Kobling B

Programmering og skaperverksted for 8. - 10. klasse

Kapittel 1 – Introduksjon til programmering

Innhold

Kapittel 1

Analog introduksjon til programmering

Oppgave 1

Dere skal jobbe i par.

Begge bygger en liten figur av de utdelte legoklossene. Ta vare på figuren din helt til slutten, men ikke vis den til noen andre.

Lag en oppskrift slik at den andre du jobber med klarer å bygge en helt lik figur. Du får ikke forklare ekstra underveis, alt må stå skrevet ned.

Så bytter dere oppskrift og bygger hverandres figurer.

- Ble de like?
- Hva måtte du endre i oppskriften for at figurene skulle bli helt like?
- Kunne du laget oppskriften kortere eller enklere?

Oppgave 2

Dere skal jobbe i par.

Begge lager en oppskrift på hvordan den andre må gå for å komme seg ut døra. Det er ikke lov å forklare underveis.

Så bytter dere oppskrift, og en om gangen prøver ut den andres oppskrift og ser hvor du havner.

Etterpå justerer dere begge oppskriftene, men da kan dere bare bruke kodeklossene på neste side.

Hvor få klosser er mulig å bruke for å få til oppgaven?

Feilsøking og forbedring

1. Skjedde det som skulle skje? / Fikk du til oppgaven?
2. Hva måtte du endre på i oppskriften for å få til oppgaven?
3. Kunne du laget oppskriften kortere eller enklere?

Kodeklosser til analog programmering

Gå ett steg

Gå to steg

Gå tre steg

Disse klossene sier hvor mange steg du skal gå bortover.

Sving 90° mot høyre

Sving 90° mot venstre

Snu 180°

Disse klossene sier hvor mange grader du skal snu.

Stopp

Denne klossen sier at du skal stoppe opp og stå i ro.

Gjenta 5 ganger

Gjenta 10 ganger

Gjenta 20 ganger

Dette er løkker. De sier hvor mange ganger du skal gjenta noe. Hvor mange ganger du skal gå rundt gjennom løkka.

Dermed må du alltid koble den sammen en annen kloss som sier akkurat hva du skal gjenta i løkka di.

Hvis du krasjer, så

Hvis du sovner, så

Hvis du snubler, så

Dette er hvis-klosser. De inneholder en betingelse, altså HVIS noe skjer, SÅ skal du gjøre noe.

Dermed må du alltid koble den sammen en annen kloss som sier akkurat hva du skal gjøre i tilfelle at du krasjer, sovner eller snubler.

Introduksjon til viktige begreper

- vilkår, hvis-klosser og løkker

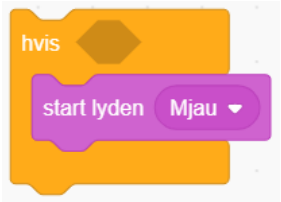
Vilkår
Et vilkår er en påstand som kan være sann eller usann. Vi sammenligner ofte to tall og ser hvilket som er størst, eller om de er like store.

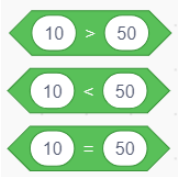
Eksempler på vilkår:
- Mamma er høyere enn pappa.
- Det er høyere temperatur om sommeren enn om vinteren.
- Læreren har lik skostørrelse som jeg har.

I Scratch er vilkårene grønne, sekskanta klosser.

Hvis-blokker

- Hva: En hvis-kloss inneholder en betingelse. Når betingelsen er sann, så vil det som står inni hvis-klossen skje.
- Hvorfor: En hvis-kloss gjør at et program kan gjøre flere forskjellige ting avhengig av om betingelsen er sann eller usann.





Diskuter
For hvilke av de tre grønne betingelsene vil vi høre mjauing med programmet til venstre?

Løkker

- Hva: Løkker gjør at en del av koden i et program blir gjentatt flere ganger. Det finnes flere typer løkker. Noen gjentar det du skal gjøre et bestemt antall ganger, mens andre løkker fortsetter evig. Den tredje typen gjentar frem til et vilkår ikke lenger er sant.

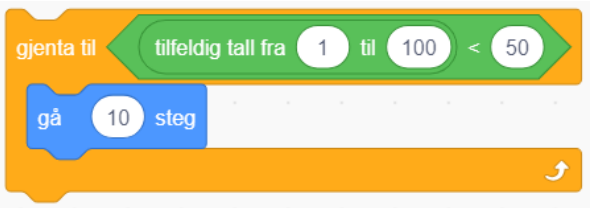
- Hvorfor: Programmet blir lettere å få oversikt over, og det blir færre klosser enn om man ikke bruker løkker.

Oppgaver

1. Lag to vilkår som er sanne og to som er usanne.
2. Er det alltid enkelt å vite om et vilkår er sant eller usant?
3. Hvilke type vilkår kan en datamaskin forstå?

Grubleoppgave

Hva tror du skjer i programmet under? Vil det samme skje hver gang?



Introduksjon til funksjoner i programmering
- hvordan lage egne klosser

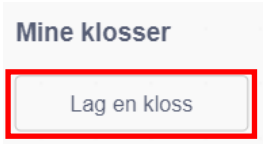
Hva? En funksjon er en kloss som du har laget selv. Altså et slags eget program. Det som skiller en funksjon fra et vanlig program, er at du først må definere en funksjon. Det vil si at du må lage den. Etterpå må du kjøre funksjonen i et program. Det som står inni en funksjon skjer ikke av seg selv før du bruker funksjonen i et program.

