# Wstęp do Robotyki - dokumentacja

Autorzy:
Mikołaj Kita, 298797
Miłosz Łopatto, 305898

## Spis treści dokumentacji z laboratoriów z przedmiotu Wstęp do Robotyki

# Spis obrazów w dokumentacji



# Cel laboratorium

Celem laboratorium było zapoznanie nas z zagadnieniami związanymi z ruchem robotów oraz z percepcją środowiska w robotach kołowych/gąsienicowych. Na laboratorium poznaliśmy różne rodzaje czujników i odpowiadające im własności oraz ich cechy szczególne poprzez zadania, których tematy obejmowały wprowadzenie do problematyki automatycznej nawigacji.

# Zadania laboratoryjne

## Linefollower

Zadanie Linefollower polegało na tym, aby robot podążał za czarną linią naklejoną na białej planszy złożonej z “płytek”.
Trasa wyznaczona w zadaniu zawierała tylko dwa kolory (czarny i biały), natomiast jako przeszkody wprowadzone zostały ostre zakręty oraz skrzyżowania i należało ją przejechać w obie strony. Zadanie nie zawierało ograniczeń czasowych i prędkość nie miała wpływu na punktację zadania.

## Zawody (wyścig)

W czasie trwania zadania zorganizowany został wyścig, który odbywał się na planszy w kształcie owalu:

*Obraz 1 Tor użyty podczas wyścigu*

Wyścig polegał na przejechaniu łącznie trzech okrążeń, przy czym:

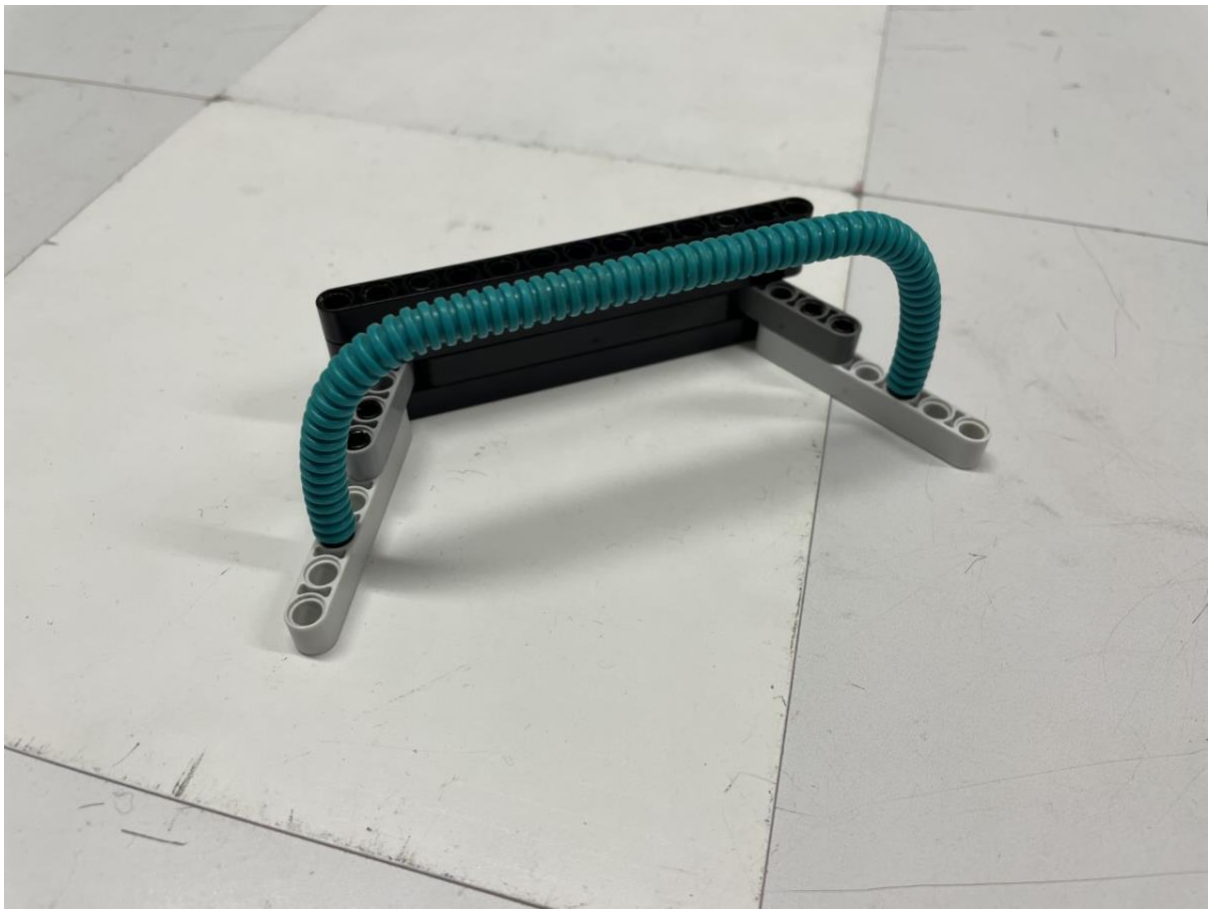
- Pierwsze okrążenie było w jednym kierunku dla wszystkich zespołów
- Drugie okrążenie było w przeciwnym kierunku dla wszystkich zespołów
- Trzecie okrążenie odbywało się w wybranym przez zespół kierunku

Po każdym okrążeniu każdy zespół otrzymywał punkty ze względu na zajęte miejsce na podstawie czasu. Nasz zespół zajął w końcowym rankingu 2 miejsce. Co ciekawe udało nam się zrobić największy progres – w pierwszym okrążeniu mieliśmy najdłuższy czas okrążenia, a ostatnie okrążenie udało nam się skończyć na pierwszym miejscu.

# Transporter

Zadanie Transporter było rozszerzeniem zadania Linefollower i polegało na tym, aby robot podążał za czarną ścieżką do czasu natrafienia na wybrany przez zespół kolor. Po natrafieniu na kolor (u nas kolor niebieski) należy skręcić w kolorową ścieżkę, następnie podążać za czarną ścieżką do czasu natrafienia na płytkę wybranego koloru. Na płytce wybranego koloru należy podnieść pakunek, następnie zawrócić i kontynuować podróż wzdłuż czarnej do linii do czasu natrafienia na wybrany kolor (u nas kolor niebieski), gdzie należy skręcić i kontynuować ruch wzdłuż czarnej ścieżki do czasu natrafienia na inny wybrany kolor (u nas kolor zielony), gdzie robot powinien skręcić, następnie poruszać się wzdłuż czarnej ścieżki do czasu natrafienia na płytkę wybranego koloru, gdzie powinien odstawić pakunek i zakończyć działanie.

