Analog introduksjon til programmering

Oppgåve 1

De skal jobbe i par.

Begge bygger ein liten figur av dei utdelte legoklossane. Ta vare på figuren din heilt til slutten, men ikkje vis han til nokon andre.

Lag ei oppskrift slik at den andre du jobbar med klarer å bygge ein heilt lik figur. Du får ikkje forklare ekstra undervegs, alt må vere skrive ned.

Så byter de oppskrift og bygger den andre sin figur.

- Blei dei like?
- Kva måtte du endre i oppskrifta for at figurane skulle bli heilt like?
- Kunne du ha laga oppskrifta kortare eller enklare?

Oppgåve 2

De skal jobbe i par.

Begge lagar ei oppskrift på korleis den andre må gå for å kome seg ut døra. Det er ikkje lov å forklare undervegs.

Så byter de oppskrift, og ein om gongen prøver ut den andre si oppskrift og ser kor du hamnar.

Deretter justerer de begge oppskriftene, men da kan de berre bruke kodeklossane på neste side.

Kor få klossar er mogleg å bruke for å få til oppgåva?

Feilsøking og forbetring

1. Skjedde det som skulle skje? / Fekk du til oppgåva?
2. Kva måtte du endre på i oppskrifta for å få til oppgåva?
3. Kunne du ha laga oppskrifta kortare eller enklare?

Kodeklossar – eit lite oversyn

Gå ett steg

Gå to steg

Gå tre steg

Desse klossane seier kor mange steg du skal gå bortover.

Sving 90° mot høgre

Sving 90° mot venstre

Snu 180°

Desse klossane seier kor mange grader du skal snu.

Stopp

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

Repeter 5 gonger

Repeter 10 gonger

Repeter 20 gonger

Dette er lykkjer. Dei seier kor mange gonger du skal repetere noko. Kor mange gonger du skal gå rundt gjennom lykkja.

Difor må du alltid kople han saman med ein annan kloss som seier akkurat kva du skal repetere i lykkja di.

Dersom du krasjar, så

Dersom du somnar, så

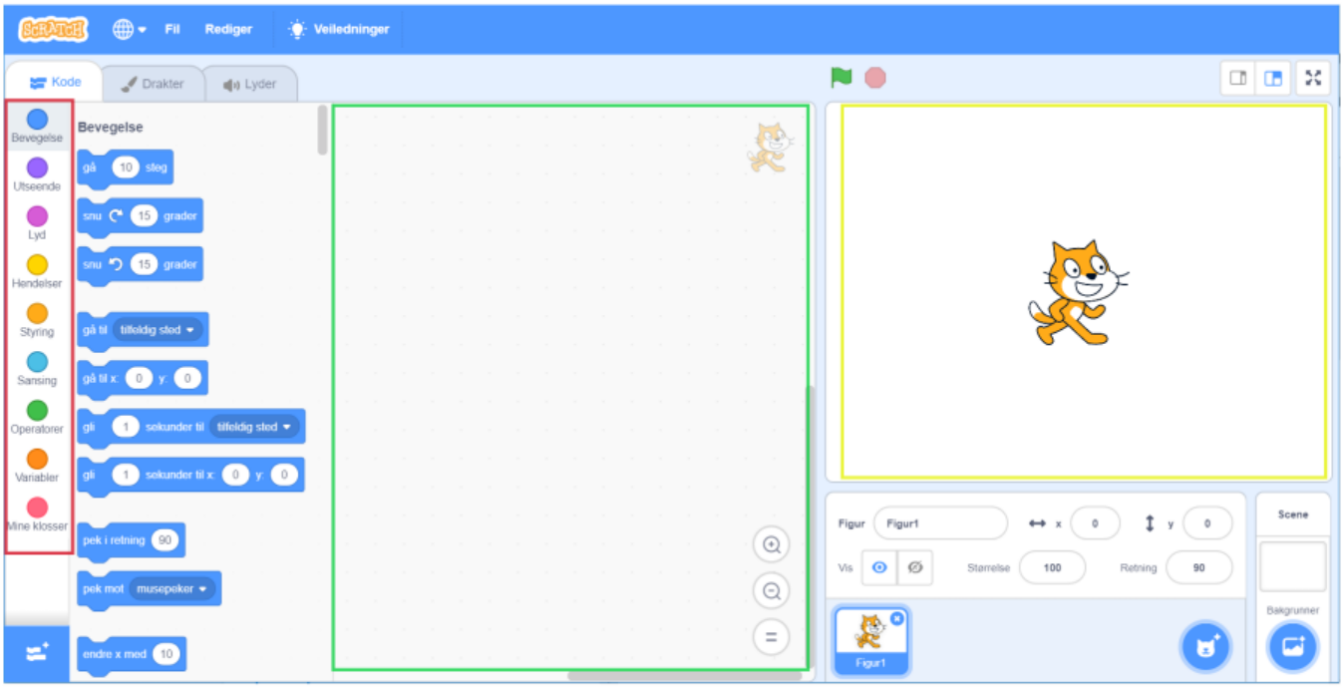
Dersom du snublar, så

Dette er dersom-klossar. Dei inneheld eit vilkår, altså DERSOM noko hender, SÅ skal du gjere noko.

Difor må du alltid kople han saman med ein annan kloss som seier akkurat kva du skal gjere dersom du krasjar, somnar eller snublar.

Scratch
- ein introduksjon

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



I den raude ramma er det ulike kategoriar som gjer det enklare å finne dei klossane ein treng.

