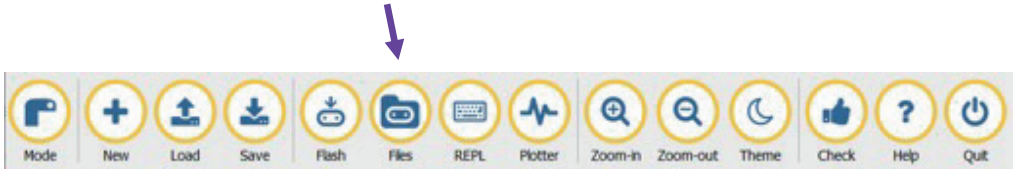
Opplegg 12 – **Momentan vekstfart og speedometer**

En sonar sender ut et lydsignal, og måler hvor lang tid det tar før signalet blir reflektert tilbake til sonaren. Dersom man vet hva lydfarten er, kan man enkelt regne ut avstanden til det som reflekterer lydsignalet. Sonaren vet hva lydfarten er, og vi kan dermed bruke den til å måle avstander med. For å kunne bruke en sonar i Python Mu, så må vi importere et bibliotek på selve micro:biten. Dette biblioteket er egentlig et program som gjør det enklere for oss å programmere sonaren, uten å måtte tenke på hvordan alle detaljene i målingene.

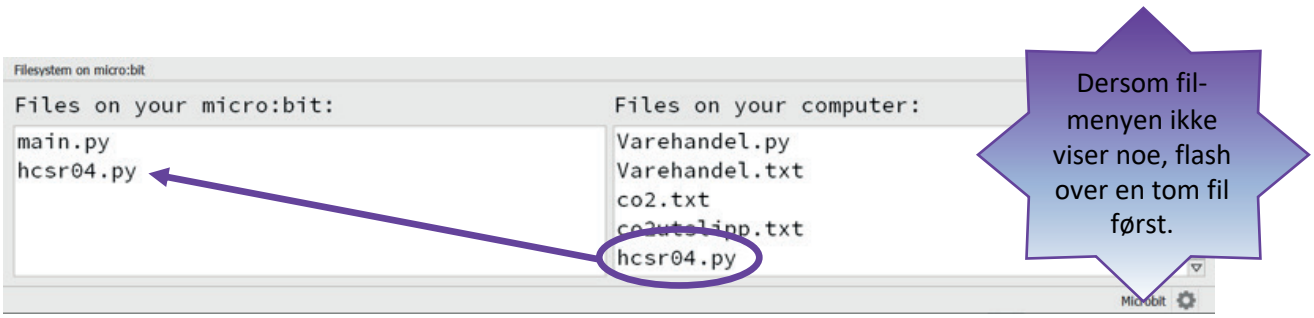
Det første vi må gjøre er å få tak i biblioteket til sonaren, og det kan finnes på www.kunnskapsfilm.no

Kopier koden i filen «hcsr04.py» over i en ny fane (fil) i Python Mu og lagre den som «hcsr04.py» ved å trykke på «Save» og lagre den i katalogen som dukker direkte opp.

Deretter må denne fila legges over på selve micro:biten, og det gjøres ved å koble micro:biten til datamaskinen, og trykke på «Files».



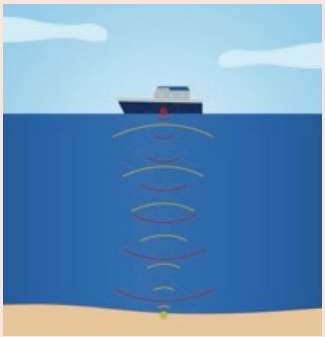
Da kommer denne menyen opp i Mu, og du må lete i feltet på høyre side til du finner fila som du nettopp lagret: «hcsr04.py». Når du finner den, drar du den over i det venstre feltet. Da vil lyset på baksiden av micro:biten blinke til programmet er overført.



Da gjenstår det å lage selve programmet som bruker sonaren til å måle avstander med. Under er det en puslespill-oppgave der kodelinjene må ryddes i riktig rekkefølge for å lage dette programmet.

Puslespill-oppgave

```
print((HCSR04(),))
from hcsr04 import HCSR04
sleep(1000)
while True:
from microbit import *
```



Lag et speedometer med micro:bit

Oppgave

Bruk en micro:bit med en tilkoblet sonar til å lage et speedometer. For å få til dette må dere kalibrere avstandsmålingene fra sonaren slik at dere får oppgitt avstanden i meter. Se tips nederst på siden for kalibrering.

Fase 1: Gjennomfør informasjonsinnhenting for å få idéer. Hva slags tilkoblinger trengs?

Fase 2: Idémyldre og planlegge
Det er viktig at dere er åpne for alle slags ideer og ikke er for kritiske, da kan nyttige forslag bli avfeid for tidlig.

1. Tenk selv først og tegn gjerne skisser.
2. Forklar ideen din for de andre på gruppa.
3. Hele gruppa diskuterer de ulike ideene, og lager en felles plan for bygging.

Fase 3: Gjennomføring

- Lag speedometeret.
- Lag programmet for å måle avstanden ved hjelp av sonaren. Husk at resultatet av kalibreringen må inkluderes i programmet.
- Utvid programmet slik at det kan beregne momentanfarten fra de kalibrerte avstandsmålingene.

Fase 4: Test om speedometeret, inkludert programmeringen, virker som planlagt.

