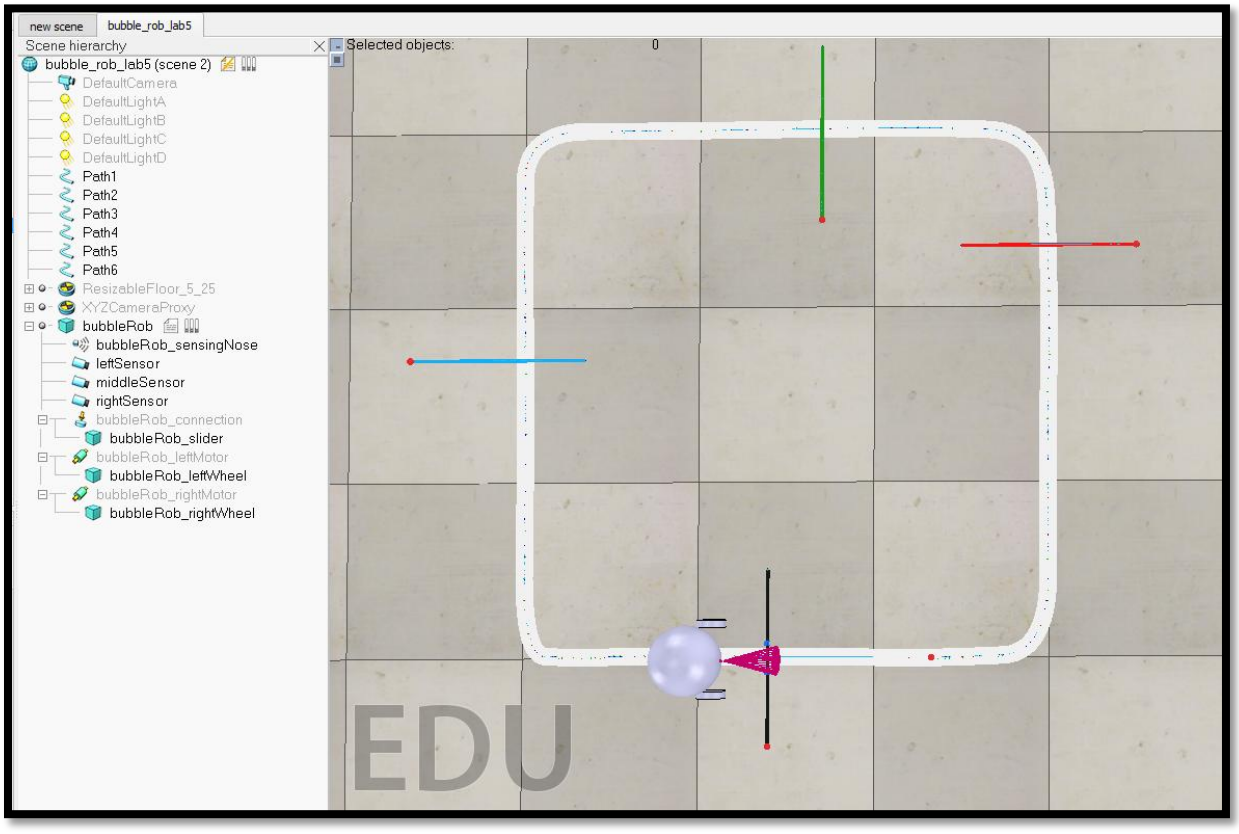
# Sprawozdanie Lab 5

Andrzej Żaba, gr_lab 4, nr indeksu: 401490

Postój robota w określonych punktach trasy postanowiono rozwiązać z użyciem kolorowych linii na zamkniętej trasie.