Pakunek, który należało przenieść, każda drużyna konstruowała samodzielnie. Pakunek skonstruowany przez nas wyglądał następująco:



*Obraz 2 Pakunek użyty w zadaniu Transporter*

# Dostępne materiały — czujniki i silniki

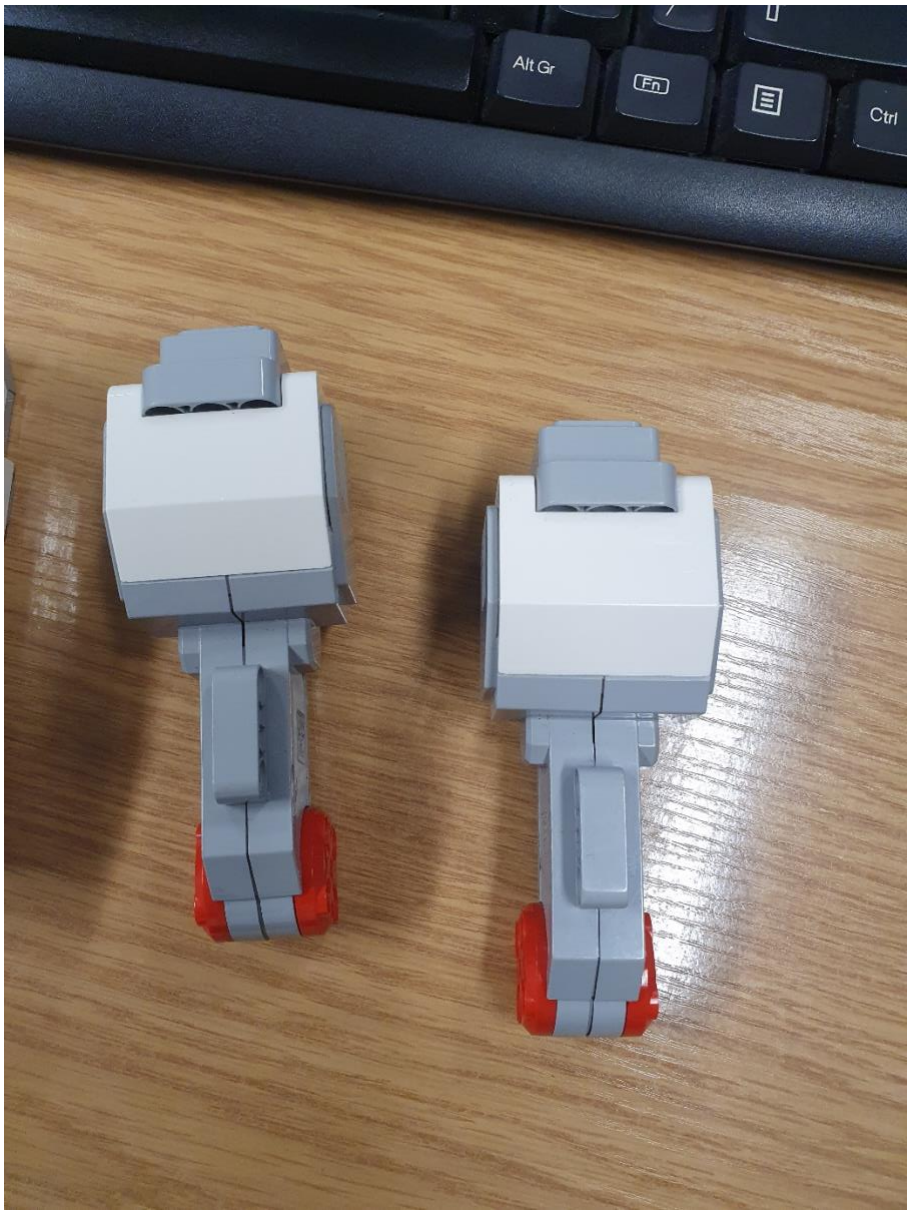
Pełny opis składowych robota: Dokumentacja wszystkich części EV3

## EV3 Brick


*Obraz 3 EV3 Brick*

Układ komputerowy odpowiedzialny za przekazywanie zasilania, interpretacje i wykonywanie kodu programu. Umożliwia podłączenie się do niego przez protokół SSH. Ekran służy do wyświetlania menu robota lub/oraz do wyświetlania komend (poprzez funkcję print) przy wykonywaniu programu.

# Large Motor




*Obraz 4 Large Motor*

Silniki elektryczne zasilające koła robota - dostępne w ilości podwójnej, jak na załączonym obrazku.

# Touch Sensor

*Obraz 5 Touch Sensor*

Czujnik, który pozwala np. włączyć nasz program poprzez naciśnięcie przycisku. Czujnik nie został wykorzystany w naszym projekcie robota.

# Ultrasonic Sensor

*Obraz 6 Ultrasonic Sensor*

Czujnik odległości, który wykorzystywany był głównie do lokalizacji oraz mierzenia odległości pakunka.

# Color Sensor

*Obraz 7 Color Sensor*

Czujnik kolorów, wykorzystywany do sprawdzania koloru, obsługujący RGB. Dostępny w ilości sztuk 2, tak jak na załączonym obrazku.

# Medium Motor

*Obraz 8 Medium Motor*

Silnik elektryczny, który umożliwiał podnoszenie pakunku.