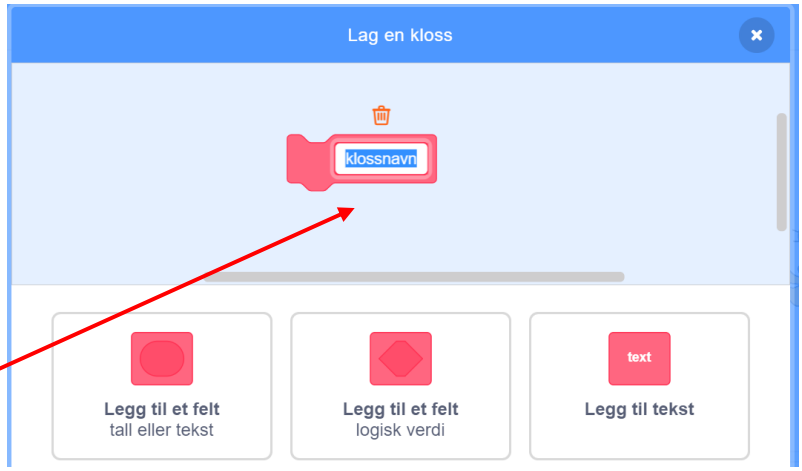
Hvorfor? Programmet du lager tar mindre plass, akkurat som når du bruker løkker. Funksjonen kan brukes flere steder i programmet uten å måtte lage alle blokkene på ny.

Hvordan?

1. steg: Velg mine klosser i menyen på venstre side.

2. steg: Trykk på lag en kloss-knappen som da dukker opp.





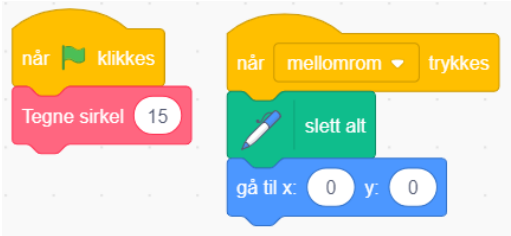
3. steg: Nå må du skrive inn hva funksjonen din skal hete der det står klossnavn.

4. steg: Du kan velge å legge til om funksjonen din trenger å få noe informasjon for å virke. Dette kalles ofte for inndata eller input. Det trenger vi for eksempel om funksjonen skal regne ut forskjellige regnestykker.

Dersom du lager en funksjon som skal regne om en verdi fra meter til centimeter, må funksjonen din få vite hvor mange meter du vil gjøre om til centimeter. Da må du legge til et tall når du oppretter klossen til funksjonen din.

Diskuter

1. Hva gjør funksjonen til venstre?
2. Hvorfor er det en funksjon, og ikke et eget program?
3. Hva gjør programmet under?
4. Hvorfor er dette et program?



Oppgave

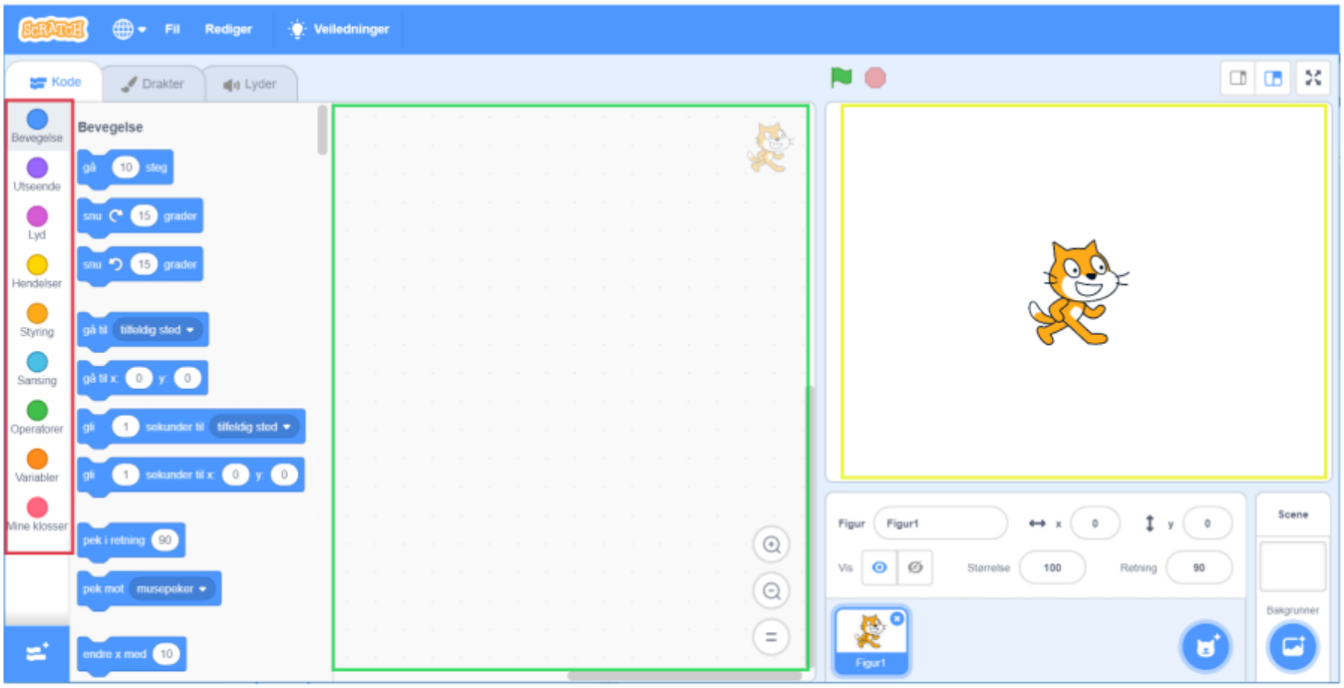
Lag en funksjon for at katten beveger seg i disse formene:

1. Kvadrat
2. Rektangel
3. Trekant

Kan du bruke funksjonene dine til å tegne en sammensatt figur?

Scratch
- en introduksjon

Gå inn på nettadressen: https://scratch.mit.edu



I den røde ramma er det ulike kategorier som gjør det enklere å finne de klossene man trenger.

