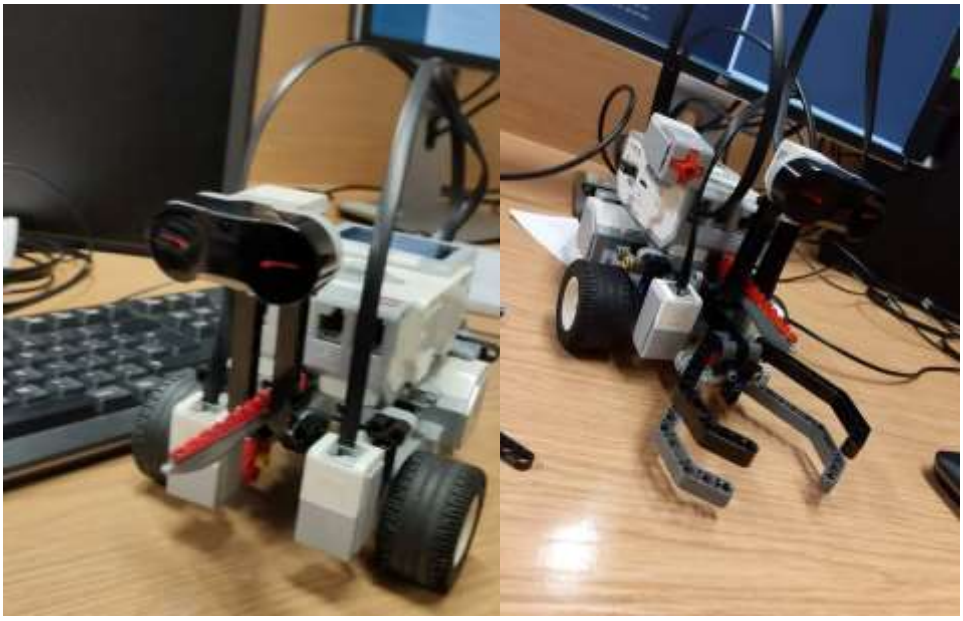
Wstęp do Robotyki

Robot line follower & transporter

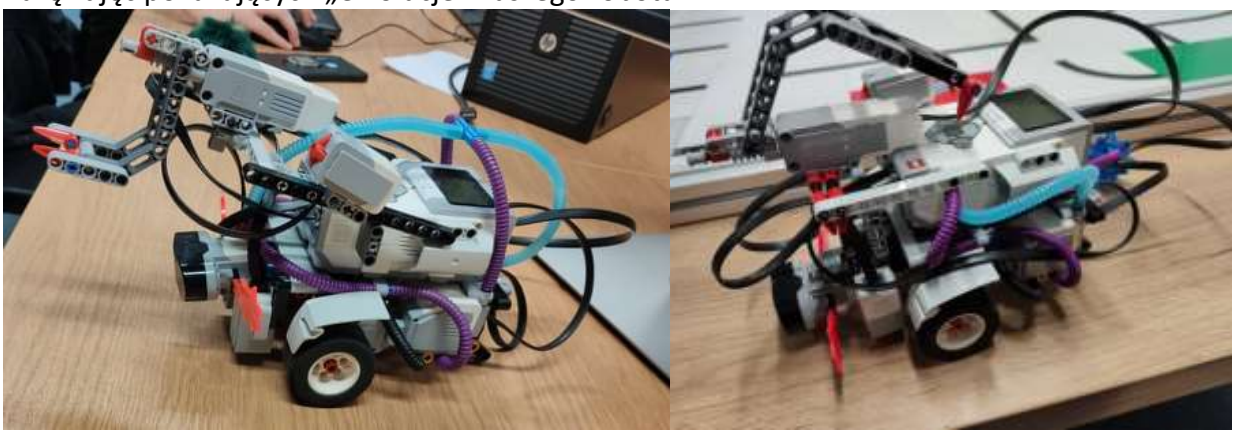
Michał Kwarciński, Kacper Marchlewicz

Budowa

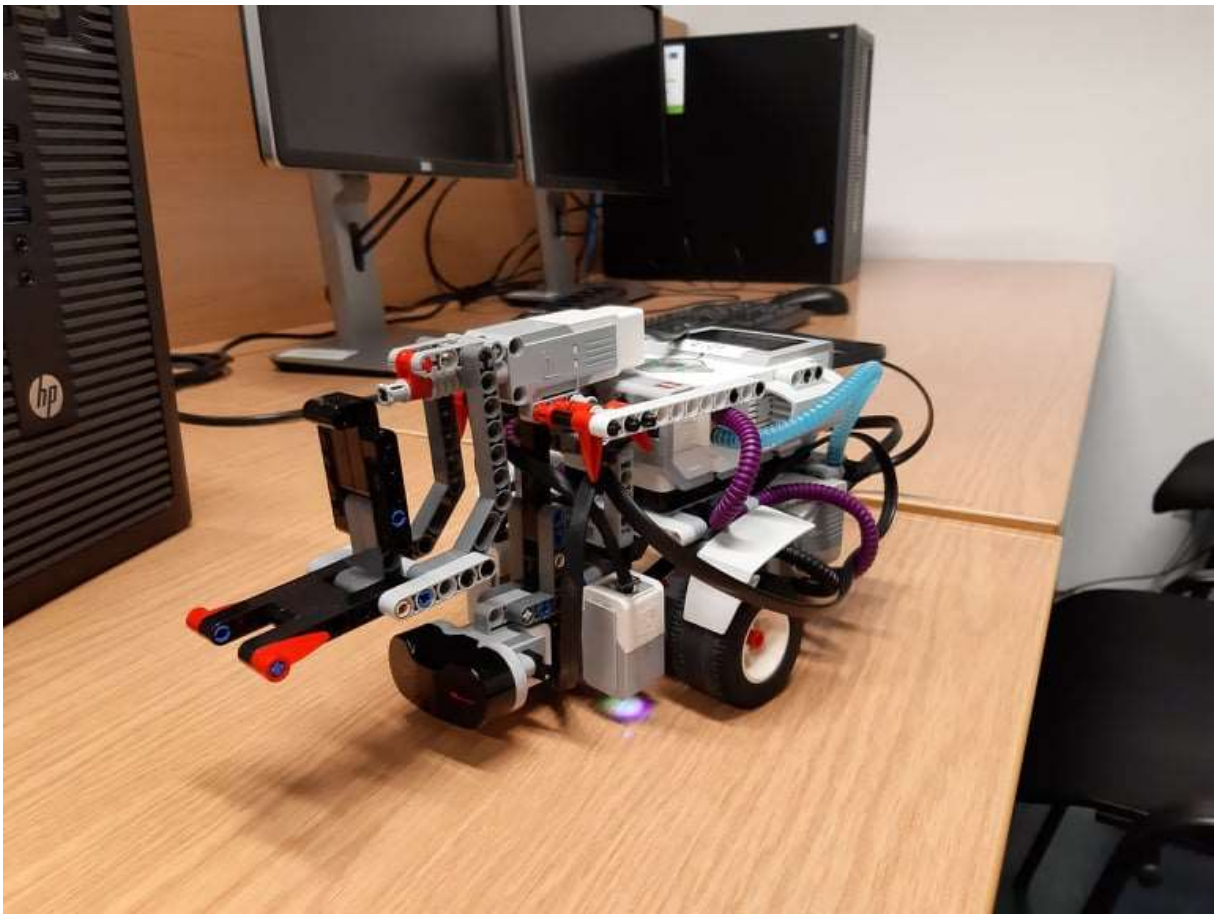
Robot składał się z podstawowych elementów LEGO Mindstorms – kostki EV3, dwóch dużych serwomotorów, dwóch czujników koloru, czujnika podczerwieni i średniego serwomotoru. Za napęd odpowiadały dwa duże serwomotory znajdujące się po bokach konstrukcji, wyśrodkowane jak tylko się dało. Z tyłu umieściliśmy małe koło, którego zadaniem było zapewnienie równowagi konstrukcji. Na przedzie zamontowaliśmy dwa czujniki koloru z odstępem około szerokości linii. Pomiędzy nimi zamontowaliśmy czujnik podczerwieni. Podnośnik stworzyliśmy w oparciu o średni serwomotor. Całość stanowiła swego rodzaju karoserię do której w prosty sposób można było zamontować kostkę.



Parę zdjęć pokazujących „ewolucje” naszego robota



Tak oto powstała wersja finalna:



Zadanie 1 – Line follower

Postanowiliśmy stworzyć kod w oparciu o prosty algorytm – serię komend „if”. Czujniki koloru pracowały w trybie zwracającym ilość odbitego światła. Po serii pomiarów mogliśmy stwierdzić czy robot znajduje się na białym bądź czarnym kolorze, czy też zaraz na krawędzi. Odczyty były inne niż proste wykrywanie czarny-biały, gdy czujnik znajdował się na krawędzi kolorów wartość była prawie średnią wartości. Pozwoliło to na określenie czy czujnik zbliża się do linii. Na tej podstawie dobraliśmy tak wartości (ulubioną metodą prób i błędów), aby program działał jak najskuteczniej. Podczas testów zdarzało mu się wyjechać z ostrego zakrętu, lecz ostateczny tor pokonał bez problemu.