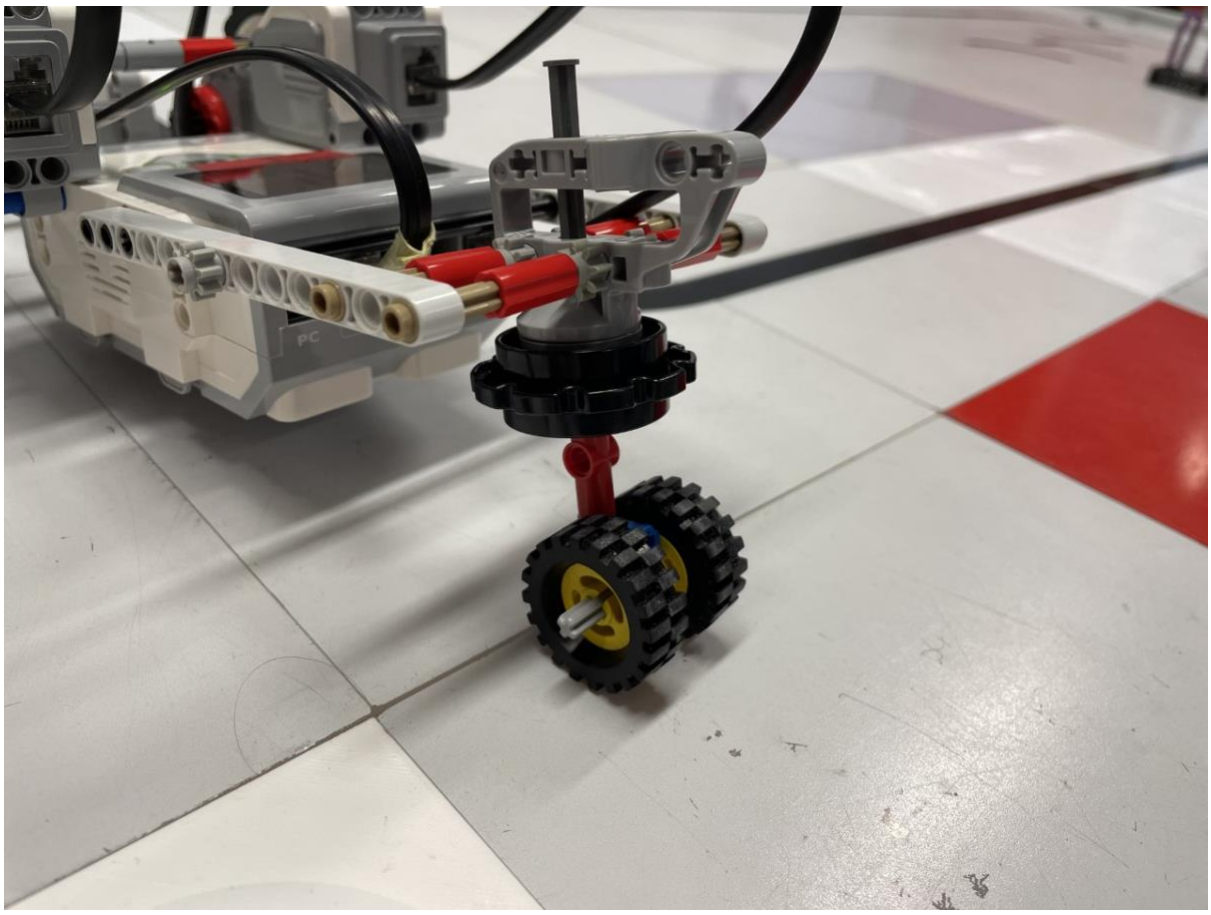
# Budowa robota

## Robot do zadania Linefollower

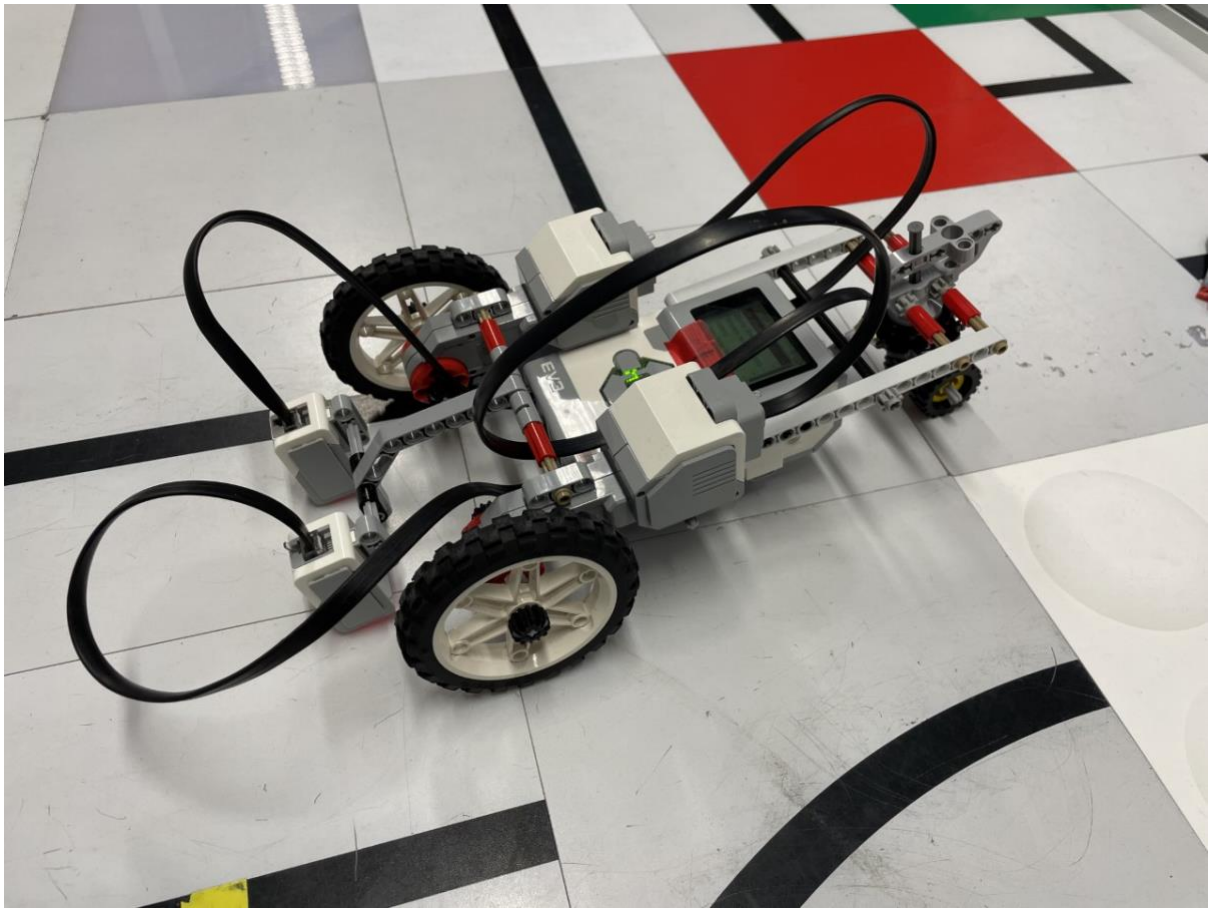
Zbudowany przez nas robot do zadania Linefollower składał się z:

- EV3 Brick – wykorzystywany do przetwarzania kodu programu oraz zasilania reszty komponentów w energię elektryczną
- 2x Large Motor – wykorzystywane do napędzania kół przednich
- 2x Color Sensor – wykorzystywane do utrzymywania się na właściwej ścieżce i detekcji odpowiednich kolorów poprzez RGB