Fase 5: Sammenlign resultatene med andre i klassen.

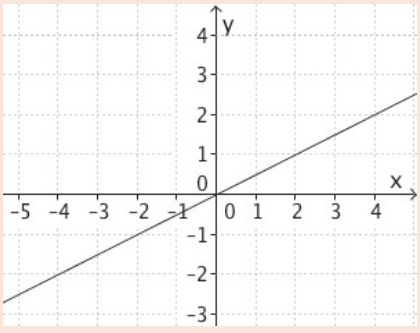
- Virker svarene deres fornuftige?

Fase 6: Gå tilbake til de andre fasene for å gjøre eventuelle forbedringer.

Fase 7: Dokumenter prosjektet på en valgfri måte.

Kalibreringsoppgave – regresjon

1. Plott alle micro:bitens avstandsmålinger og de tilhørende målingene dere gjør med linjal eller metermål i Geogebra.
2. Finn en matematisk modell ved å foreta en regresjon for de målte dataene.
3. Hvordan ser funksjonen ut?
4. Passer den bra med datapunktene deres?



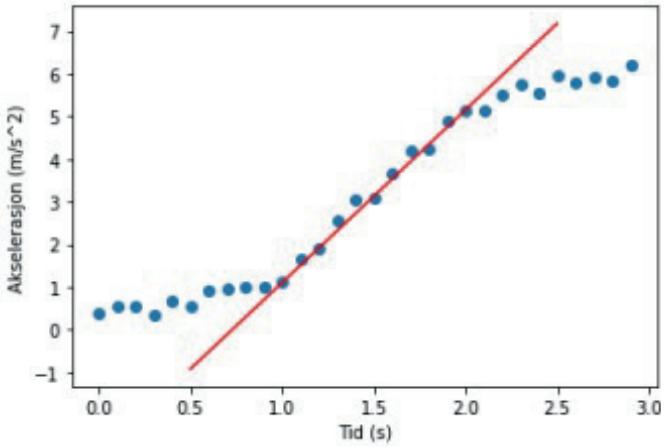
Momentan vekstfart

To former for vekstfart

Vi skiller mellom gjennomsnittlig og momentan vekstfart. Den gjennomsnittlige vekstfarten forteller oss hvor raskt noe øker mellom to x-verdier, mens den momentane vekstfarten forteller oss hvor rask økningen er i et bestemt punkt. Vi begynner med den gjennomsnittlige.

Gjennomsnittlig vekstfart

I følgende kode forutsetter vi at vi allerede har importert et datasett med x- og y-verdier. Vi starter med å plotte punktene. Deretter beregner vi den gjennomsnittlige akselerasjonen ved å bruke $a=\Delta x/\Delta y$. I vår figur var N = 30, og vi ønsker å beregne den gjennomsnittlige vekstfarten for indeksverdier mellom 10 og 20.




```
scatter(x,y)
xlabel("Tid (s)")
ylabel("Akselerasjon (m/s^2)")

aks_gj = (y[20]-y[10])/(x[20]-x[10])
```

Plotter x- og y-verdiene. Beregner deretter den gjennomsnittlige akselerasjonen.

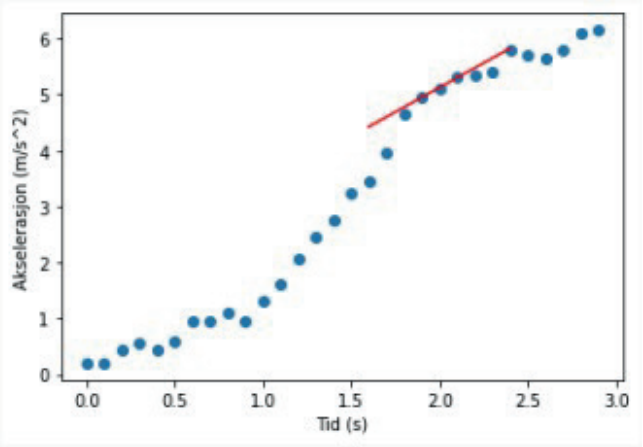
```
def f(x,a,x1,y1):
    return a*(x-x1)+y1

s = linspace(0.5,2.5)
t = f(s,aks_gj,x[10],y[10])
plot(s,t,'r')
```

Bruker ettpunktsformelen til å finne funksjonsuttrykket til den røde linja på figuren. Til sist plotter vi linja.

Momentan vekstfart

Hvis vi vil finne den momentane vekstfarten, gjør vi akkurat som for den gjennomsnittlige, men vi velger bare to punkter som ligger rett i nærheten av hverandre. Dermed får vi et uttrykk for hvordan funksjonsverdiene er i ferd med å endre seg akkurat der. I eksempelet nedenfor har vi valgt indeksverdier 19 og 21 for å se på veksten rundt i=20, altså x=2,0.



Plotter grafen og beregner gjennomsnittlig akselerasjon for x-verdier i nærheten av hverandre.

```
scatter(x,y)
xlabel("Tid (s)")
ylabel("Akselerasjon (m/s^2)")
aks_gj = (y[21]-y[19])/(x[21]-x[19])
```

Bruker ettpunkts-formelen og plotter den røde linja

```
def f(x,a,x1,y1):
    return a*(x-x1)+y1
s = linspace(1.6,2.4)
t = f(s,aks_gj,x[19],y[19])
plot(s,t,'r')
```