Bevegelse: Her ligger klossene for hvor katten skal bevege seg.

Styring: Her ligger klossene som sier hvor mange ganger noe skal gjøres eller om det skal gjøres i det hele tatt.

Operatorer: Her ligger en del regneoperasjoner, som addisjon, subtraksjon og sammenligninger.

I den grønne ramma er feltet der man lager programmet.

Du trekker klossen du vil bruke inn hit, da er den en del av programmet ditt.

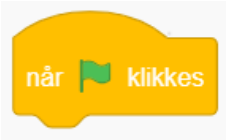
I den gule ramma ser du resultatet av programmet ditt.

Katten vil bevege seg slik programmet gir den beskjed om.

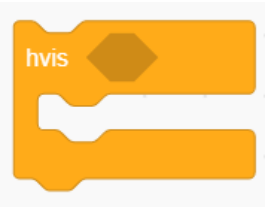
Dette området fungerer som et koordinatsystem, der katten starter i origo.

Intro til noen klosser i Scratch

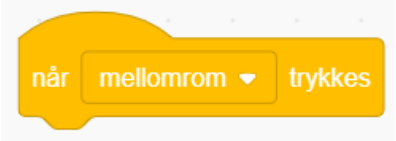
- Alle disse klossene finner du i kategorien som har lik farge som klossene



Når man klikker på det grønne flagget, skjer det som står i klossen koblet fast i denne.



Denne klossen kalles en hvis-kloss. Og hvis det som står i det grå feltet er sant, så vil det som står inni denne klossen skje.

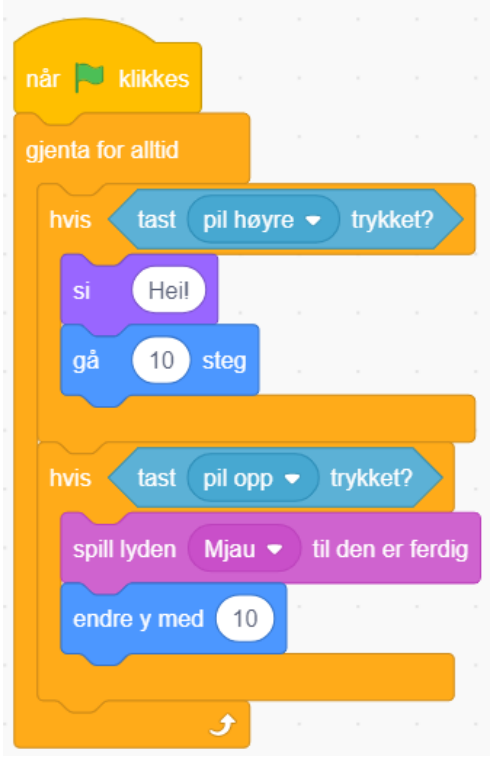


Når man klikker på mellomrom-tasten, skjer det som står i klossen som er festet i denne.

Eksempel på en betingelse som kan settes inn i det grå feltet til hvis-klossen. Dersom vi putter inn et tall som er større enn 50, vil det som står inne i hvis-klossen skje.

Denne klossen kalles en løkke. Det som står inne i denne klossen vil gjentas for alltid.

Denne klossen sier hvilke koordinater katten skal gå til.



Oppgave

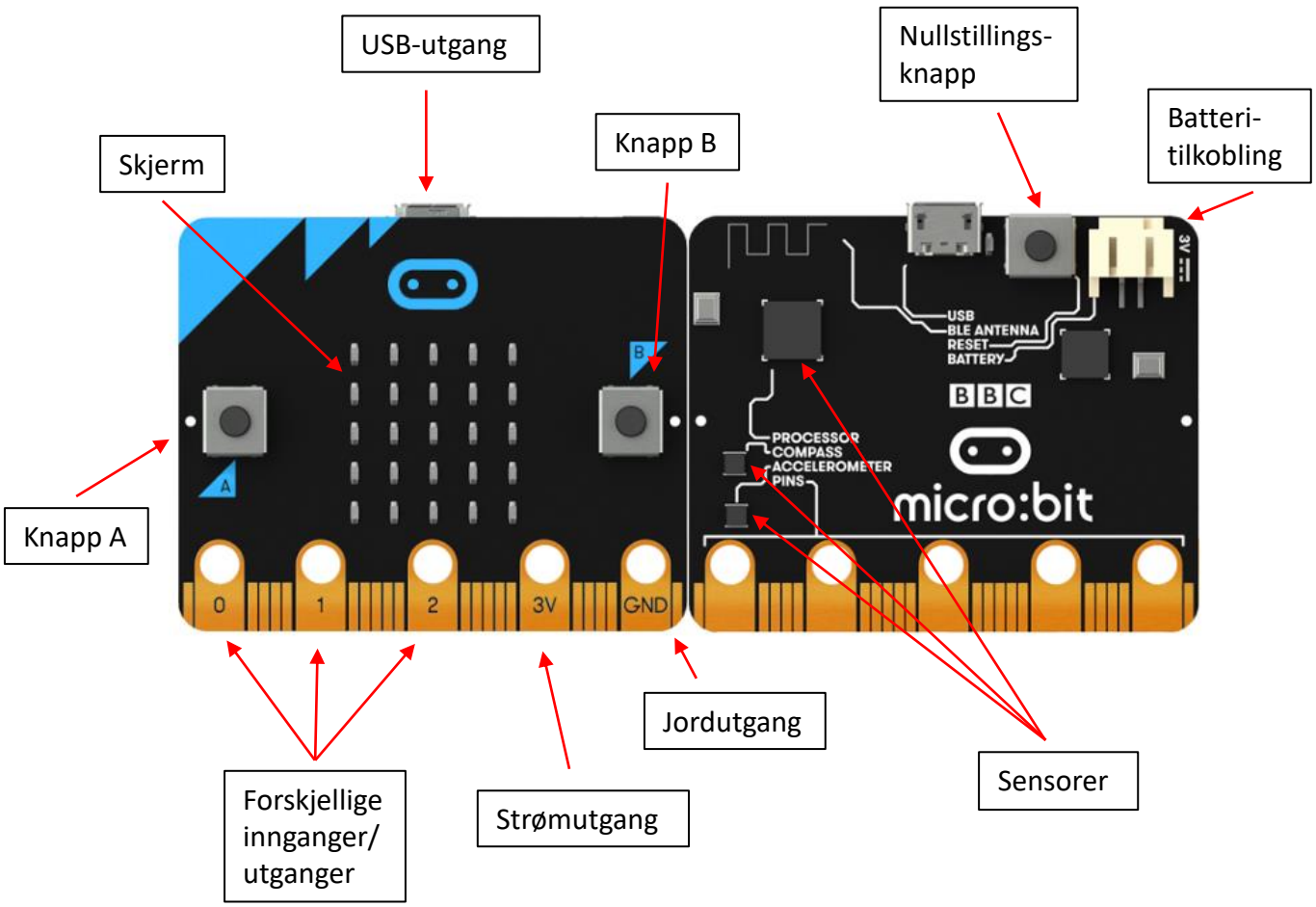
1. Gjett hva dette programmet gjør.
   - Fyll inn i det utdelte arket
2. Lag programmet i Scratch og se hva som skjer.
3. Hva må du gjøre for at katten skal kunne komme tilbake til origo?
   - HINT: Tenk på x- og y-koordinater.