Rørsle: Her ligg klossane for kvar katten skal røre seg.

Styring: Her ligg klossane som seier kor mange gonger noko skal gjerast eller om det skal gjerast i det hele teke.

Operatorar: Her ligg ein del rekneoperasjonar, som addisjon, subtraksjon og samanlikningar.

I den grøne ramma er feltet der ein lagar programmet.

Du trekker klossen du vil nytte inn hit, da er han ein del av programmet ditt.

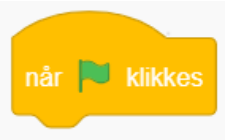
I den gule ramma ser du resultatet av programmet ditt.

Katten vil røre seg slik programmet gir han melding om.

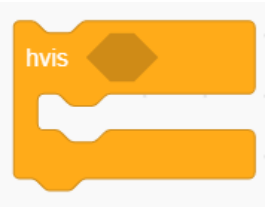
Dette området fungerer som eit koordinatsystem, der katten startar i origo.

Intro til nokre klossar i Scratch

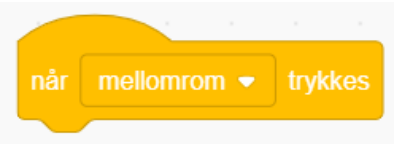
- Alle desse klossane finn du i kategorien som har lik farge som klossane



Når ein klikkar på det grøne flagget, skjer det som står i klossen kopla fast i denne.



Denne klossen kallast en viss-kloss. Og dersom det som står i det grå feltet er sant, vil det som står inni denne klossen hende.



Når ein klikkar på mellomrom-tasten, hender det som står i klossen som er festa i denne.

Døme på eit vilkår som kan settast inn i det grå feltet til dersom-klossen. Dersom vi set inn eit tal som er større enn 50, vil det som står inne i viss-klossen hende.

Denne klossen kallast ei lykkje. Det som står inne i denne klossen vil repeterast for alltid.

Denne klossen sier kva for nokre koordinatar katten skal gå til.

Oppgåve

1. Giss kva dette programmet gjer.
   - Fyll inn i det utdelte arket
2. Lag programmet i Scratch og sjå kva som hender.
3. Kva må du gjere for at katten skal kunne kome attende til origo?
   - HINT: Tenk på x- og y-koordinatar.

Ekstraoppgåve

1. Kan du få katten til å flytte seg i ei rektangelform utan å bruke nokon tastar?
2. Kva med ei trekantform?

Introduksjon til vilkår, viss-klossar og lykkjer i Scratch

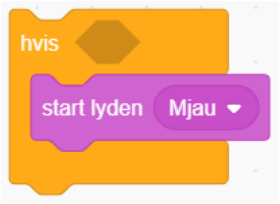
Vilkår
Eit vilkår er ein påstand som kan vere sann eller usann. Vi samanliknar ofte to tal og ser kva for eit som er størst, eller om dei er like store.

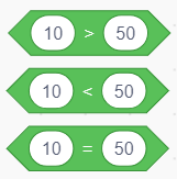
Døme på vilkår:
- Mamma er høgare enn pappa.
- Det er høgare temperatur om sommaren enn om vinteren.
- Læraren har lik skostorleik som eg har.

I Scratch er vilkåra grøne, sekskanta klossar.

Viss-blokker

- Kva: Ein viss-kloss inneheld eit vilkår. Når vilkåret er sant, vil det som står inni viss-klossen hende.
- Kvifor: Ein viss-kloss gjer at eit program kan gjere fleire ulike ting avhengig av om vilkåret er sant eller usant.





Diskuter
For kva for nokre av dei tre grøne vilkåra vil vi høyre mjauing med programmet til venstre?

Lykkjer

- Kva: Lykkjer gjer at ein del av koden i eit program blir teke opp att fleire gonger. Det finst fleire typer lykkjer. Nokre tek opp att det du skal gjere et visst tal gonger, medan andre lykkjer held fram for evig. Den tredje typen tek opp att fram til eit vilkår ikkje lenger er sant.

- Kvifor: Programmet blir lettare å få oversikt over, og det blir færre klossar enn om ein ikkje nyttar lykkjer.

Oppgåver

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

Grubleoppgåve

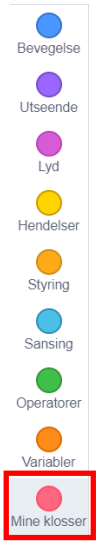
Kva trur du hender i programmet under? Vil det same hende kvar gong?

Introduksjon til funksjonar i programmering
- korleis lage eigne klossar

Kva? Ein funksjon er ein kloss som du har laga sjølv. Altså eit slags eige program. Det som skil ein funksjon frå eit vanleg program, er at du først må definere ein funksjon. Det vil seie at du må lage han. Etterpå må du køyre funksjonen i eit program. Det som står inni ein funksjon, hender ikkje av seg sjølv før du nyttar funksjonen i eit program.

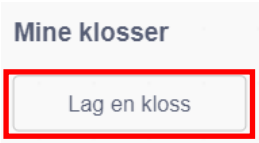
Kvifor? Programmet du lagar tek mindre plass, akkurat som når du bruker lykkjer. Funksjonen kan bli nytta fleire stader i programmet utan å måtte lage alle blokkene på ny.