Kod:

```
#!/usr/bin/env python3
import ec3dev.ev3 as ev3
import time

m_1 = ev3.LargeMotor('outA')
m_2 = ev3.LargeMotor('outB')
m_m = ev3.MediumMotor('outC')
l_right = ev3.ColorSensor('in1')
```

```
l_left = ev3.ColorSensor('in2')
ir = InfraredSensor('in4')

l_right.mode = 'COL-REFLECT'
l_left.mode = 'COL-REFLECT'

while True:
    loaded = False
    r_color = l_right.value()
    l_color = l_left.value()
    if r_color > 20 and l_color > 20:
        m_1.run_timed(time_sp = 10, speed_sp = 100)
        m_2.run_timed(time_sp = 10, speed_sp = 100)
    elif r_color < 20 and l_color > 20:
        m_1.run_timed(time_sp = 10, speed_sp = -50)
        m_2.run_timed(time_sp = 10, speed_sp = 50)
    elif r_color > 20 and l_color < 20:
        m_1.run_timed(time_sp = 10, speed_sp = 50)
        m_2.run_timed(time_sp = 10, speed_sp = -50)
    elif r_color < 20 and l_color < 20:
        m_1.run_timed(time_sp = 10, speed_sp = 100)
        m_2.run_timed(time_sp = 10, speed_sp = 100)
```

Zadanie 2 – Transporter

Musieliśmy trochę pozmieniać nasz algorytm. Z różnymi kolorami musieliśmy zmienić tryb działania czujników na rozpoznawanie kolorów. Był to tryb dość zerojedynkowy, więc postanowiliśmy wkomponować metodę stanową w kod. Do zadania postanowiliśmy wybrać jako kolory startu i mety kolejno czerwony i żółty. Z niebieskiego i zielonego musieliśmy zrezygnować, gdyż czujniki miały problem z rozpoznawaniem ich – potrafiły dawać zamienne odczyty. Robot jechał po linii na tej samej zasadzie jak poprzednio (lecz tutaj wyłącznie na rozpoznawaniu biały-czarny co w widoczny sposób obniżyło dynamikę ruchów), po wykryciu koloru robot przechodził w odpowiedni stan. Pozwalało nam to na zaplanowanie co robot będzie robił dalej, np.: po kolorze skręć, jedź aż wjedziesz na kwadrat, potem podnieś obiekt, wróć na tor. Zauważyliśmy, że jeśli zwiększymy czas obrotu tylko jednego koła podczas zakrętu, wykonywał on się łagodniej (nie szarpał się na boki). Ostatecznie udało się uzyskać kod zapewniający poprawność wykonania zadania. Robot poprawnie dotarł do docelowego kwadratu, zabrał obiekt i dostarczył go w odpowiednie miejsce.