Ekstraoppgave

1. Kan du få katten til å bevege seg i en rektangelform uten å bruke noen taster?
2. Hva med en trekantform?

Introduksjon til micro:bit

Micro:bit er en mikrokontroller, det vil si en enkel datamaskin som vi kan programmere. Vi kan programmere ved hjelp av en datamaskin, et nettbrett eller en mobiltelefon.



Skjerm: Micro:biten har en «skjerm» som består av 25 LED-lamper som vi kan programmere til å lyse, blinke og vise forskjellige bilder, tall eller bokstaver.

Knapper: Den har to knapper på framsiden, og en nullstillingsknapp på baksiden. Nullstillingsknappen gjør at programmet på micro:biten begynner på ny.

Mini-usb: Her kan du koble micro:bit til en USB-kabel dersom du bruker en datamaskin med USB-utgang.

Batteritilkobling: Her kan du koble til batterier for at micro:biten skal få strøm.

Sensorer: Micro:biten har noen innebygde sensorer som kan måle akselerasjon, temperatur og kompassretninger.

Tilkoblinger: Det går an å koble på andre sensorer, motorer, LED-pærer eller høyttalere på micro:biten ved hjelp av alligatorledninger.

Bluetooth (blåtann): Micro:biten har bluetooth-tilkobling (blåtann-tilkobling) slik at den kan få overført programmer fra for eksempel iPad som ikke har USB-utgang.

Introduksjon til bruk av micro:bit

- for datamaskiner

Gå inn på nettadressen: https://makecode.microbit.org/#editor

Da kommer du inn på denne nettsiden hvor du kan begynne å programmere micro:biten.

Simuleringsområde

I den gule ramma ser du resultatet av programmet ditt.

Klosseliste

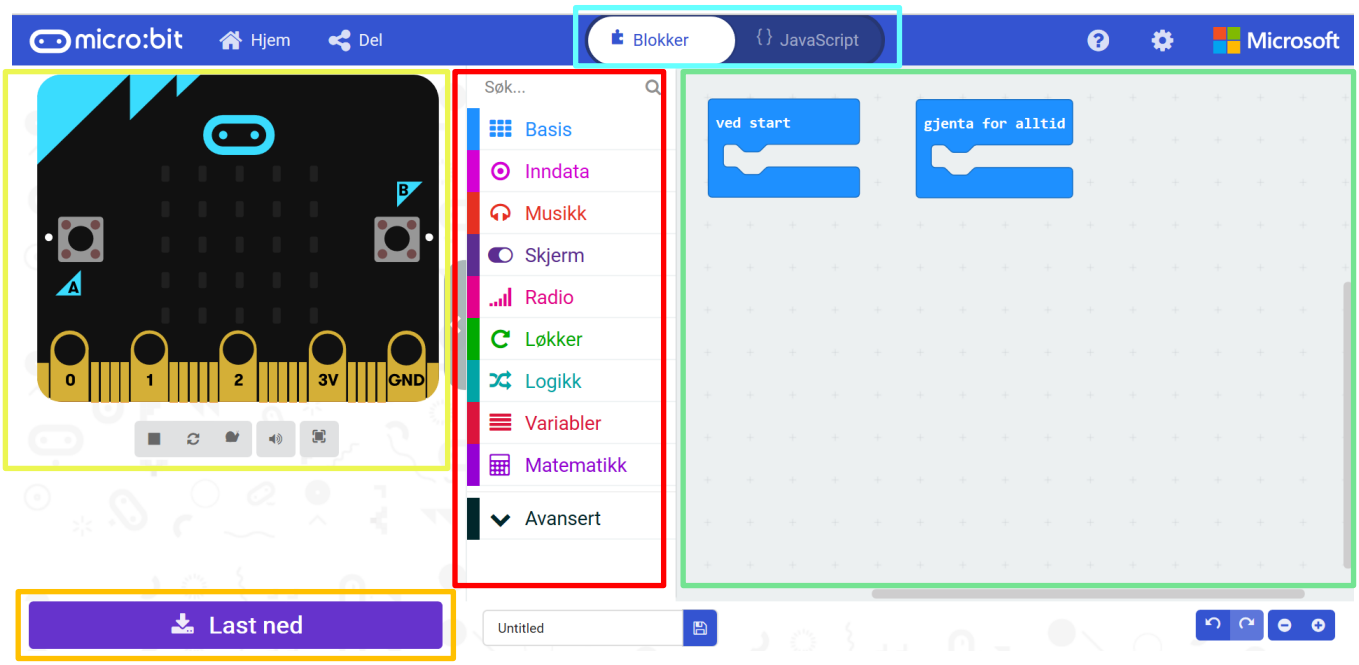
I den røde ramma er det ulike kategorier som gjør det enklere å finne de klossene man trenger.