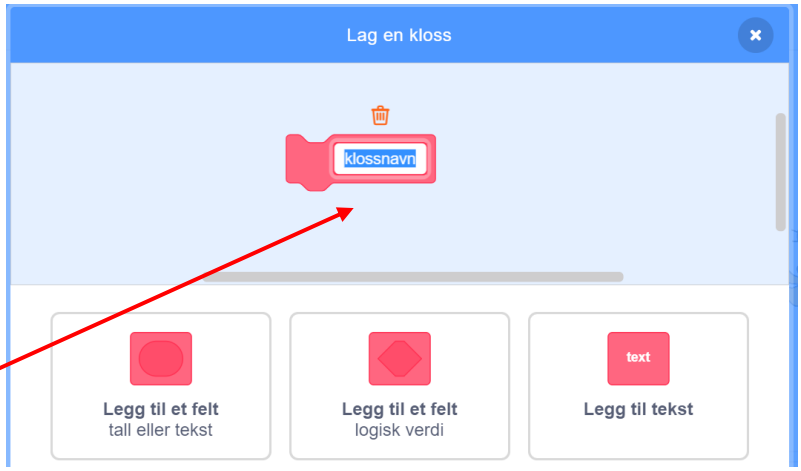
Korleis?



1. steg: Vel mine klosser i menyen på venstre sida.

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





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

4. steg: Du kan velje å legge til om funksjonen din treng å få noko informasjon for å verke. Dette kallast ofte for inndata eller input. Det treng vi til dømes om funksjonen skal rekne ut ulike reknestykke.

Dersom du lagar ein funksjon som skal rekne om ein verdi frå meter til centimeter, må funksjonen din få vite kor mange meter du vil gjere om til centimeter. Da må du legge til eit tal når du opprettar klossen til funksjonen din.

Diskuter

1. Kva gjer funksjonen til venstre?
2. Kvifor er det ein funksjon, og ikkje eit eige program?
3. Kva gjer programmet under?
4. Kvifor er dette eit program?

Oppgåve

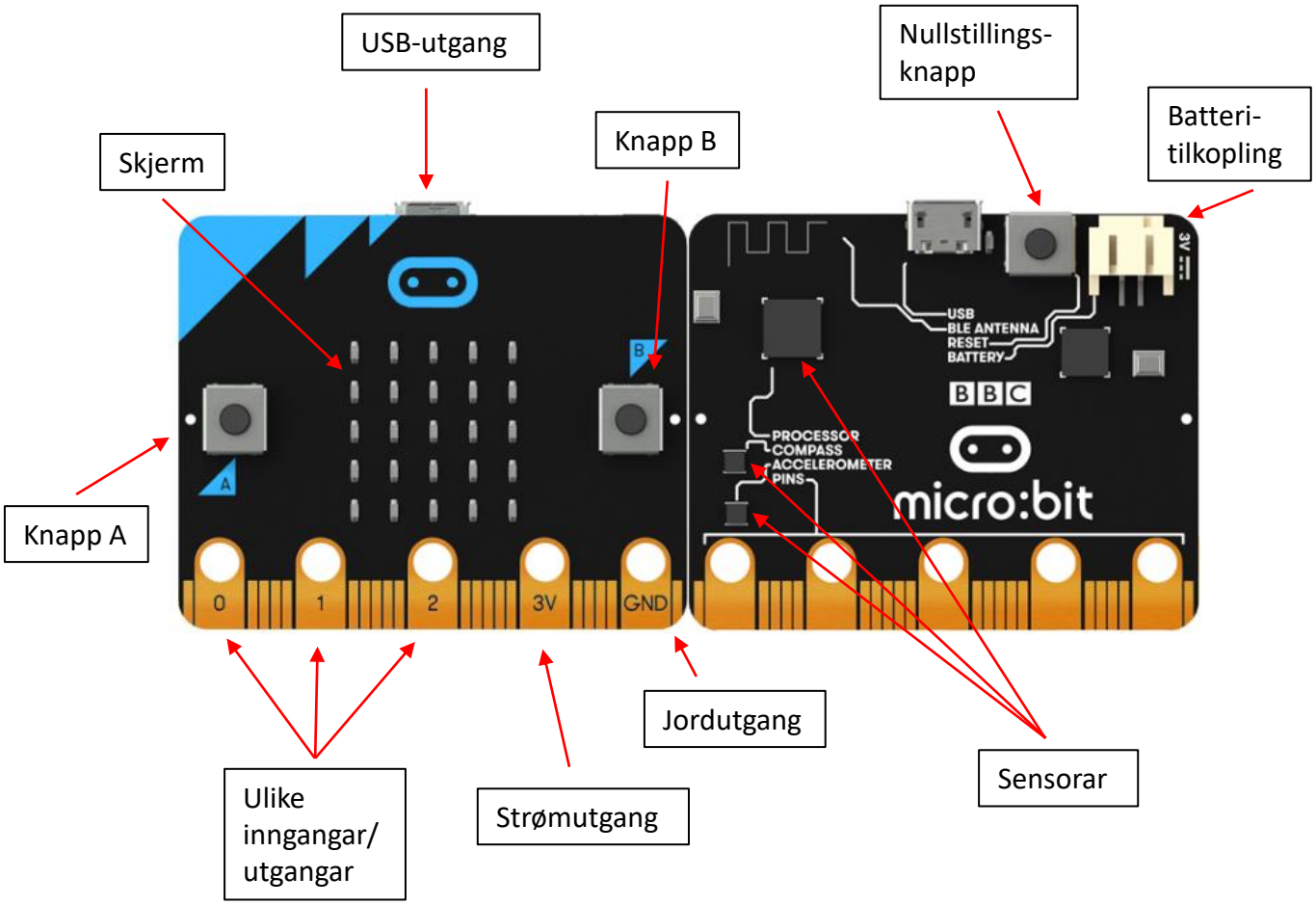
Lag ein funksjon for at katten flyttar seg i desse formene:

1. Kvadrat
2. Rektangel
3. Trekant

Kan du bruke funksjonane dine til å teikne ein samansett figur?