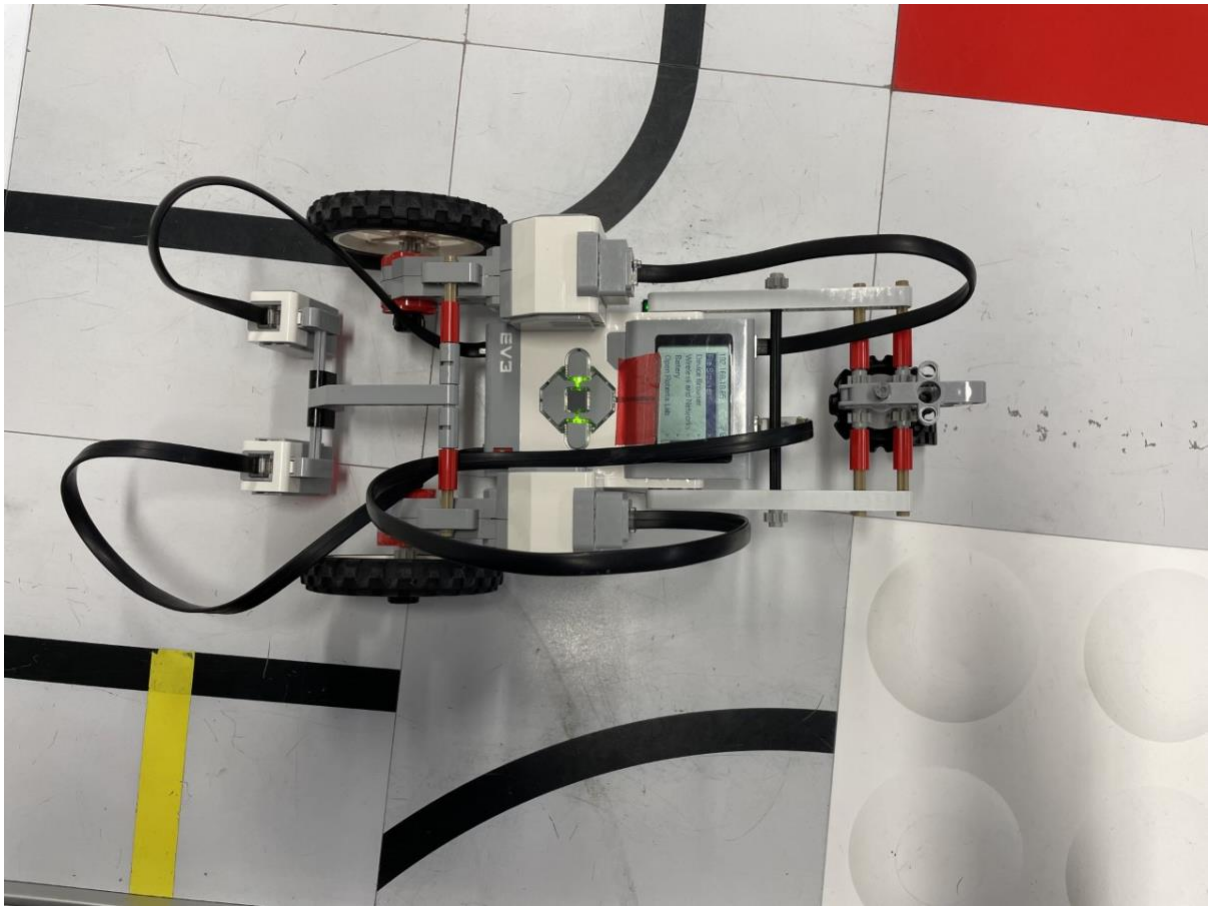
- Dwa koła przednie, podłączone do silników elektrycznych Large Motor
- Jedno obrotowe koło tylne konstrukcji własnej:



