver av motorene

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



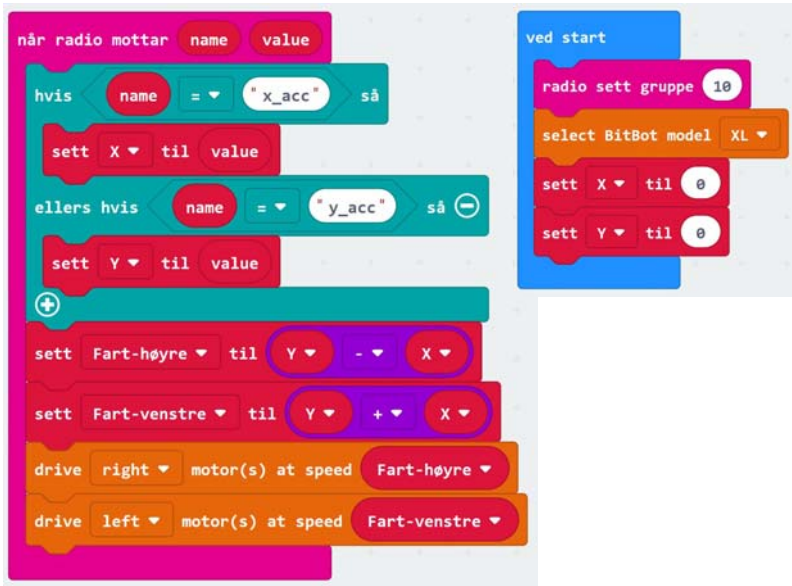
Om vi setter inn disse blokkene etter at verdiene er hentet fra radiomottakeren, vil programmet kunne se slik ut:



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[20].

12. Forenkling

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

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.

3.2 Tilleggsoppgaver

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

20. 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.

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.

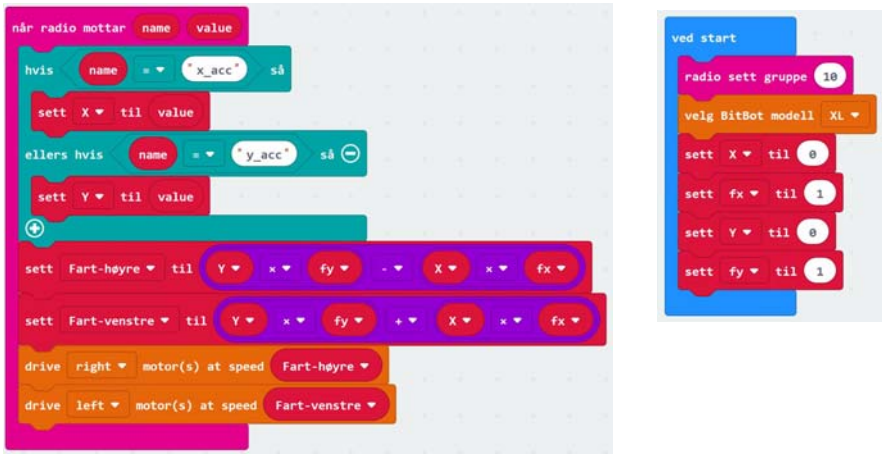
Økt følsomhet betyr at håndbevegelsen får større konsekvenser for farten, mens redusert følsomhet betyr at håndbevegelsen for mindre konsekvenser for farten.

$$\text{Fart-høyre} = Y*fy - X*fx \tag{3.5}$$

$$\text{Fart-venstre} = Y*fy + X*fx \tag{3.6}$$

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.