Introduksjon til micro:bit

Micro:bit er ein mikrokontrollar, det vil seie ein enkel datamaskin som vi kan programmere. Vi kan programmere med hjelp av ein datamaskin, eit nettbrett eller ein mobiltelefon.



Skjerm: Micro:biten har ein «skjerm» som er samansett av 25 LED-lamper som vi kan programmere til å lyse, blenke og vise ulike bilete, tal eller bokstavar.

Knappar: Han har to knappar på framsida, og ein nullstillingsknapp på baksida. Nullstillingsknappen gjer at programmet på micro:biten startar på ny.

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

Batteritilkopling: Her kan du kople til batteri for at micro:biten skal få straum.

Sensorar: Micro:biten har nokre innebygde sensorar som kan måle akselerasjon, temperatur og kompassretningar.

Tilkoplingar: Det går an å kople på andre sensorar, motorar, LED-pærer eller høgtalarar på micro:biten med hjelp av alligatorleidningar.

Bluetooth (blåtann): Micro:biten har bluetooth-tilkopling (blåtann-tilkopling) slik at han kan få overført program frå til dømes iPad som ikkje har USB-utgang.

Introduksjon til bruk av micro:bit

- for datamaskinar

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

Da kjem du inn på denne nettsida, der du kan byrje å programmere micro:biten.

Klosseliste

I den raude ramma er det ulike kategoriar som gjer det enklare å finne dei klossane ein treng.

Når du trykker på ein av kategoriane, vil klossane som høyrer til komme fram rett til høgre.

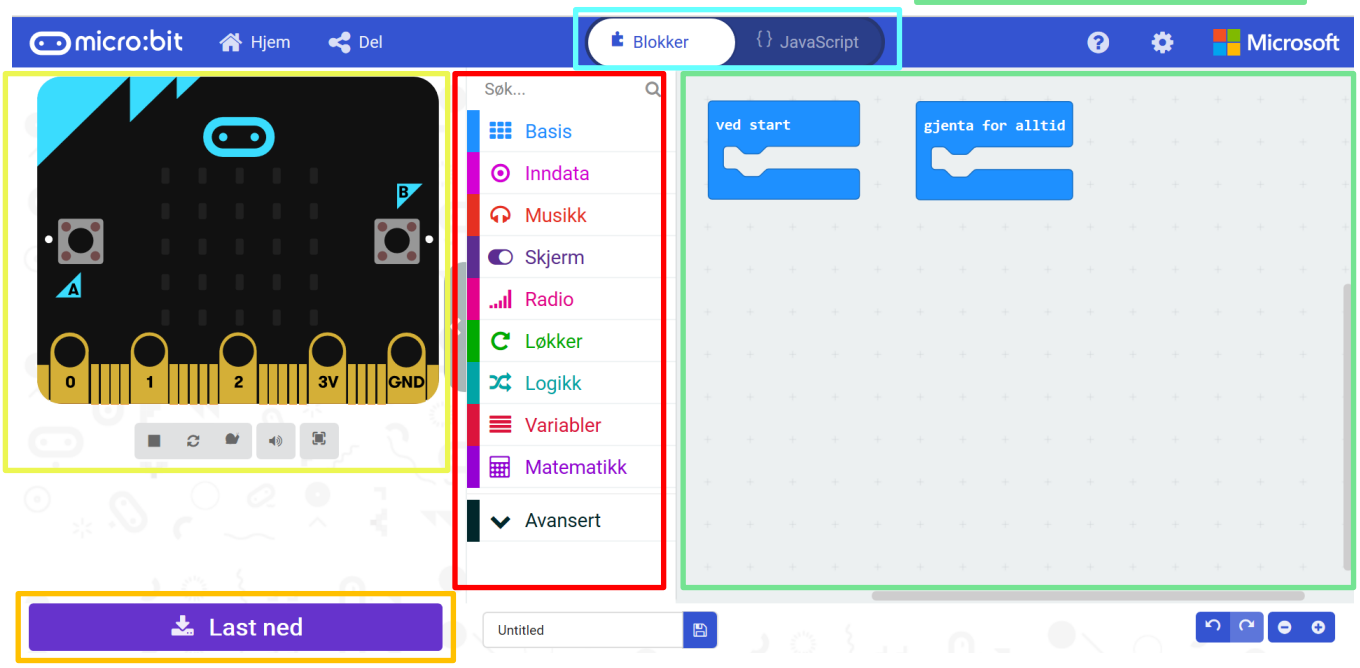
Programmeringsområde

I den grøne ramma er feltet der ein lagar programmet.

Du trekk klossen du vil bruke inn hit, da er han ein del av programmet ditt.

Simuleringsområde

I den gule ramma ser du resultatet av programmet ditt.



Nedlastingsområde

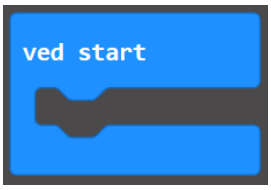
I den oransje ramma kan vi laste programmet over til micro:biten.