*Obraz 9 Obrotowe koło tylne*



*Obraz 10 Widok skonstruowanego robota do zadania Linefollower od boku*

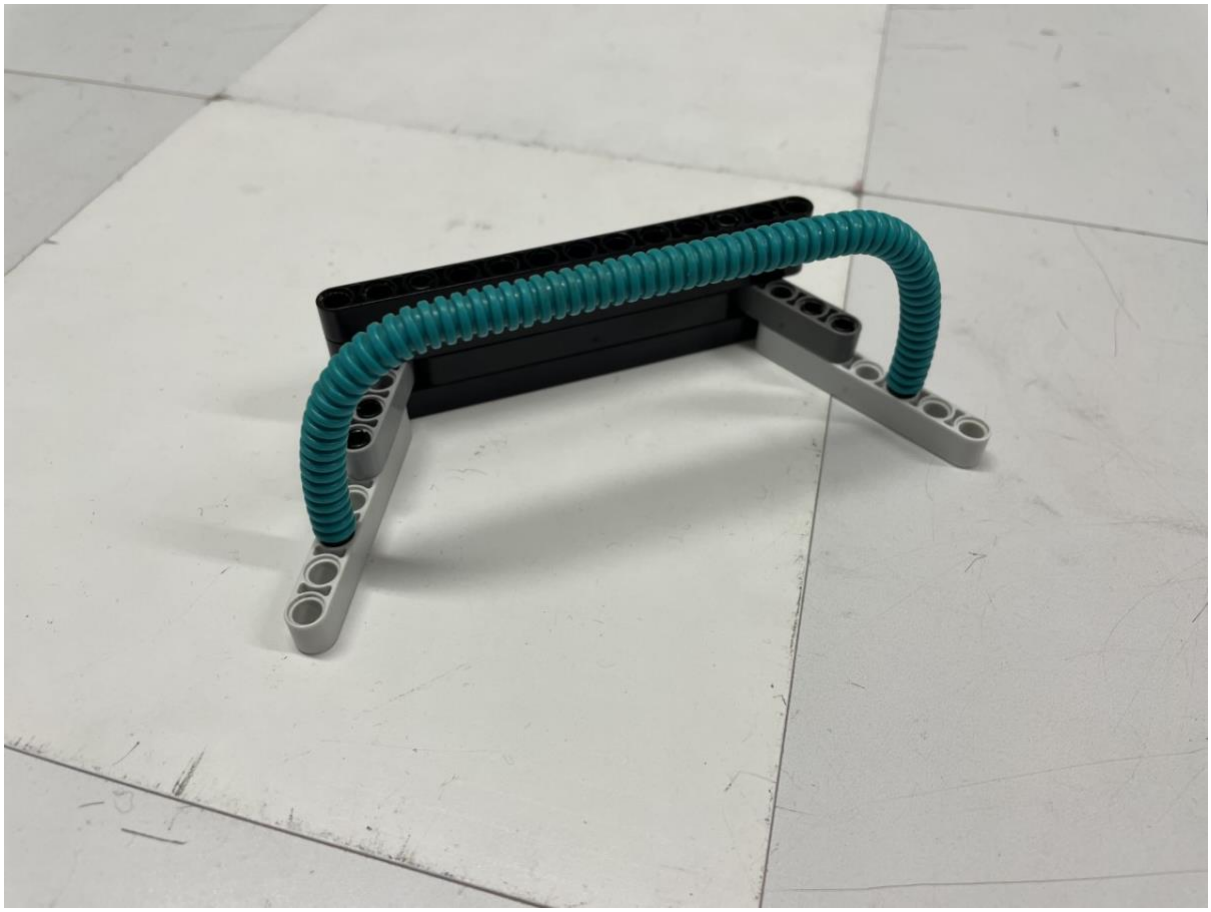



*Obraz 11 Widok skonstruowanego robota do zadania Linefollower z lotu ptaka*

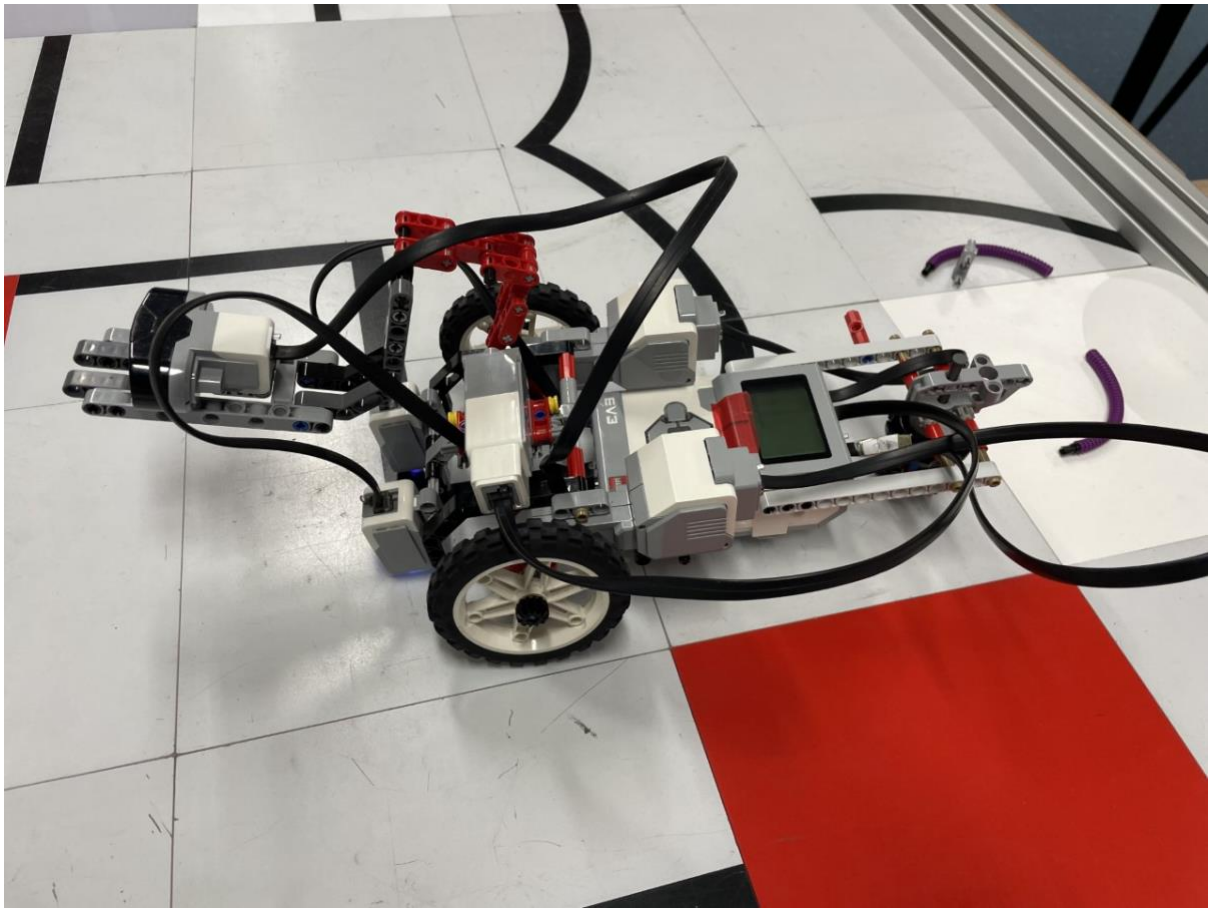
## Robot do zadania Transporter

Zbudowany przez nas robot do zadania Transporter składał się z:

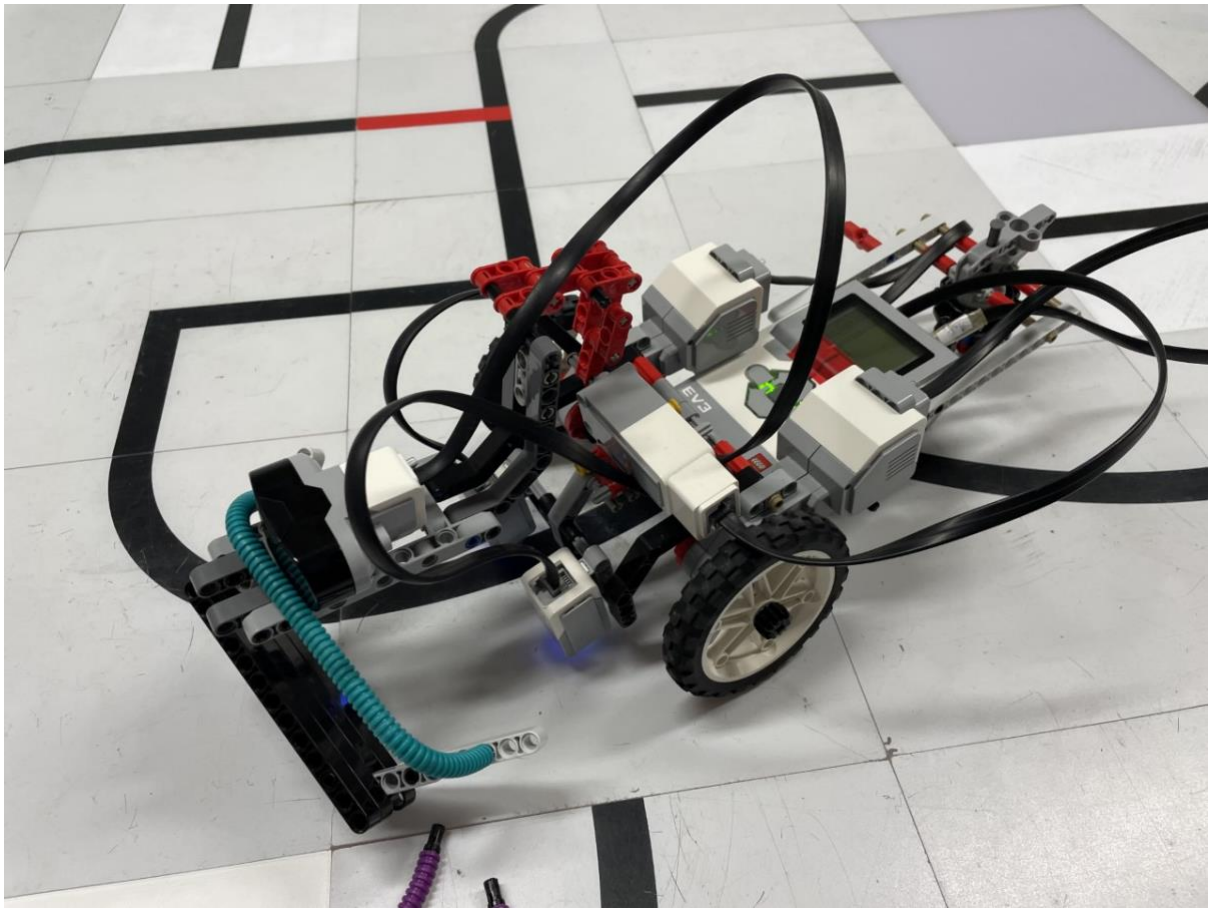
- EV3 Brick – wykorzystywany do przetwarzania kodu programu oraz zasilania reszty komponentów w energię elektryczną
- 2x Large Motor – wykorzystywane do napędzania kół przednich
- 2x Color Sensor – wykorzystywane do utrzymywania się na właściwej ścieżce i detekcji odpowiednich kolorów poprzez RGB
- 1x Medium Motor – wykorzystywany do podnoszenia oraz opadania ramienia z pakunkiem
- 1x Ultrasonic Tensor – wykorzystywany do lokalizowania paczki i jej odległości od przodu robota, co umożliwiało podjazd w taki sposób, aby umożliwić podniesienie pakunku
- Dwa koła przednie, podłączone do silników elektrycznych Large Motor
- Jedno obrotowe koło tylne konstrukcji własnej, pokazane wyżej