Når du trykker på en av kategoriene, vil klossene som hører til komme fram rett til høyre.

Programmeringsområde

I den grønne ramma er feltet der man lager programmet.

Du trekker klossen du vil bruke inn hit, da er den en del av programmet ditt.



Nedlastingsområde

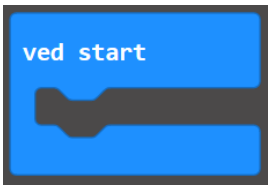
I den oransje ramma er kan vi laste programmet over til micro:biten.

Programmeringsspråk

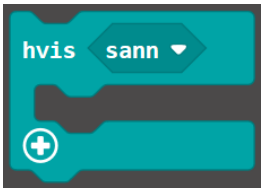
I den turkise ramma er feltet der du kan velge om du skal bruke klosser å programmere med, eller et tekstbasert programmeringsspråk kalt Python eller JavaScript.

Intro til MakeCode for micro:bit

- Alle disse klossene finner du i kategorien som har lik farge som klossene



Straks programmet er overført til micro:biten skjer det som står inni denne klossen.



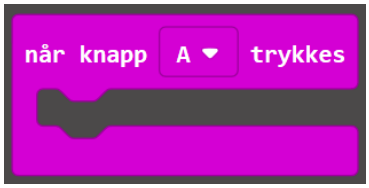
Denne klossen kalles en hvis-kloss. Og hvis det som står i det mørkegrønne feltet er sant, så vil det som står inni denne klossen skje.

Denne klossen kalles en løkke. Det som står inne i denne klossen vil gjentas for alltid.

Eksempler på vilkår som kan settes inn i det mørkegrønne feltet i hvis-klossen. I det øverste vilkåret kan vi sette inn to tall som den sjekker om er like. I den nederste kan vi sette inn to tall der vilkåret sjekker om det første er mindre enn det andre.

Denne klossen gjør at programmet tar en pause før den går videre til neste kloss.

Når verdien er 1, vil denne klossen gjøre at det sendes strøm ut gjennom P0-utgangen. Når verdien er 0, sendes det ikke strøm ut gjennom utgangen.

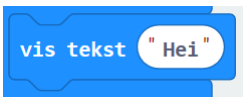


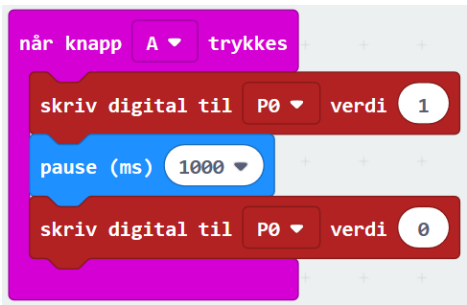
Denne klossen utfører det som står inni den når man trykker på knapp A på micro:biten.

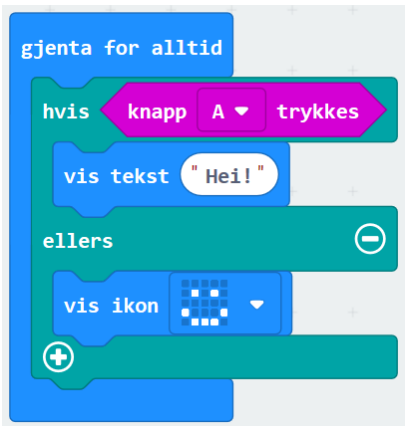
Output til skjerm

Et viktig poeng med å bruke micro:bit er at vi kan vise informasjon på den. Det kalles gjerne utdata eller output, og kan vises på skjermen til micro:biten. Skjermen er ikke en vanlig skjerm, men består av 25 røde lysdioder som kan være av- eller påskrudd.

I micro:biten har vi ulike klosser som kan vise ting på skjermen. Vi kan vise tekster, bilder eller tall. Tekstene kan være litt vanskelige å lese, for de ruller over skjermen og består av en blanding av store og små bokstaver. Vi kommer til å vise flest tall i denne boka.







Oppgave

Gjett hva de to programmene gjør.

Lag programmene i makecode og overfør til micro:biten – ett om gangen.

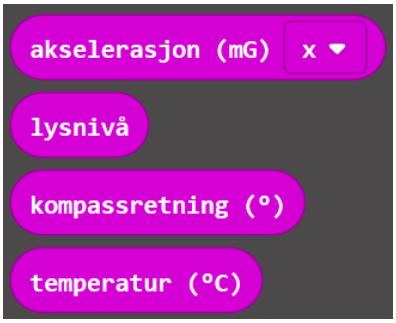
Skjedde det du trodde?

Kan du gjøre noen morsomme endringer?

Inndata/input for micro:bit

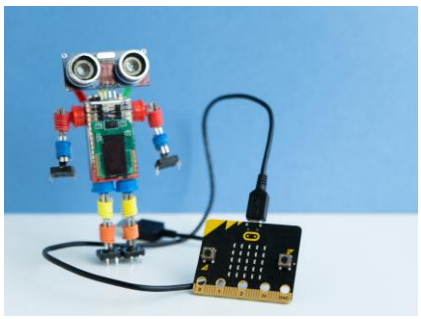
Input fra sensorer

Mesteparten av poenget med å bruke micro:bit er at vi kan samle inn informasjon med den. Det kalles gjerne inndata eller input, og kommer fra forskjellige sensorer. De kan være innebygd i selve micro:biten, eller vi kan koble til andre sensorer.