Programmeringsspråk

I den turkise ramma er feltet der du kan velje om du skal bruke klossar å programmere med, eller eit tekstbasert programmeringsspråk kalla Python eller JavaScript.

Intro til MakeCode for micro:bit

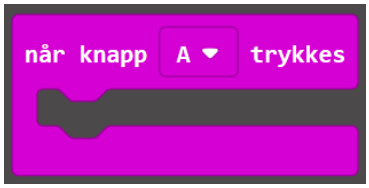
- Alle desse klossane finn du i kategorien som har lik farge som klossane



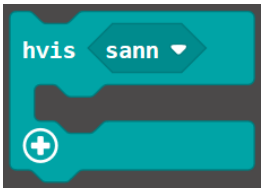
Straks programmet er overført til micro:biten hender det som står inni denne klossen.

Denne klossen kallast ei lykkje. Det som står inne i denne klossen vil bli teke opp att for alltid.

Denne klossen gjer at programmet tek ein pause før det går vidare til neste kloss.



Denne klossen gjer det som står inni han når ein trykker på knapp A på micro:biten.



Denne klossen kallest ein viss-kloss. Og dersom det som står i det mørkegrøne feltet er sant, vil det som står inni denne klossen hende.

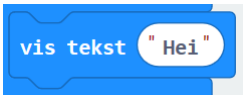
Døme på vilkår som kan bli sett inn i det mørkegrøne feltet i viss-klossen. I det øvste vilkåret kan vi sette inn to tal som han sjekker om er like. I den nedste kan vi sette inn to tal der vilkåret sjekker om det første er mindre enn det andre.

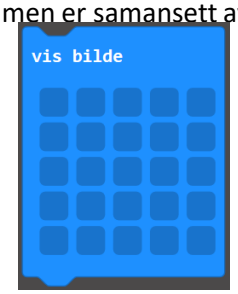
Når verdien er 1, vil denne klossen gjere at det vert sendt straum ut gjennom P0-utgangen. Når verdien er 0, blir det ikkje sendt straum ut gjennom utgangen.

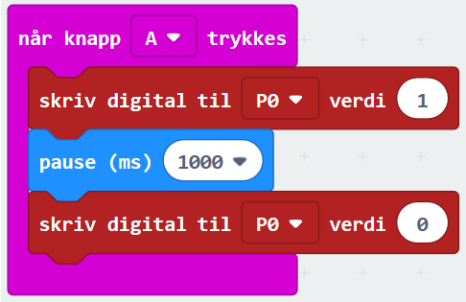
Output til skjerm

Eit viktig poeng med å bruke micro:bit er at vi kan vise informasjon på han. Det blir gjerne kalla utdata eller output, og kan visast på skjermen til micro:biten. Skjermen er ikkje ein vanleg skjerm, men er samansett av 25 raude lysdiodar som kan vere av- eller påskrudde.

I micro:biten har vi ulike klossar som kan vise ting på skjermen. Vi kan vise tekstar, bilete eller tal. Tekstane kan vere litt vanskelege å lese, for dei ruller over skjermen og kjem som ei blanding av store og små bokstavar. Vi kjem til å vise flest tal i denne boka.







Oppgåve

Giss kva dei to programma gjer.

Lag programma i makecode og før over til micro:biten – eitt om gongen.

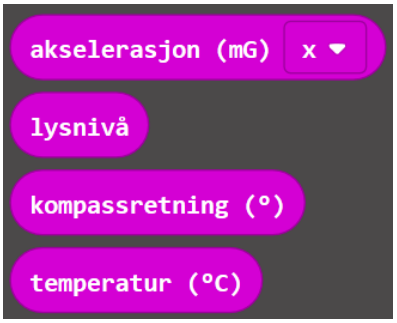
Skjedde det du trudde?

Kan du gjere nokre morosame endringar?

Inndata/input for micro:bit

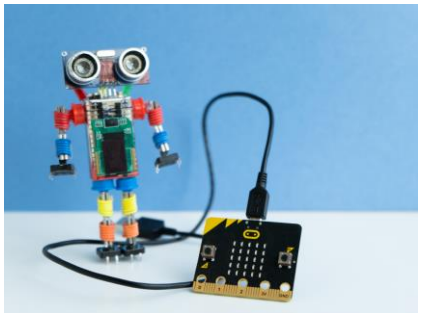
Input frå sensorar

Mesteparten av poenget med å bruke micro:bit er at vi kan samle inn informasjon med han. Det kallast gjerne inndata eller input, og kjem frå ulike sensorar. Dei kan være innebygde i sjølve micro:biten, eller vi kan kople til andre sensorar.



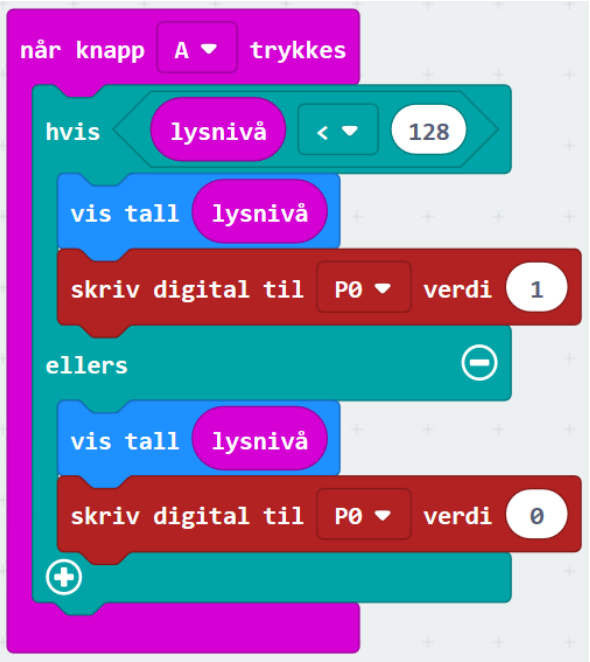
I micro:biten har vi innebygde sensorar som kan måle akselerasjonen, lysnivået, kompassretninga og temperaturen. Vi kjem oftast til å bruke temperaturen i denne boka.