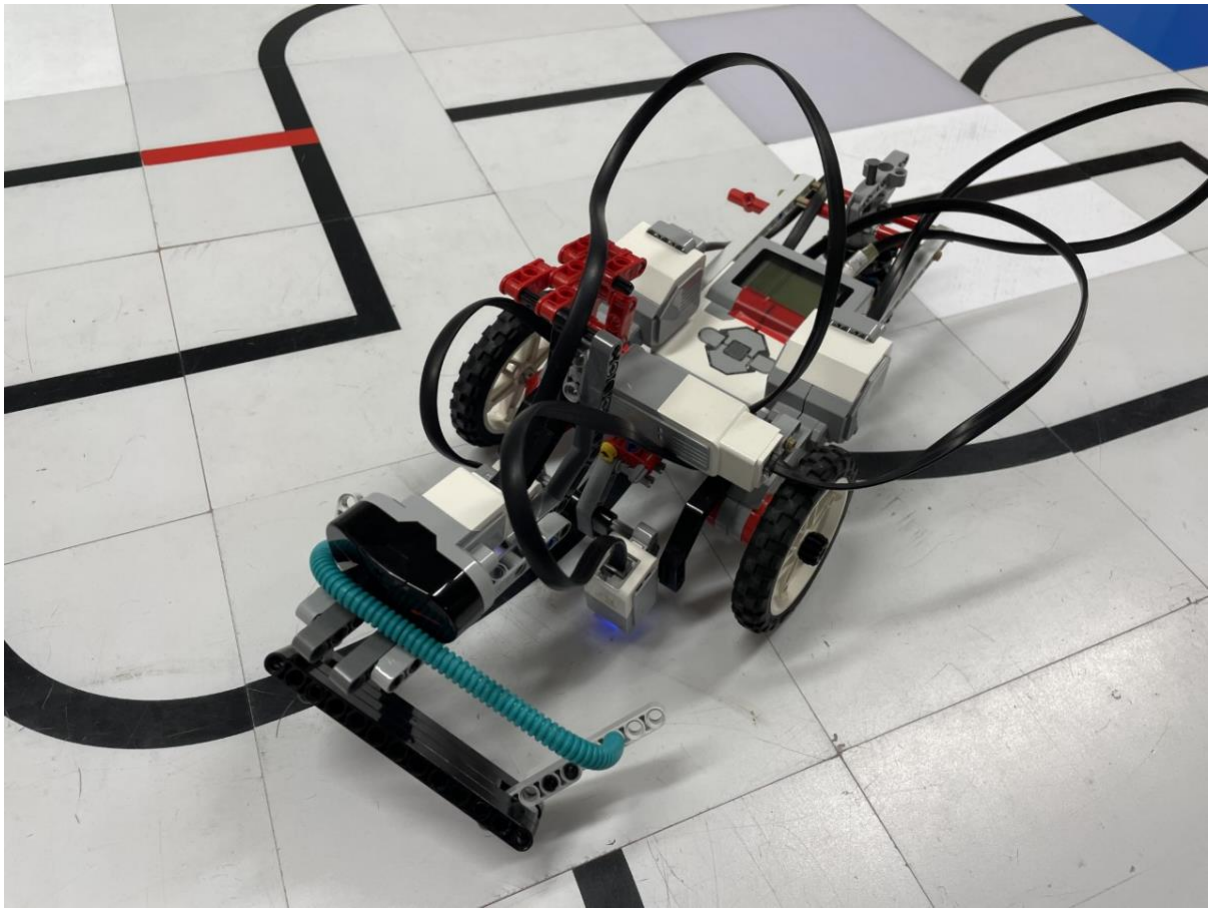
*Obraz 12 Pakunek użyty w zadaniu Transporter*



*Obraz 13 Widok skonstruowanego robota do zadania Transporter od boku*



*Obraz 14 Robot podczas wykonywania zadania Transporter z pakunkiem na pętli prowadzącej do skrzyżowania*



*Obraz 15 Robot podczas wykonywania zadania Transporter z pakunkiem na prostej*

# Wytworzone oprogramowanie

## Kod do zadania Linefollower

```
def follow_line():

    def adjust_steering():
        light_difference = color_sensor_l.reflected_light_intensity -
color_sensor_r.reflected_light_intensity
        if light_difference > light_threshold:
            return 100
        if light_difference < -light_threshold:
            return -100
        if light_difference < 15:
            return 0
        return light_difference*STEERING_SCALAR
```

```
def adjust_speed(steering):
    if abs(steering) > SPEED_THRESHOLD:
        return SpeedPercent(turn_speed)
    else:
        return SpeedPercent(go_forward_speed)

while True:
    steering = adjust_steering()
    speed = adjust_speed(steering)
    steering_drive.on(steering, speed)
```

Kod użyty w zadaniu Linefollower znajduje się w pliku *linefollower.py*. Powyżej wklejony został jego fragment. Widzimy tutaj główną pętlę sterującą promieniem skrętu oraz prędkością robota. Sterowanie było oparte o *reflected_light_intensity* zwracane przez oba sensory. W związku z tym, że mieliśmy podążać za czarną linią, a dookoła mogliśmy mieć tylko biały kolor, było to najlepsze rozwiązanie (początkowo próbowaliśmy sterować robotem poprzez odczytywanie koloru z listy kolorów, a więc sprawdzaliśmy, czy sensor zwróci *White* czy *Black*, ale działało to dużo gorzej). Na podstawie różnicy *reflected_light_intensity* zwróconego przez oba sensory decydujemy o skręcie:

- Dla różnicy < 15 jedziemy prosto
- Dla różnicy przkraczającej zadany *light_threshold* skręcamy maksymalnie w lewo lub w prawo
- Dla pozostałych przypadków wykonujemy lekki skręt w lewo lub w prawo (promień skrętu jest zależny od tego jak duża jest różnica między intensywnością światła zwróconą przez sensory).