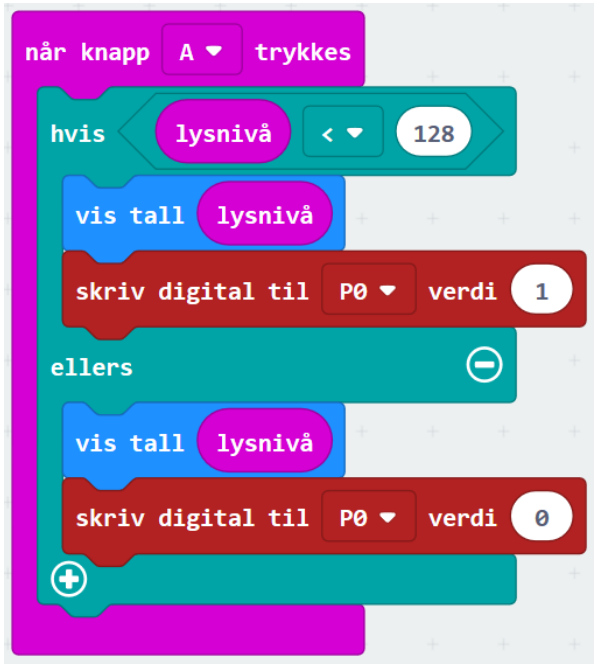
I micro:biten har vi innebygde sensorer som kan måle akselerasjonen, lysnivået, kompassretningen og temperaturen. Vi kommer oftest til å bruke temperaturen i denne boka.

Dersom vi ønsker å bruke en sensor som ikke er innebygget i micro:biten, må vi først koble den til en inngang på micro:biten. I tillegg trenger vi en kloss som leser av hvilken verdi sensoren sender til micro:biten. Da bruker vi denne klossen. Den leser bare av hva verdien er, men gjør ingenting med den. De fleste sensorer vi kobler på gir verdier mellom 0 og 1023, men det kan variere fra sensor til sensor. Det står mer om sensorer og hvordan de virker på neste side.



Snakk om

1. Hvilke sensorer vet du om?
2. Hvordan kan en micro:bit vise fram målingene fra sensorene?
3. Hva kan man bruke forskjellige sensorer til?
4. Kan du tenke ut en oppfinnelse som bruker en sensor for å løse en oppgave.



Oppgave

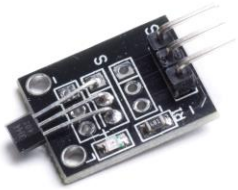
1. Gjett hva dette programmet gjør.
2. Lag programmet i MakeCode, overfør det til micro:biten og se hva som skjer.
3. Hva kan det tenkes at skal være koblet til micro:biten?
4. Hva ville du gjort med programmet for å forbedre det?

Sensorer

En sensor blir brukt til å måle forskjellige verdier, slik som temperatur, lysstyrke, fart, akselerasjon eller lydnivå. Sensoren kobles til en micro:bit og man kan måle hvor stor spenning som kreves for å trykke elektronene gjennom den. Da er det slik at i en temperatursensor vil spenningen variere i samme takt som temperaturen, og vi kan bruke spenningen vi måler til å regne ut nøyaktig temperatur.

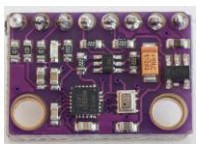
Termistor
- måler temperatur

Lyssensor
- måler lysstyrke

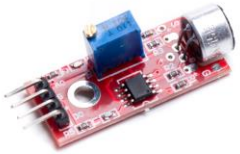


Halleffekt-sensor
- måler magnetfelt

Ultralydsensor
- måler avstand



Akselerometer
- måler akselerasjon



Lydsensor
- Måler lydstyrke

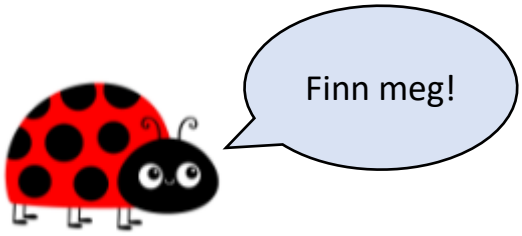
Hva er motstand?

Når et materiale har stor motstand, er det vanskelig for elektronene å bevege seg gjennom materialet. Da trenger man høy spenning for å dytte elektronene gjennom. Motstand blir også kalt resistans, og enheten er ohm.

Elektriske ledere har liten motstand slik at elektronene kan bevege seg lett gjennom dem, mens isolatorer har veldig stor motstand slik at det er nærmest umulig for elektroner å bevege seg gjennom dem.

I sensorer varierer motstanden, og da vil også spenningen variere. Denne spenningen er enkel å måle med en micro:bit. I en lyssensor vil motstanden variere med lysstyrken, og vi får målt forskjellig spenning om det er mørkt eller lyst. For en ultralydsensor vil spenningen varierer ut fra hvor stor avstand sensoren har til en gjenstand, og tilsvarende for andre sensorer.

Feilsøking
- debugging



Absolutt ingen som programmerer gjør alt riktig på første forsøk. Ikke engang de aller beste programmererne som har jobbet med programmering i mange år! Når programmet ikke oppfører seg som planlagt, trenger vi å finne ut hva som er årsaken til det. Vi må finne feilene i koden vår, også kalt bugs. Det kan være lurt å lage seg en egen liste med vanlige feil for deg, og bruk denne lista for å finne bugs i nye programmer.