Dersom vi ønsker å bruke ein sensor som ikkje er innebygd i micro:biten, må vi først kople han til ein inngang på micro:biten. I tillegg treng vi ein kloss som les av kva for ein verdi sensoren sender til micro:biten. Da nyttar vi denne klossen. Klossen les berre av kva verdien er, men gjer ingenting med han. Dei fleste sensorar vi kopler på, gir verdier mellom 0 og 1023, men det kan variere frå sensor til sensor. Det står meir om sensorar og korleis de verkar på neste side.



Snakk om

1. Kva for nokre sensorar veit du om?
2. Korleis kan ein micro:bit vise fram målingane frå sensorane?
3. Kva kan ein bruke ulike sensorar til?
4. Kan du tenke ut ei oppfinning som nyttar ein sensor for å løyse ei oppgåve.



Oppgåve

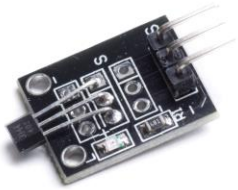
1. Giss kva dette programmet gjer.
2. Lag programmet i MakeCode, overfør det til micro:biten og sjå kva som hender.
3. Kva kan det tenkast skal vere kopla til micro:biten?
4. Kva ville du gjort med programmet for å forbetre det?

Sensorar

Ein sensor blir brukt til å måle ulike verdiar, slik som temperatur, lysstyrke, fart, akselerasjon eller lydnivå. Sensoren vert kopla til ein micro:bit og ein kan måle kor stor spenning som krevst for å trykke elektrona gjennom han. Da er det slik at i ein temperatursensor vil spenninga variere i same takt som temperaturen, og vi kan bruke spenninga vi måler til å rekne ut nøyaktig temperatur.

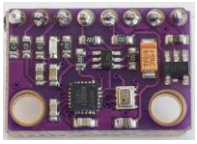
Termistor
- måler temperatur

Lyssensor
- måler lysstyrke

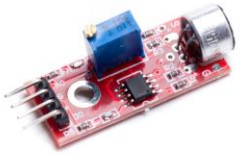


Halleffekt-sensor
- måler magnetfelt

Ultralydsensor
- måler avstand



Akselerometer
- måler akselerasjon



Lydsensor
- Måler lydstyrke

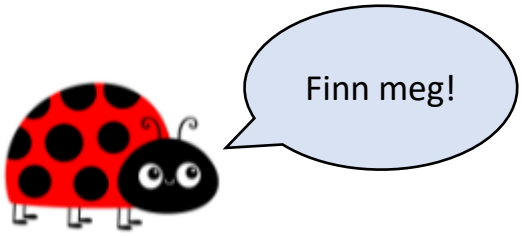
Kva er motstand?

Når eit materiale har stor motstand, er det vanskeleg for elektrona å flytte seg gjennom materialet. Da treng ein høg spenning for å dytte elektrona gjennom. Motstand blir og kalla resistans, og eininga er ohm.

Elektriske leiarar har liten motstand slik at elektrona kan røre seg lett gjennom dei, medan isolatorar har veldig stor motstand slik at det er mest umogleg for elektron å flytte seg gjennom dei.

I sensorar varierer motstanden, og da vil og spenninga variere. Denne spenninga er enkel å måle med ein micro:bit. I ein lyssensor vil motstanden variere med lysstyrken, og vi får målt ulik spenning om det er mørkt eller lyst. For ein ultralydsensor vil spenninga varierer ut frå kor stor avstand sensoren har til ein gjenstand, og tilsvarande for andre sensorar.

Feilsøking
- debugging



Absolutt ingen som programmerar gjer alt rett på første forsøk. Ikkje ein gong dei aller beste programmerarane som har jobba med programmering i mange år! Når programmet ikkje oppfører seg som planlagt, treng vi å finne ut kva som er grunnen til det. Vi må finne feila i koden vår, og kalla bugs. Det kan vere lurt å lage seg ei eiga liste med vanlege feil for deg, og bruke denne lista for å finne bugs i nye program.

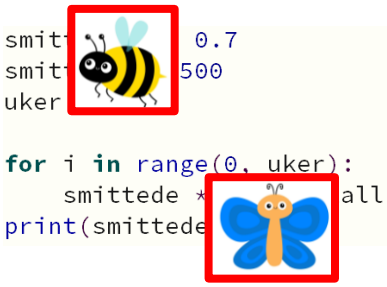
Døme på ei sjekkliste for feilsøking

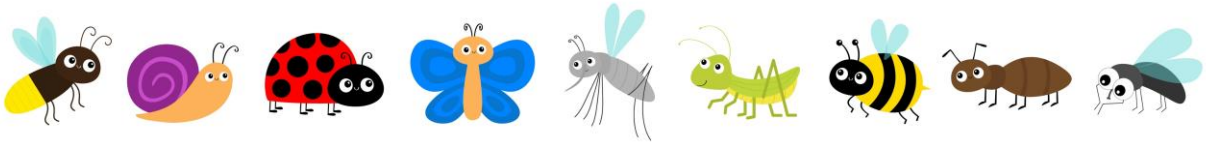
- gjer gjerne det som er enklast først, ikkje slett heile programmet

1. Kjør programmet og sjå kva som hender
   - Kva er det som ikkje stemmer med det du ønsker at programmet skal gjere?
   - Skjer ting i den rekkefølga du hadde planlagt?
   - Får du nokon feilmeldingar (tekst-programmering)?