## Kod wykorzystany w trakcie ostatniego (najlepszego) okrążenia wyścigu

Kod, który użyliśmy w trakcie ostatniego i najlepszego okrążenia znajduje się w pliku o nazwie *linefollower_f1.py*. Od zwykłego linefollowera wersja ta różni się następującymi rzeczami:

- zdecydowanie zwiększyliśmy prędkość (pozwolił nam na to brak ostrych zakrętów)
- dla niskiej różnicy *reflected_light_intensity* dla obu sensorów lekko skręcaliśmy zamiast jechać prosto. Powodowało to, że robot dziwnie jechał na prostych (bo niepotrzebnie skręcał), ale w rezultacie mógł on przez dłuższy czas jechać z większą prędkością (nie wpadał w „turbulencje") oraz czasami udawało mu się idealnie wejść w zakręt tak, że prawie nie musiał na nim zwalniać.

## Kod do zadania Transporter

Kod do tego zadania znajduje się w pliku *colorfollower.py*. Jest on zdecydowanie dłuższy i bardziej skomplikowany od kodu użytego w poprzednim zadaniu. Oczywiście mógłby on zostać uproszczony (wiele fragmentów się w nim powtarza). Priorytetem było dla nas natomiast zaprogramowanie robota, który jest w stanie bezproblemowo ukończyć swoje zadanie (niestety nie starczyło czasu na refaktoring).

Na początku zadanie próbowaliśmy wykonać wykorzystując funkcję zwracającą kolory rejestrowane przez sensory. Niestety czasami czarny był rozpoznawany jako zielony, a zielony jako niebieski. Próbowaliśmy tego rozwiązania także z innymi kolorami, takimi jak czerwony czy żółty, ale efekt był taki sam. Zmusiło nas to do wykorzystania funkcji zwracającej wartości RGB. Tutaj nastąpiło przełomowe odkrycie, ponieważ okazało się, że nasze sensory działają bardzo różnie. Na poniższym fragmencie kodu widać wartości RGB zwracane przez sensory dla różnych kolorów:

```
# define colors rgb lists
RED_L = [130, 14, 17]
RED_R = [210, 17, 35]
BLUE_L = [17, 59, 90]
BLUE_R = [25, 57, 180]
GREEN_L = [12, 75, 32]
GREEN_R = [22, 70, 63]
BLACK_L = [16, 22, 19]
BLACK_R = [22, 22, 38]
WHITE_L = [100, 145, 110]
WHITE_R = [180, 175, 255]
```

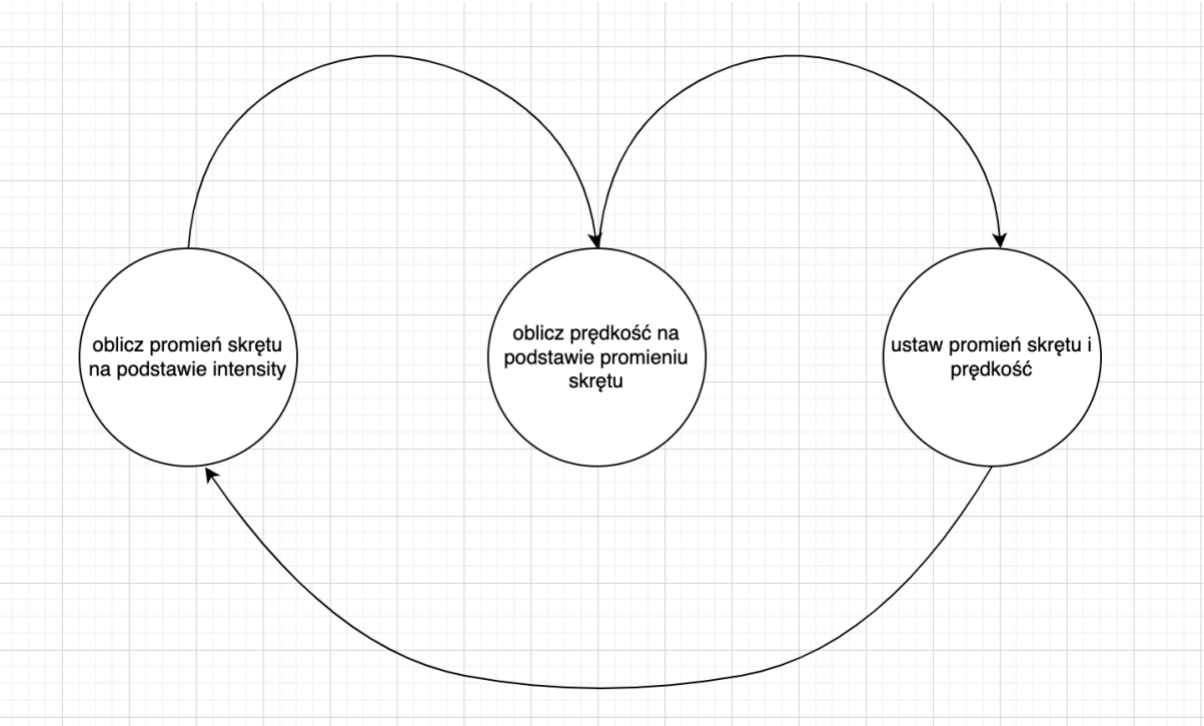
Jak widać, w niektórych przypadkach różnice są bardzo duże. Taka kalibracja zdecydowanie ułatwiła nam sterowanie robotem. Następnie na podstawie zwróconych przez sensory wartości RGB liczymy odległość euklidesową od skalibrowanych wartości różnych kolorów. Najmniejsza odległość euklidesowa wskazuje nam na jaki kolor w rzeczywistości najprawdopodobniej wskazuje sensor.

Cała reszta sterowania w tym zadaniu polega na skręcaniu w odpowiednim kierunku, jeśli jakiś kolor został zwrócony ileś razy z rzędu.