Kod:

```
#!/usr/bin/env python3
from ev3dev2.motor import LargeMotor, MediumMotor, OUTPUT_A, OUTPUT_B,
OUTPUT_C, OUTPUT_D, SpeedPercent, MoveTank, MoveSteering
from ev3dev2.sensor import INPUT_1, INPUT_2, INPUT_3, INPUT_4
from ev3dev2.sensor.lego import ColorSensor, InfraredSensor
from ev3dev2.led import Leds
```

```
import time

# left
m_1 = LargeMotor(OUTPUT_B)
# right
m_2 = LargeMotor(OUTPUT_A)
m_m = MediumMotor(OUTPUT_D)
l_right = ColorSensor(INPUT_4)
l_left = ColorSensor(INPUT_1)
ir = InfraredSensor(INPUT_3)

l_right.mode = 'COL-COLOR'
l_left.mode = 'COL-COLOR'

# lift repair


def turn_right(time = 80):
    m_1.run_timed(time_sp = time, speed_sp = -200)
    m_2.run_timed(time_sp = time + 150, speed_sp = 400)

def turn_left(time = 80):
    m_1.run_timed(time_sp = time + 150, speed_sp = 400)
    m_2.run_timed(time_sp = time, speed_sp = -200)

def turn_right_90(time = 2500):
    m_1.run_timed(time_sp = time, speed_sp = -200)
    m_2.run_timed(time_sp = time + 500, speed_sp = 400)

def turn_left_90(time = 2500):
    m_1.run_timed(time_sp = time + 500, speed_sp = 400)
    m_2.run_timed(time_sp = time, speed_sp = -200)

def go_straight(time, speed):
    m_1.run_timed(time_sp = time, speed_sp = speed)
    m_2.run_timed(time_sp = time, speed_sp = speed)

def turn_180():
    m_1.run_timed(time_sp = 4350, speed_sp = -110)
    m_2.run_timed(time_sp = 4350, speed_sp = 110)


def run_repair(time = 2000, speed = 400):
    m_m.run_timed(time_sp = time, speed_sp = speed)

#run_repair(2000, -450)
#run_repair(2000, 10)

"""
```

```
    0 -> No color
    1 -> Black
    2 -> Blue
    3 -> Green
    4 -> Yellow
    5 -> Red
    6 -> White
    7 -> Brown
"""

def main():

    loaded = False
    back_on_track = False
    is_going_to_unload = False
    on_track = False
    stan_2 = False
    duble_cocol = True

    while True:

        r_color = l_right.value()
        l_color = l_left.value()
        ir_value = ir.value()

        # 2
        if loaded and not back_on_track and not stan_2:
            back_on_track = True
            stan_2 = True
            time.sleep(5)
            turn_180()
            time.sleep(5)
        # 3
        elif loaded and back_on_track and r_color == 1 and l_color == 1:
            m_1.run_timed(time_sp = 1500, speed_sp = -200)
            m_2.run_timed(time_sp = 1500 + 500, speed_sp = 400)
            time.sleep(0.3)
            back_on_track = False
            on_track = True
        # 1
        elif r_color == 5 and l_color == 5 and ir_value <= 4 and not loaded:
            run_repair()
            loaded = True

        # 6
        elif loaded and (r_color == 4 and l_color == 4) and not on_track and
not duble_cocol:
            time.sleep(1)
            run_repair(2000, -450)
```

```
            time.sleep(5)
            go_straight(5000, -50)
            time.sleep(5)
            # turn_180()
            # time.sleep(5)
            loaded = False
            back_on_track = False
            is_going_to_unload = False
            on_track = False
            stan_2 = False
            duble_cocol = True
            break
        # 4
        elif on_track and loaded and ((r_color == 4) or (l_color == 4)) and
duble_cocol:
            is_going_to_unload = True
            on_track = False
            duble_cocol = False
        # 5
        elif is_going_to_unload and (r_color == 1 or r_color == 6) and
(l_color == 4):
            turn_left_90()
            on_track = False

        # 5
        elif is_going_to_unload and (r_color == 4) and (l_color == 1 or
l_color == 6):
            turn_right_90()
            on_track = False
        else:
            if r_color == 5 and l_color == 5:
                go_straight(50, 100)
            elif r_color == 5 and l_color == 6:
                turn_right_90()
            elif r_color == 6 and l_color == 5:
                turn_left_90()
            elif r_color == 5 and l_color == 1:
                turn_right()
            elif r_color == 1 and l_color == 5:
                turn_left()
            elif r_color == 6 and l_color == 6:
                go_straight(50, 150)
            elif r_color == 1 and l_color == 6:
                turn_right()
            elif r_color == 6 and l_color == 1:
                turn_left()
            elif r_color == 1 and l_color == 1:
                go_straight(50, 150)
            else:
```

```
                go_straight(50, 150)

main()
```

Finalne spostrzeżenia

Przez większość czasu borykaliśmy się z dziwnymi problemami z robotem. Długi czas oczekiwania na wgranie bądź uruchomienie programu, dziwne glicze. Robot robił rzeczy których nie powinien – raz jechał gdy nie widział koloru, a na planszy stał w miejscu, zamiast skrętu o 90 stopni robił obroty o prawie 720 stopni. Lewy czujnik często nagle przestawał odczytywać kolory, jakby się wyłączał. Na ostatnich zajęciach (jedna grupa skończyła przed nami, więc były wolne elementy) postanowiliśmy wymienić kostkę i czujnik. Problemy zniknęły jak ręką odjął. Uważamy, że dziwne zachowanie robota na konkursie wynikało z problematycznej kostki. (Za te wnioski z glicza podczas konkursu obiecał nam Pan punkt 😊)