☐ Python: Bevegelse

LAST NED PDF

Introduksjon

Din micro:bit er utstyrt med et akselerometer som måler bevegelse langs tre akser:

- X - tilte fra venstre til høyre.
- Y - tilte fremmover og bakover.
- Z - bevegelse opp og ned.

Det er en funksjon for hver akse som returnerer et positivt eller negativt tall som indikerer antall milli-g krefter. Den viser 0 når du står i vater langs den aksen.

For eksempel, her er et enkelt program som viser deg hvor mye i vater enheten din er langs X aksen:

```
from microbit import *

while True:
    reading = accelerometer.get_x()
    if reading > 20:
        display.show("H")
    elif reading < -20:
        display.show("V")
    else:
        display.show("-")
```

Dersom du holder enheten flatt skal den vise `-`; om du derimot roterer den til venstre eller høyre burde den respektivt vise `V` og `H`.

Siden vi ønsker at enheten vår skal reagere på forandring hele tiden bruker vi en `while`-løkke. Den første tingen som skjer *inne i while-løkka* er at den måler langs `X` aksen og lagrer resultatet i variabelen `reading`. Fordi akselerometerer er *så* sensitivt har jeg gitt den et slingringsmonn på +/-20. Det er derfor `if` og `elif`-setningene sjekker for respektivt `> 20` and `< -20`. Mens `else`-setningen betyr at dersom verdien til `reading` er mellom -20 og 20 så sier vi at enheten er i vater. For hver av disse setningene så bruker vi displayet til å vise passende tekst.

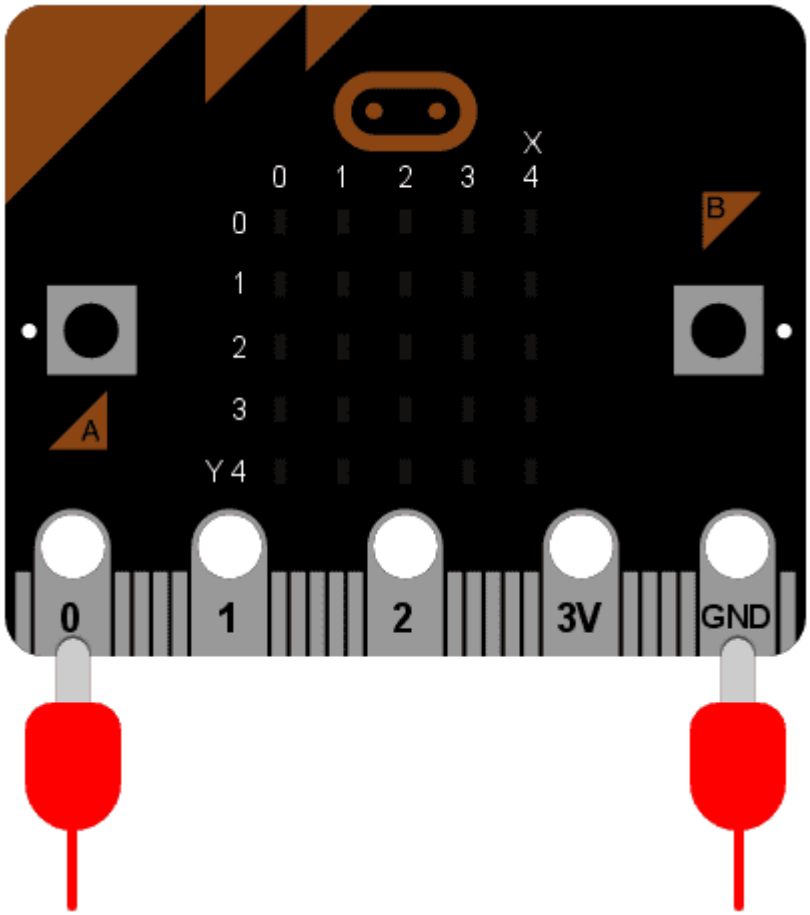
Det er og en `get_y` metode for Y aksen og en `get_z` metode for Z aksen.

Dersom du lurer på hvordan en mobiltelefon vet om du holder mobilen horisontalt eller vertikalt så er det fordi den bruker et akselerometer på akkuratt samme måte som programmet ovenfor. Spillkontrollere inneholder også akselerometer som kan hjelpe deg å navigere.

Musikalsk galskap

En av de beste egenskapene til MicroPython er hvor sømløst du kan sette sammen ulike funksjoner til micro: bit'en. For eksempel, la oss forvandle enheten til et "musikalsk" instrument. Hvorfor jeg satte hermetegn omkring musikalsk finner du nok fort ut ;-)

Sett inn en høytaler slik som du gjorde i oppgaven "Lage musikk med micro:bit". Bruk krokodilleklemmer til å feste pin 0 og GND (jord) til den positive og negative inngangen på høytaleren - det spiller ingen rolle hvilken vei de er koblet.



Hva skjer dersom vi leser av akselerometer og spiller de som toner? La oss finne det ut:

```
from microbit import *
import music

while True:
    music.pitch(accelerometer.get_y(), 10)
```

Den viktigste linjen er uten tvil på slutten, og den er relativt enkel. Vi *nøster* lesingen fra Y aksen som frekvensen og mater den inn i `music.pitch` metoden. Vi lar denne freksenen bare spille i 10 millisekunder fordi vi ønsker å forandre tone raskt når enheten tippes til en side. Siden jeg bruker en `while`-løkke som går for alltid så leser den hele tiden endringene til Y aksen.

Det er alt!

Tipp enheten fremmover og bakover. Dersom lesingen langs Y aksen er positive så vil den endre tonehøyden avspilt av micro:bit'en.

Klarer du å spille en melodi på dette enkle instrumentet? I siste del skal vi se på noen enkle forbedringer du kan gjøre.

Prøv det ut selv

- [ ] Endre instrumentet ditt slik at du kan tippe det både bakover og fremmover for å endre tonehøyden.

Dette kan for eksempel gjøres ved enten å legge inn en `if`setning, eller med å bruke `abs` funksjonen

HINT

- [ ] Endre koden slik at du kan styre hvor lenge tonene varierer ved å variere høyden i z-retningen.

HINT

Vi mennesker har problemer med å høre frekvenser over 18 000Hz og under 40Hz. Mens de frekvensene som er behagelige å høre på gjerne ligger mellom 80 - 400Hz. For å fikse micro:bit'en slik at den bare spiller toner i dette intervalet kan vi gjøre noe som ligner på dette

```
from microbit import *
import music

while True:
    Y = A * abs(accelerometer.get_y())
    if Y < 80:
        Y = 80
    elif Y > 400:
        Y = 400
    Z = abs(accelerometer.get_z())
    music.pitch(Y, Z)
```

ALTERNATIV

- [ ] Finn gode tallverdier for `A` i koden over. Forstår du hvordan koden fungerer?