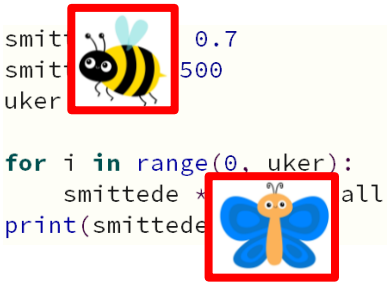
Eksempel på en sjekkliste for feilsøking

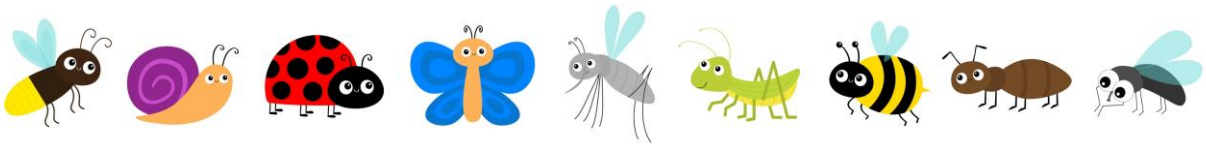
- gjør gjerne det som er enklest først, ikke slett hele programmet

1. Kjør programmet og se hva som skjer
    - Hva er det som ikke stemmer med det du ønsker at programmet skal gjøre?
    - Skjer ting i den rekkefølgen du hadde planlagt?
    - Får du noen feilmeldinger (tekst-programmering)?

2. Gjør synlig det som skjer inni programmet og kjør det
    - Skrive variablene til skjerm
    - Legge inn pauser i koden for å rekke å se hvilke verdier forskjellige variabler har

3. Endre på programmet og kjør det
    - Har variablene riktig navn med stor/liten bokstav alle steder du bruker samme variabel?
    - Har løkkene riktig antall repetisjoner?
    - Endre bare på en ting om gangen, slik at det er enkelt å se hva en forandring fører til
    - Gir feilmeldinger noen hint om i hvilke linjer feilen ligger (tekst-programmering)?
    - Gir feilmeldinger noen hint om hva feilen er (tekst-programmering)?
        - Feil variabeltype
        - Glemt innrykk
        - Manglende kolon eller andre tegn
    - Søk på feilkoder på internett (tekst-programmering)



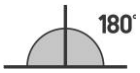


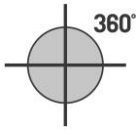
Søke etter svar

- Selv de flinkeste programmererne bruker mye av tiden på å søke opp feilkoder på internett. Det kan også være lurt å søke på navnet til forskjellige funksjoner for å se hvilke inndata de skal ha, og hvilke utdata de gir.
- Står det noe i læreboks di? Les gjerne nærmere gjennom oppgaven.
- Spør noen i klassen eller læreren.

Hvordan virker en servo?

En servo er en liten elektrisk motor som kan dreie frem og tilbake eller rundt og rundt. Den som bare kan dreie frem og tilbake kalles en 180°-servo. Den som kan dreie rundt og rundt kalles en 360°-servo (eller kontinuerlig servo).





Vinkler

Servoene får navn fra vinkler. En vinkel kan være mellom 0° og 360°. En vinkel på 90° kalles en rett vinkel og ser slik ut.

En vinkel på 180° er det dobbelte av en rett vinkel, og da får vi faktisk en rett linje!

Hvis du tar fire rette vinkler, ender du opp med en vinkel på 360°. Da har du kommet helt rundt! De servoene som kalles 360°-servoer går jo rundt og rundt.

Hva betyr kontinuerlig?
Noe som fortsetter uten opphold.

En 180°-servo kan bare dreie 180°, og ikke rundt og rundt.

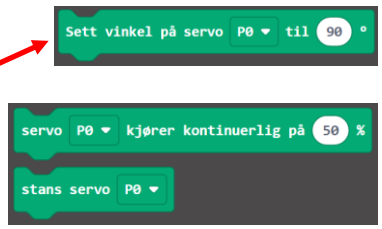
Det går typisk ikke an å styre farten til denne typen servoer, bare vinkelen de dreier til.

Programmering av servoer

Da kommer du til å trenge noen av disse klossene.

180 °-graders servo:
Denne setter hvilken vinkel servoen skal dreie til.

360°-graders servo:
Denne setter farten til servoen.
Denne stopper servoen.

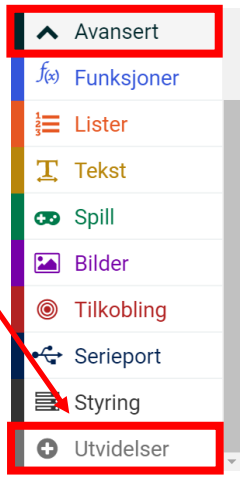


Diskuter

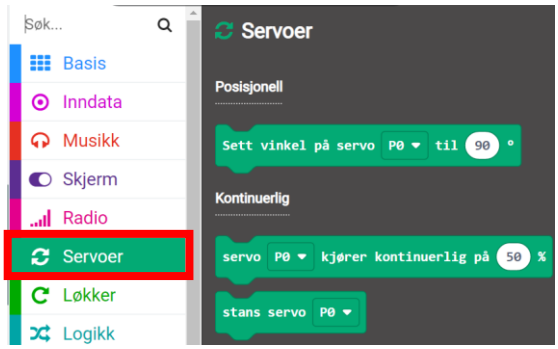
Hva må du gjøre for å få 360°-servoen til å dreie i motsatt retning?

Disse klossene finner du slik

1. Velg avansert i menyen, den er svart.
2. Trykk der det står utvidelser, nederst.
3. Så velger du servo-utvidelsen.



4. Velg servo i menyen, så finner du klossene du trenger.



Servoene trenger litt mer strøm enn det som kommer fra micro:biten. For å være sikker på at den får nok strøm til å virke slik den skal, må man bruke et kretskort som gjør at vi kan koble til batterier i tillegg til micro:biten.