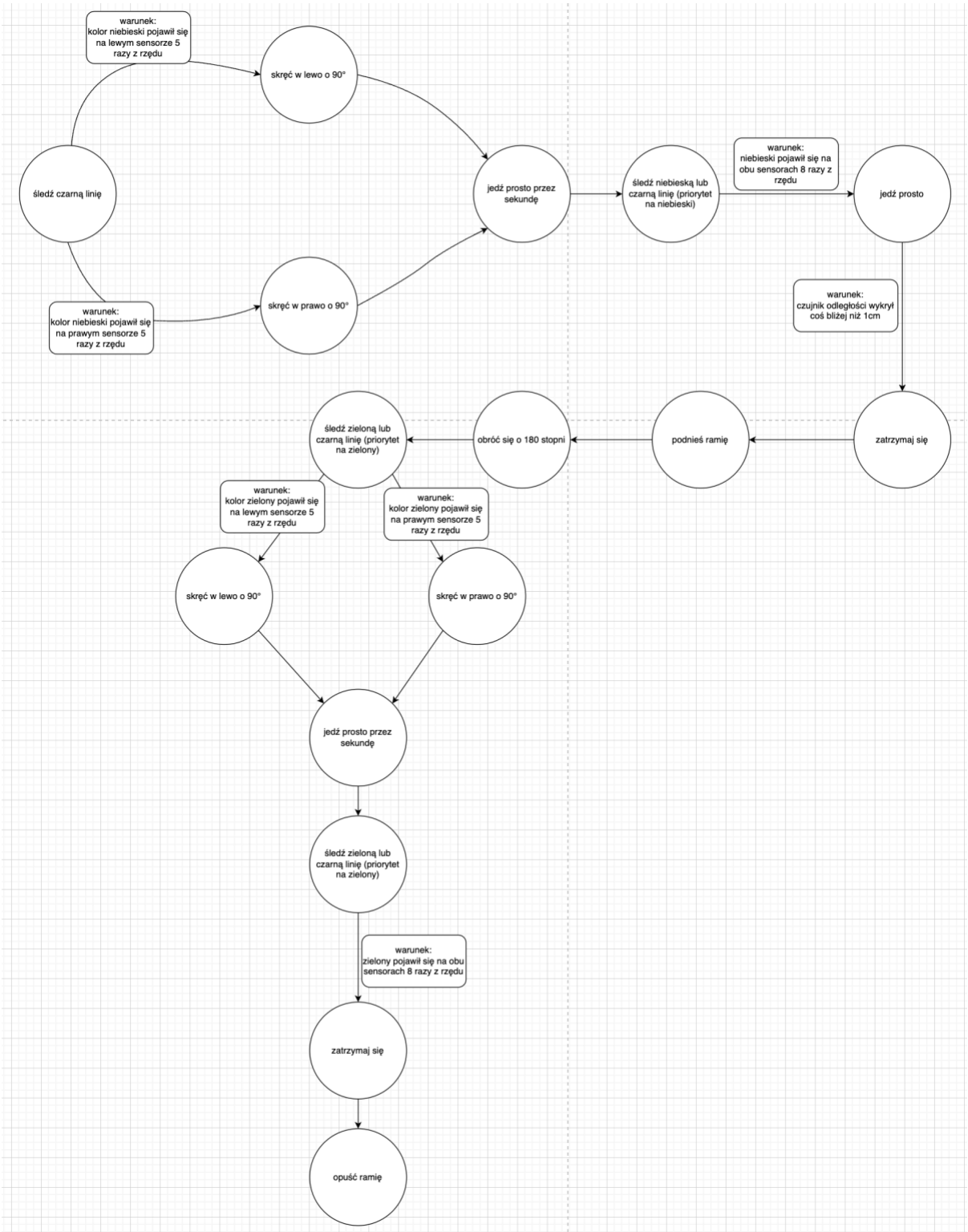
Warto jeszcze dodać, że nieprzypadkowo wybraliśmy kolor niebieski i zielony. Są one najbardziej podobne do czarnego. Dzięki temu, robot śledzący linię zieloną będzie także śledził linię czarną, ale z priorytetem na kolor zielony (analogicznie z kolorem niebieskim). Sytuacja wyglądałaby trochę inaczej, gdybyśmy wybrali sobie kolor żółty (jest on za bardzo podobny do białego – wtedy śledzenie czarnego nie działałoby zbyt dobrze).

# Automaty do obu zadań

## Automat do zadania pierwszego (Linefollower)



# Automat do zadania drugiego (Transporter)



# Podział pracy

## Mikołaj Kita

Mikołaj Kita zajmował się głównie konstrukcją robota, wykonywaniem eksperymentów oraz pisaniem dokumentacji.

## Miłosz Łopatto

Miłosz Łopatto był głównym programistą w projekcie - zajmował się głównie wykonywaniem oprogramowania. Jest on także autorem automatów w niniejszej dokumentacji.

## Praca wspólna

Warto jednak podkreślić, że większość zadań wykonywana była wspólnie, a wszystkie decyzje dotyczące konstrukcji robota oraz użytego oprogramowania były podejmowane wspólnie. Nasza kooperacja odbywała się również poza zajęciami na terminach dodatkowych, co pokazuje nasze zaangażowanie w projekt.

# Podsumowanie

## Realizacja zadań

Udało się zrealizować oba zadania - zarówno za Linefollower, jak i Transporter otrzymaliśmy maksymalną ilość punktów. Dodatkowo zajęliśmy drugie miejsce w wyścigu. Można więc powiedzieć, że wszystkie cele zostały więc zrealizowane, ale na pewno nie udałoby się ich osiągnąć bez przychodzenia na dodatkowe terminy laboratoryjne.

## Wnioski

Zadanie Transporter okazało się trudniejsze niż początkowo zakładaliśmy. Robot często podejmował niezrozumiałe dla nas decyzję, co prowadziło do konieczności powrotu do oprogramowania oraz jego budowy i wprowadzenie zmian. Szczególnie źle radziły sobie czujniki wykrywania  kolorów – o ile z czarnym i białym nie było problemu, to pomylenie koloru niebieskiego i zielonego było częstym przypadkiem, co wymusiło na nas odpowiednią kalibrację czujników.

## Największe wyzwania

W zadaniu Linefollower największym wyzwaniem okazało się pokonywanie skrzyżowań, ponieważ pierwotnie zdarzały się sytuacje, w których robot na skrzyżowaniach skręcał zamiast poruszać się naprzód.
W zadaniu Transporter największym wyzwaniem okazała się odpowiednia kalibracja czujników. Czujniki często myliły kolory, przykładowo zielony był często mylony z

niebieskim, a niebieski z czarnym. Dodatkowo, prawy czujnik przy rozkładzie na RGB pokazywał wartość koloru niebieskiego na poziomie dwukrotnie wyższym niż czujnik lewy. Wymusiło to na nas kalibrację czujników, aby prawidłowo odbierały rzeczywiste kolory.