W celu określenia trybów ruchu dodano następujące zmienne:

```
-- Color based movement variables
 colorMode = 2      -- 1 - black | 2 - white
 whichColorToStop = 1   -- 0 - black | 1 - red | 2 - green | 3 – blue
```

*colorMode* – determinuje kolor linii po jakiej domyślnie ma się poruszać robot (pozostałość z poprzedniej instrukcji)
*whichColorToStop* – determinuje kolor linii na której robot ma się zatrzymać (czerwony, zielony lub niebieski).
Linię czarną potraktowano jako przystanek bazowy robota (np. magazyn) w którym zatrzymuje się po każdym kursie.

```
tresh_red = 0.39
tresh_green = 0.28
tresh_blue = 0.61
tresh_black = 0.09
```

Kod odpowiadający za zatrzymanie robota wygląda następująco:

```
    for i=1,3,1 do
        vision_result,data = sim.readVisionSensor(floorSensorHandles[i])
        if(vision_result>=0) then
            if data[11] > .90 then
                sensorReading[i] = true
            elseif whichColorToStop == 1 and i == 2 and data[11] > tresh_red - 0.06
and data[11] < tresh_red + 0.06 then
                wait(1)
            elseif whichColorToStop == 2 and i == 2 and data[11] > tresh_green -
0.06 and data[11] < tresh_green + 0.06 then
                wait(1)
            elseif whichColorToStop == 3 and i == 2 and data[11] > tresh_blue -
0.06 and data[11] < tresh_blue + 0.06 then
                wait(1)
            elseif i == 2 and data[11] > tresh_black - 0.06 and data[11] <
tresh_black + 0.06 then
                wait(1)
            end
        end
    end
```

funkcję *wait()* zadeklarowano w następujący sposób:

```
function wait(seconds)
  local start = os.time()
  repeat until os.time() > start + seconds
end
```

Całkowicie wstrzymuje ona działanie symulacji na podaną ilość sekund. Stwierdzono wysoką skuteczność przyjętego rozwiązania, jednak uznano zatrzymanie całej symulacji za nieoptymalną praktykę.

Podjęto próby modyfikacji dotychczas stworzonego programu, aby zatrzymywał on pracę silników a nie całą symulację.

Dodano w tym celu zmienne kontrolujące tryb zatrzymania się:

```
-- Stop mode and time variables
 stopTime = 1          -- how many seconds to wait()
 stopFlag = 0          -- gives a signal to enter stop mode
 stopUntillTime = -1    -- wheather stop mode is active or not
```

Następnie zmodyfikowano pętlę odczytu z czujników:

```
   if stopFlag == 0 then
      for i=1,3,1 do
         vision_result,data = sim.readVisionSensor(floorSensorHandles[i])
         if(vision_result>=0) then
            if data[11] > .90 then
               sensorReading[i] = true
            elseif whichColorToStop == 1 and i == 2 and data[11] > tresh_red - 0.06
and data[11] < tresh_red + 0.06 then
               stopFlag = 1
               stopUntillTime = sim.getSimulationTime() + 1

            elseif whichColorToStop == 2 and i == 2 and data[11] > tresh_green -
0.06 and data[11] < tresh_green + 0.06 then
                stopFlag = 1
               stopUntillTime = sim.getSimulationTime() + 1
            elseif whichColorToStop == 3 and i == 2 and data[11] > tresh_blue -
0.06 and data[11] < tresh_blue + 0.06 then
                stopFlag = 1
               stopUntillTime = sim.getSimulationTime() + 1
            elseif i == 2 and data[11] > tresh_black - 0.06 and data[11] <
tresh_black + 0.06 then
                stopFlag = 1
               stopUntillTime = sim.getSimulationTime() + 1
            end
         end
      end
   end
```

Dodano również kilka niezbędnych funkcji:

```
if (stopUntillTime < sim.getSimulationTime()) then
   -- When in forward mode, we simply move forward at the desired speed
      sim.setJointTargetVelocity(leftMotor,leftV)
      sim.setJointTargetVelocity(rightMotor,rightV)
      stopFlag = 0
   else
   -- When in stop mode, both engines are off
      sim.setJointTargetVelocity(leftMotor,0)
      sim.setJointTargetVelocity(rightMotor,0)
end


if (backUntilTime<sim.getSimulationTime()) then
   -- When in forward mode, we simply move forward at the desired speed
      if (stopFlag == 0) then -- check if not in stop mode
         sim.setJointTargetVelocity(leftMotor,leftV)
         sim.setJointTargetVelocity(rightMotor,rightV)
      end
else
   -- When in backward mode, we simply backup in a curve at reduced speed
      if (stopFlag == 0) then
         sim.setJointTargetVelocity(leftMotor,-speed/2)
         sim.setJointTargetVelocity(rightMotor,-speed/8)
      end
end
```

