Øving 3

INGT1001 - Ingeniørfaglig innføringsemne

Formål med øvingen

I denne oppgaven skal dere bli kjent med bruk av **fargesensor** og **ultrasonisk sensor**, i tillegg til mer kunnskap om **pybricks** for å programmere lego-roboten deres.
Det er en fordel om dere eksperimenterer litt med disse sensorene, slik at dere behersker bruken av dem godt ved avsluttet øving.

Oppgavebeskrivelse

Mens roboten deres vikarierer for Bernt skal den årlige robot-revyen avholdes på Gløshaugen. Bernt lurer derfor på om du kan steppe inn og ta underholdningen han bruker å gjøre.

Bernt bruker å kjøre rundt Frimerket[1]. Deres robot kommer ikke til å testes på selve Frimerket, men heller på en liten versjon vi har laget. Hvor dere skal kjøre er markert med en svart strek (15 mm bred), så dere må bruke fargesensoren til å følge etter denne streken. Banen dere skal kjøre langs vil være firkantet med avrundede hjørner. Dere kommer til å kjøre mot klokka.

Mens roboten deres kjører langs banen skal den stoppe hvert 10. sekund og underholde publikum. Dere kan selv velge hva dere har lyst til å gjøre på disse stoppene, det kan være alt fra å spille av lyd fra høyttaleren til roboten til å få den til å danse. Dere må implementere minst 4 forskjellige underholdnings-bidrag, men ved hvert stopp skal det velges tilfeldig hvilket roboten deres faktisk utfører (se kode-eksempel for hvordan man implementerer tilfeldigheter i Python).

Når underholdningsnummeret til roboten deres nærmer seg slutten kommer det til å, på magisk vis, dukke opp en hindring foran den på banen. Roboten deres skal stoppe 5 - 20 cm fra denne hindringen (dere velger selv nøyaktig hvor innenfor dette intervallet dere vil stoppe), og spille av enten lyden “CHEERING” eller “FANFARE”.

Leveranse

Vis frem fungerende løsning til en av læringsassistentene og gå gjennom koden sammen med dem.

[1] Gressplenen han holder til på, ligger mellom Hovedbygget og Stripa.

Tips og kode-eksempler

Relevante klasser for øvingen

I dokumentasjonen står det hvordan et objekt kan instansieres og hvilke metoder som er tilgjengelig for hver type objekt.

Dokumentasjonen for de forskjellige klassene finner dere i menyen til venstre på denne nettsiden:
https://docs.pybricks.com/en/latest/index.html

- EV3Brick (hubs – Programmable Hubs)
- Motor (ev3devices – EV3 Devices)
- DriveBase (robotics – Robotics)
- UltrasonicSensor (ev3devices – EV3 Devices)
- ColorSensor (ev3devices – EV3 Devices)
- Alle klasser og metoder dere bruker for å lage underholdning

Tilkobling av motorer og sensorer

Motor-klasser kobles til porter på EV3-en markert med bokstav, for eksempel `Motor(Port.B)` for en motor som er koblet til port B på EV3-en.

Sensor-klasser kobles til porter på EV3-en markert med tall. I koden deres **må** tall-portene skrives med S først, for eksempel `TouchSensor(Port.S1)` for en trykk-sensor som er koblet til port 1 på EV3-en.

Merk: Sensorene har ulik nøyaktighet. Dere må derfor eksperimentere med disse, slik at dere vet hva dere kan forvente fra hver sensor.

Kode-eksempel: generering av tilfeldige tall

I python kan man benytte seg av biblioteket `random` for å generere tilfeldige tall. For å generere tilfeldige heltall kan man benytte seg av randoms `randint`-metode:

```
import random

# Generate a random integer from 2 to 5, including both limits
random_integer = random.randint(2,5)
```

Kode-eksempel: beregning av tid

I python kan man bruke `time`-biblioteket for å finne ut hva klokken er på et gitt tidspunkt. Man kan også få antall sekunder siden 1. januar 1970, noe som gir en enkel måte å programmere gjentagende hendelser på.

```
import time

# Record the time when the program starts
last_time = time.time()

# Variable to keep the program running until a break-condition is reached
execute_program = True

# Count the number of times a pause has happened
pause_numbers = 0

while execute_program:
    print('executing program')

    current_time = time.time()
    if current_time - last_time >= 10:
        print('10 seconds or more have passed. Wait 2s before resuming')
        wait(2000)

        # Increase the number of times a pause has happened
        pause_numbers += 1

        # Update last time this break happened
        last_time = time.time()

    # Add a stop-condition to ensure the program can end
    if pause_numbers == 7:
        execute_program = False
```

Kode-eksempel: kalibrering av fargesensor

Det kan være nyttig å definere en eller flere *globale variabler* som lar dere enkelt endre verdier dere trenger å benytte dere av i koden mens dere jobber med utvikling. Disse globale variablene kan enten skrives i en egen python-fil eller dere kan skrive de øverst i filen dere arbeider i. I vårt tilfelle passer det siste best, siden vi ikke trenger særlig mange variabler.

Vi trenger globale variabler som lar oss bestemme hvilken terskel vi skal bruke for å si at en farge er svart/hvit. Her er det mange muligheter, eksemplet nedenfor krever at både rødt, grønt, og blått skal ha verdier under 50% for at en farge skal kalles svart. Pybricks-biblioteket lar oss kalle metoden `rgb()` på et fargesensor-objekt, og vi får hvor mange prosent av henholdsvis rødt, grønt, og blått fargesensoren har oppfattet. 100% av alle tre fargene tilsvarer hvit, 0% av alle tre fargene tilsvarer svart.

I dokumentasjonen for `ColorSensor`-klassen kan dere lese mer om tilgjengelige metoder.

```
# Instantiate an object for the color-sensor
colorSensor = ColorSensor(Port.S4)

# Global variables that defines the maximum RGB-values a sensor-reading
from the color-sensor can have for the color to be defined as black
RED = 50
GREEN = 50
BLUE = 50

# Get the percentage of red, green, and blue color from the color-sensor
(red, green, blue) = colorSensor.rgb()
# If any of the colors from the sensor has lower rgb-values than what we
# defined in our global variables, then the color is assumed to be black
is_black = red < RED or green < GREEN or blue < BLUE
```