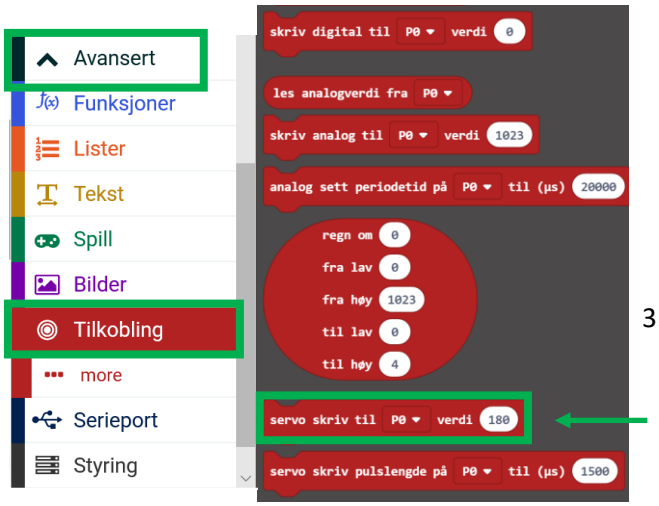
Dere kan la være å legge til servo-tillegget, da må dere bruke en annen kloss enn vist på forrige side. Det er bare en kloss uavhengig av hvilken type servo man bruker, men verdiene man skriver inn får ulikt resultat ut fra servotypen.

180°-servo: Verdien angir vinkelen servoen skal dreie til.

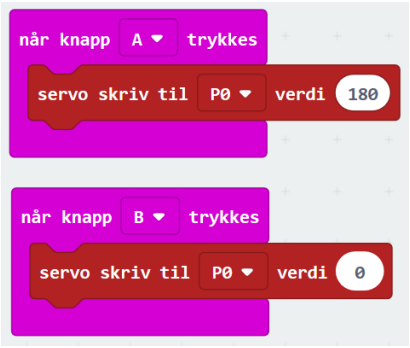
360°-servo: Verdien angir dreiehastighet som servoen dreier rundt og rundt med.

Denne klossen finner du slik

1. Velg avansert i menyen, den er svart.
2. Gå nedover i menyen til du finner tilkobling. Den er mørkerød.
3. Så finner du klossen du trenger nest nederst (og nederst).



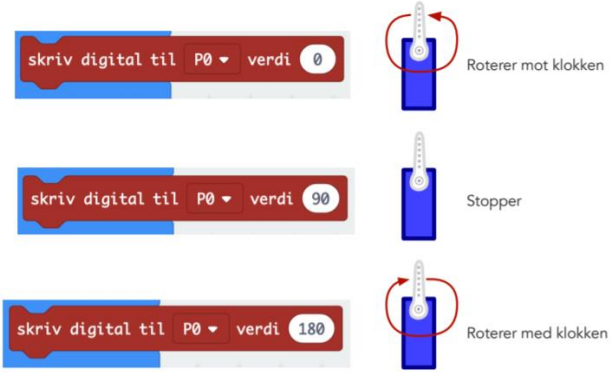
Programmering av 180°-servoer



En 180°-servo kan dreie 180° hver vei, men ikke dreie helt rundt. Man kan typisk ikke styre hastigheten til denne typen servoer, bare hvilken vinkel de dreier til.

Det øverste programmet gjør at servoen dreier 180° i den ene retningen. Det nederste programmet gjør at servoen dreier tilbake til utgangspunktet.

Programmering av 360°-servoer (kontinuerlige servoer)



Om vi skriver verdien 0 til servoen, vil den rotere mot klokken så fort den klarer.

Når vi øker verdien, vil den rotere saktere helt til vi kommer til 90 hvor motoren står stille.

Når vi øker verdien over 90 opp mot 180 grader roterer den raskere og raskere med klokken.

Her er verdien altså ikke vinkelen, men dreiehastighet.

Miniutfordringer

Det skjeve tårn i Pisa

Materialer: Papplate (30 cm x 30 cm), 18 papirark, tape og saks.

Tidsbruk: Planlegging 2 min, gjennomføring 10 min.

Krav:

Bruk de utdelte materialene.

Bygg tårn på papplata med gjennomsnittshøyde på minst 160 cm.

Klossene må være i kontakt med plata og plata skal være intakt.

Utskytningsmekanisme

Materialer: Alt som ligger klart.

Tidsbruk: Planlegging 2 min, gjennomføring 10 min.

Krav:

Lag en utskytningmekanisme som får svampen til å fly minst 5 meter.

Utskytningsmeknismen må bestå av 5 ulike materialer.

Uidentifisert flygende objekt (UFO)

Materialer: Alt som ligger klart.

Tidsbruk: Planlegging 2 min, gjennomføring 10 min.

Krav:

Lag et flygende objekt som skal være i luften lengst mulig.

Bruk de utdelte materialene.

Det flygende objektet skal dra en blyant.

Bakkeløp

Materialer: Alt som ligger klart.

Tidsbruk: Planlegging 2 min, gjennomføring 10 min.

Krav:

Lage en gjenstand som bruker nøyaktig 5 sekunder på å trille ned papplatebakken.

Bruk de utdelte materialene.

Det er ikke lov å berøre gjenstanden etter at den er sluppet.

Brobygging

Materialer: Alt som ligger klart.

Tidsbruk: Planlegging 2 min, gjennomføring 10 min.

Krav:

Bruk de utdelte materialene.

Bygg en bro med et spenn på minst 50 cm.

Broa skal tåle en full melkekartong midt på brospennet (1 kg).

Bordbasket

Materialer: Alt som ligger klart.

Tidsbruk: Planlegging 2 min, gjennomføring 10 min.

Krav:

Ballen må følge en bane og gå gjennom en kurv.

Ballen kan ikke berøres etter at den er satt i gang.

Kurven som ballen skal gå gjennom må lages.

Banen må være minst 2 meter og kan ikke være en rett linje.