Stwierdzono, że jest to lepsze rozwiązanie od zatrzymania całej symulacji. Natomiast zauważono problemy w funkcjonowaniu zastosowanego skryptu. Mianowicie zdarzało się, że buble rob zatrzymywał się na danej linii np. 4 razy z rzędu. Działo się tak ponieważ czujnik wizyjny był w stanie zarejestrować dany kolor ponownie, nim robot zdążył wyjechać poza linię. Nie wykonywał przez to poprawnie postawionego zadania a sama jazda wydawała się przez to mało płynna. Zmiana grubości linii nie rozwiązywała problemu. Przy zbyt cienkiej linii zdarzało się, że robot całkowicie ją pomijał i nie zatrzymywał się w wyznaczonym punkcie.

Podjęto próbę rozwiązania tej kwestii w sposób softwearowy. Skojarzono owy problem ze znanym z elektroniki i mikrokontrolerów problemem z drganiem styków.

Postanowiono podejść do problemu w następujący sposób:

Po prawidłowym odczycie kolorowej linii i zatrzymaniu się robota wyłączyć czujnik wizyjny na krótką chwilę, tak aby robot mógł bez przeszkód, płynnie opuścić kolorową linię i jechać po białej trasie.

Do tego celu zadeklarowano konieczne wartości:

```
stop_mode_ticks = 50
stop_mode_debouncing_counter = 50
```

*stop_mode_ticks* – to stała określająca przez ile ‘ticków’ nie będą rejestrowane odczyty z czujników. Drogą eksperymentalną ustalono, że 50 to wystarczająca wartość.

*stop_mode_debouncing_counter* – to zmienna która zlicza ilość ‘ticków’ inkrementując się co każde wykonanie głównej pętli programu. Ustalono ją bazowo na 50 w celu, gdyby pierwszy przystanek miał znajdować się tuż za startem.

Zmodyfikowano instrukcje warunkowe odczytu z czujników wizyjnych:

```
if stopFlag == 0 then
    for i=1,3,1 do
        vision_result,data = sim.readVisionSensor(floorSensorHandles[i])
        if(vision_result>=0) then
            if data[11] > .90 then
                sensorReading[i] = true
            elseif whichColorToStop == 1 and i == 2 and data[11] > tresh_red - 0.06 and data[11] <
tresh_red + 0.06 and stop_mode_debouncing_counter >= stop_mode_ticks then
                stopFlag = 1
                stopUntillTime = sim.getSimulationTime() + 1
                stop_mode_debouncing_counter = 0

            elseif whichColorToStop == 2 and i == 2 and data[11] > tresh_green - 0.06 and data[11] <
tresh_green + 0.06 and stop_mode_debouncing_counter >= stop_mode_ticks then
                stopFlag = 1
                stopUntillTime = sim.getSimulationTime() + 1
                stop_mode_debouncing_counter = 0
            elseif whichColorToStop == 3 and i == 2 and data[11] > tresh_blue - 0.06 and data[11] <
tresh_blue + 0.06 and stop_mode_debouncing_counter >= stop_mode_ticks then
                stopFlag = 1
                stopUntillTime = sim.getSimulationTime() + 1
                stop_mode_debouncing_counter = 0
            elseif i == 2 and data[11] > tresh_black - 0.06 and data[11] < tresh_black + 0.06 and
stop_mode_debouncing_counter >= stop_mode_ticks then
                stopFlag = 1
                stopUntillTime = sim.getSimulationTime() + 1
                stop_mode_debouncing_counter = 0


            end
        end
    end
end
```

Dodano również inkrementację licznika oraz warunek, aby co jakiś czas redukował swoją wartość – aby nie operować niepotrzebnie na dużych liczbach oraz aby nie wyjść poza zakres bajtów przeznaczonych dla zmiennej.

```
stop_mode_debouncing_counter = stop_mode_debouncing_counter + 1
if stop_mode_debouncing_counter > 100000 then
    stop_mode_debouncing_counter = stop_mode_ticks
end
```

Stosując dość proste rozwiązanie otrzymano znaczącą poprawę w funkcjonowaniu robota. Jazda teraz okazała się płynna a zatrzymania odpowiednie.