2. Gjer synleg det som skjer inni programmet og køyr det
   - Skriv variablane til skjerm
   - Legg inn pausar i koden for å rekke å sjå kva for nokre verdiar ulike variablar har

3. Endre på programmet og køyr det
   - Har variablane rett namn med stor/liten bokstav alle stadene der du bruker same variabel?
   - Har lykkjene rett tal på repetisjonar?
   - Endre berre på ein ting om gongen, slik at det er enkelt å sjå kva ei endring fører til
   - Gir feilmeldingar nokon hint om i kva for nokre linjer feilen ligg (tekst-programmering)?
   - Gir feilmeldinger nokon hint om kva feilen er (tekst-programmering)?
     - Feil variabeltype
     - Gløymt innrykk
     - Manglande kolon eller andre teikn
   - Søk på feilkodar på internett (tekst-programmering)





Søke etter svar

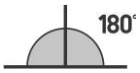
- Sjølv dei flinkaste programmerarane bruker mykje av tida på å søke opp feilkodar på internett. Det kan og vere lurt å søke på namnet til ulike funksjonar for å sjå kva for nokre inndata dei skal ha, og kva for nokre utdata dei gir.
- Står det noko i læreboka di? Les gjerne nærmare gjennom oppgåva.
- Spør nokon i klassen eller læraren.

Korleis verkar ein servo?

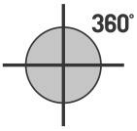
En servo er ein liten elektrisk motor som kan dreie fram og attende eller rundt og rundt. Han som berre kan dreie fram og attende blir kalla ein 180°-servo. Han som kan dreie rundt og rundt blir kalla ein 360°-servo (eller kontinuerlig servo).

Vinklar

Servoane får namn frå vinklar. Ein vinkel kan vere mellom 0° og 360°. Ein vinkel på 90°, kallast ein rett vinkel og ser slik ut.



Ein vinkel på 180° er det doble av ein rett vinkel, og da får vi faktisk ei rett linje!



Dersom du tek fire rette vinklar, endar du opp med ein vinkel på 360°. Da er du komen heilt rundt! Dei servoane som kallast 360°-servoar går jo rundt og rundt.

Kva tyder kontinuerlig?
Noko som held fram utan opphald.

Ein 180°-servo kan berre dreie 180°, og ikkje rundt og rundt.

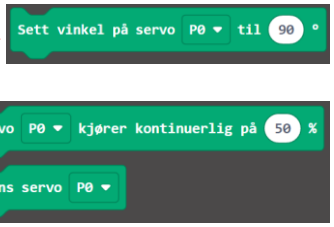
Det går typisk ikkje an å styre farten til denne typen servoar, berre vinkelen dei dreier til.

Programmering av servoar

Da kjem du til å trenge nokre av desse klossane.

180 °-graders servo:
Denne set kva for ein vinkel servoen skal dreie til.

360°-graders servo:
Denne set farten til servoen.
Denne stoppar servoen.

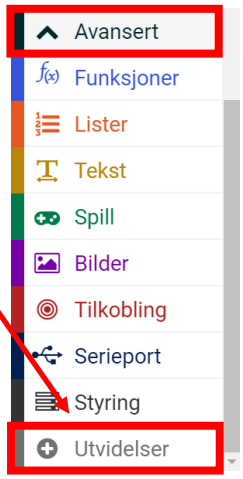


Diskuter

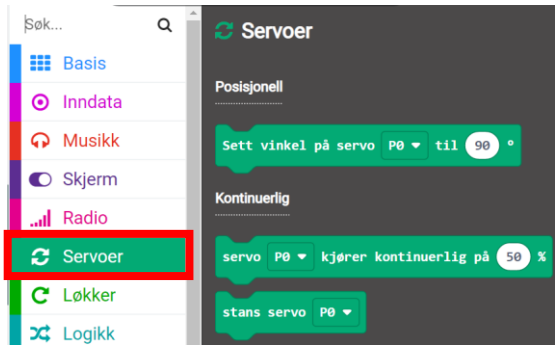
Kva må du gjere for å få 360°-servoen til å dreie i motsett retning?

Desse klossane finn du slik

1. Vel avansert i menyen, han er svart.
2. Trykk der det står utvidelser, nedst.
3. Så vel du servo-utvidelsen.



4. Vel servo i menyen, så finn du klossane du treng.



Servoen treng litt meir straum enn det som kjem frå micro:biten. For å vere sikker på at han får nok straum til å virke slik han skal, må ein bruke eit kretskort som gjer at vi kan kople til batteri i tillegg til micro:biten.