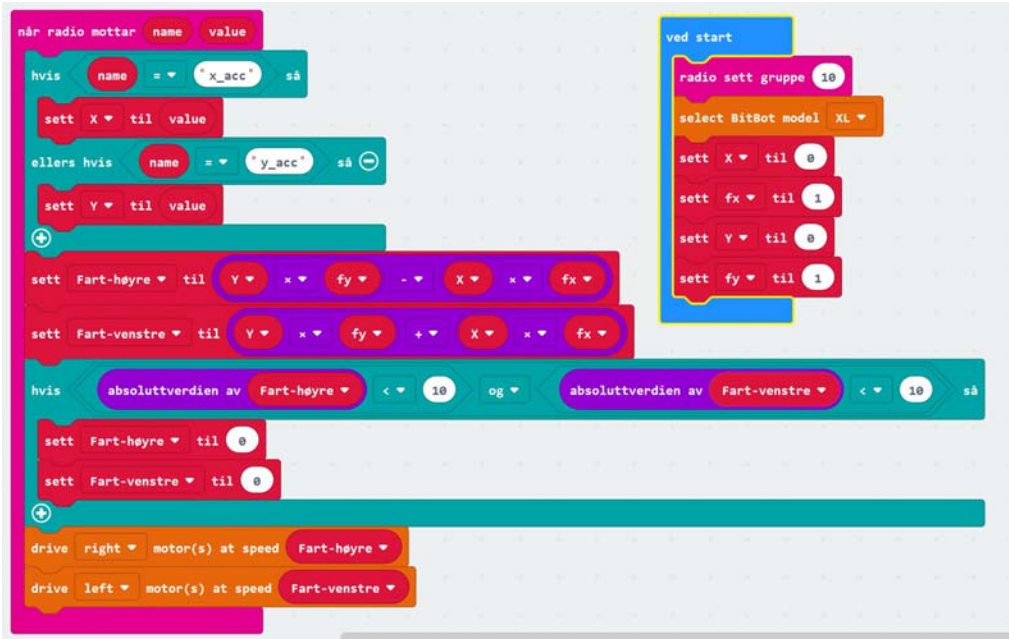


2. Sett opp et stoppvindu

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[21].

3.2.1 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.
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.

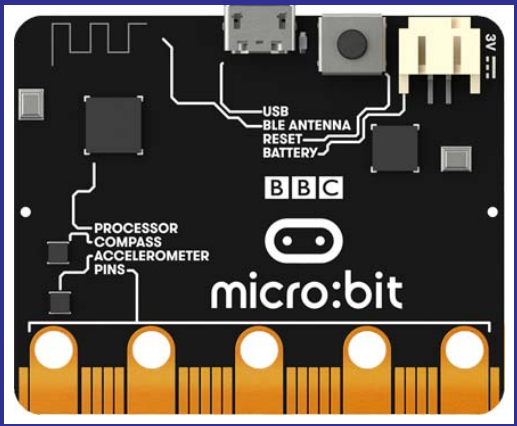
21. NB! Pass på at Bit:Bot løftes fra bordet når den slås på. Dersom den står på bordet vil lys-sensorene (reflektanssensorene) på undersiden av roboten vanligvis registrere mørke og gå inn i parringsmodus for bluetooth, hvilket vi ikke ønsker i denne omgangen. Det er mulig at dette kun er nødvendig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

4 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



Heftet beskriver radiodelen av BBCs Micro:bit og hvordan den kan brukes for å styre en Bit:Bot med en håndholdt Micro:bit og håndbevegelser.

Oppskriften er ment å være en introduksjon til temaet om sender og mottaker og hvordan trådløs kommunikasjon kan anvendes på en nyttig måte. Vi håper beskrivelsen kan inspirere til andre lignende prosjekter hvor man anvender radioen.

For flere oppgaver, se heftet: *Micro:bit – Forslag til undervisningsopplegg*.
https://www.ntnu.no/skolelab/bla-hefteserie

Nils Kr. Rossing
Prosjektleder ved Vitensenteret i Trondheim
Dosent ved Skolelaboratoriet ved NTNU
e-post: nils.rossing@ntnu.no

KreTek

KreTek (Kreativ teknologi og samskaping på ungdomsstrinnet)

Et FoU-prosjekt støttet av
Norges forskningsråd.

ISBN
Rev. 2.1 - 13.03.20