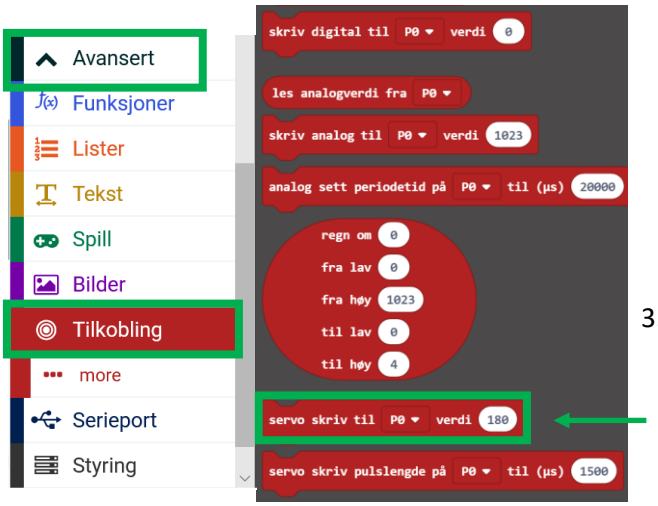
De kan la vere å legge til servo-tillegget, da må de bruke ein annan kloss enn vist på førre side. Det er berre ein kloss uavhengig av kva type servo ein nyttar, men verdiane ein skriv inn, får ulikt resultat ut frå servotypen.

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

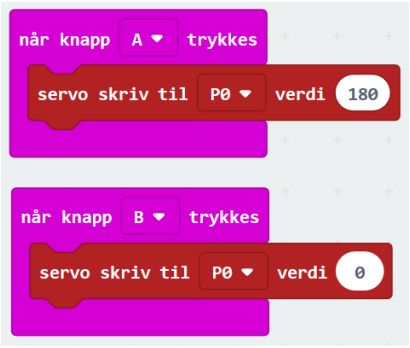
360°-servo: Verdien angir dreiefart som servoen dreier rundt og rundt med.

Denne klossen finn du slik

1. Vel avansert i menyen, han er svart.
2. Gå nedover i menyen til du finn tilkopling. Han er mørkeraud.
3. Så finn du klossen du treng nest nedst (og nedst).



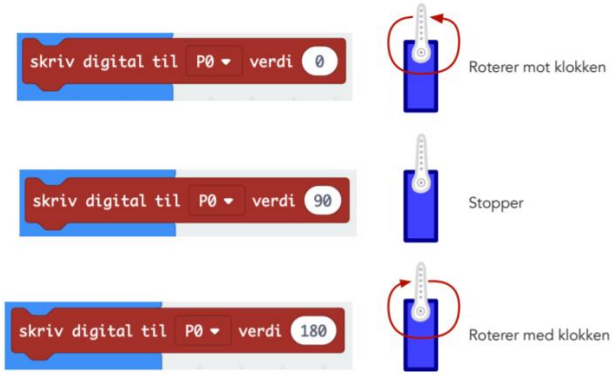
Programmering av 180°-servoar



Ein 180°-servo kan dreie 180° kvar veg, men ikkje dreie heilt rundt. Ein kan typisk ikkje styre farten til denne typen servoar, berre kva for ein vinkel dei dreier til.

Det øvste programmet gjer at servoen dreier 180° i den eine retninga. Det nedste programmet gjer at servoen dreier attende til utgangspunktet.

Programmering av 360°-servoar (kontinuerlege servoar)



Om vi skriv verdien 0 til servoen, vil han rotere mot klokka så fort han kan.

Når vi aukar verdien, vil han rotere saktare heilt til vi kjem til 90, der motoren står stille.

Når vi aukar verdien over 90 opp mot 180 grader, roterer han raskare og raskare med klokka.

Her er verdien altså ikkje vinkelen, men dreiefart.

Miniutfordringar

Det skeive tårnet i Pisa

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

Bruk av tid: Planlegging 2 min, gjennomføring 10 min.

Krav:

Bruk dei utdelte materiala.

Bygg tårn med gjennomsnittshøgd på minst 160 cm på papplata.

Klossane må vere i kontakt med plata, og plata skal vere intakt.

Utskytingsmekanisme

Materiale: Alt som ligg klart.

Bruk av tid: Planlegging 2 min, gjennomføring 10 min.

Krav:

Lag ein utskytningmekanisme som får svampen til å flyge minst 5 meter.

Utskytingsmeknismen må vere samansett av 5 ulike materiale.

Uidentifisert flygande objekt (UFO)

Materiale: Alt som ligg klart.

Tidsbruk: Planlegging 2 min, gjennomføring 10 min.

Krav:

Lag eit flygande objekt som skal vere i lufta lengst mogleg.

Bruk dei utdelte materiala.

Det flygande objektet skal dra ein blyant.

Bordtennis-basketball

Materiale: Alt som ligg klart.

Tidsbruk: Planlegging 2 min, gjennomføring 10 min.

Krav:

Ballen må følge ein bane og gå gjennom ei korg.

Ballen kan ikkje rørast etter at han er sett i gang.

Korga som ballen skal gå gjennom må lagast.

Banen må vere minst 2 meter og kan ikkje vere ei rett linje.

Bakkeløp

Materiale: Alt som ligg klart.

Tidsbruk: Planlegging 2 min, gjennomføring 10 min.

Krav:

Lage ein gjenstand som bruker nøyaktig 5 sekund på å trille ned papplatebakken.

Bruk dei utdelte materiala.

Det er ikkje lov å røre gjenstanden etter at han er sloppen.

Brubygging

Materiale: Alt som ligg klart.

Tidsbruk: Planlegging 2 min, gjennomføring 10 min.

Krav:

Bruk dei utdelte materiala.

Bygg ei bru med eit spenn på minst 50 cm.

Brua skal kunne bere ein full melkekartong midt på bruspennet (1 kg).