

LINE FOLLOWER „Burek”

Systemy Robotów Autonomicznych – Maria Krauz, Michał Pożarlik

1. Wstęp

Zawody robotów [1]

Podczas zawodów odbywa się kilka konkurencji, w których roboty mogą zmierzyć się ze sobą, są to między innymi:

- Line Follower
- Sumo
- Micromouse
- Freestyle...

W każdej konkurencji robot musi spełniać określone regulaminem wymogi, jednak przede wszystkim **robot musi być całkowicie autonomiczny**. Nie może kontaktować się z zawodnikiem podczas walk czy przejazdów, jak również być w jakikolwiek sposób zdalnie sterowany. Waże jest to, że wszystkie roboty są konstruowane – projektowane i budowane – przez zawodników. Wyjątkiem od tej reguły może być na przykład kategoria LEGO Sumo, gdzie roboty budowane są z gotowych komponentów.

1.1. Line Follower - pojazd

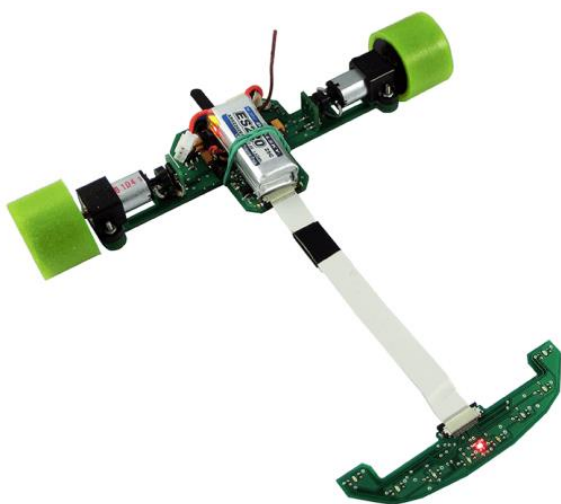
„LineFollower - (Linefollower) to roboty startujące w zawodach, na których rozgrywana jest kategoria Follow The Line (FTL). Przeważnie są to małe i szybkie konstrukcje (najszybsze Polskie roboty osiągają średnie prędkości powyżej 2,3m/s), których zadaniem jest przejechanie wyznaczonej trasy w jak najkrótszym czasie, roboty ścigają się na białym podłożu z wyznaczoną czarną trasą (lub odwrotnie).

Do zawodów dopuszczane są przeważnie roboty nieprzekraczające swoimi rozmiarami formatu kartki A4. Jest to jedna z najbardziej popularnych konkurencji w Polsce jak i na świecie. Dodatkowo od pewnego czasu rozgrywane są zawody Follow the Line Enhanced (z przeszkodami), podczas których roboty wyposażone są w czujniki odległości, dzięki którym mogą wykrywać przeszkody znajdujące się na trasie omijać je i powracać na właściwy tor.” [2]

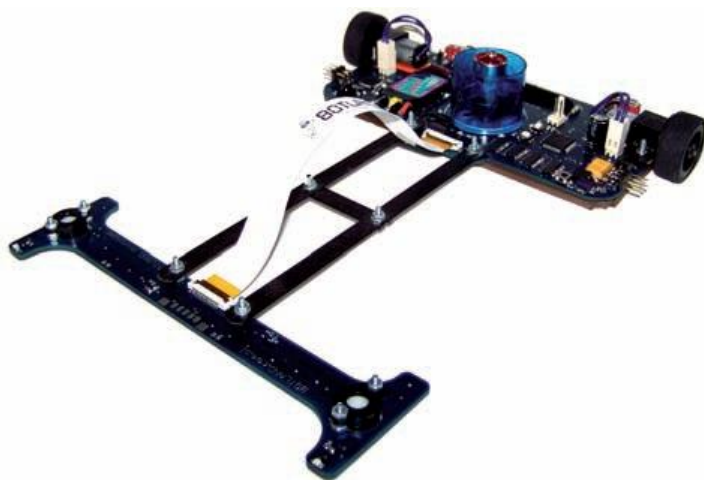
Można jednak spotkać się z pojazdami typu Line Follower, różniącymi się znacząco budową i możliwościami zastosowania.

‘Szybki’ Line Follower

Pierwszą grupę stanowią pojazdy, których zadaniem jest osiągnięcie jak najlepszych efektów czasowych przy przejazdach po wyznaczonych trasach. To te właśnie roboty można zobaczyć na zawodach. Są przystosowane do szybkiej jazdy poprzez zastosowanie odpowiednich silników, kół wykonanych ze specjalnego materiału, czy wreszcie wysunięcie czujników na przód umożliwiające reakcję z pewnym nieuniknionym opóźnieniem. Takie pojazdy standardowo ‘mieszczą się’ na kartce rozmiaru A4. Na rysunkach 1-1 oraz 1-2 przedstawiono przykładowe roboty tego typu.



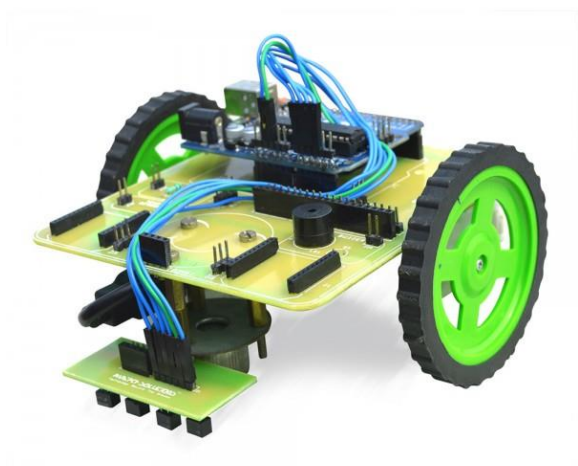
Rysunek 1-1: 'Szybki' Line Follower [6]



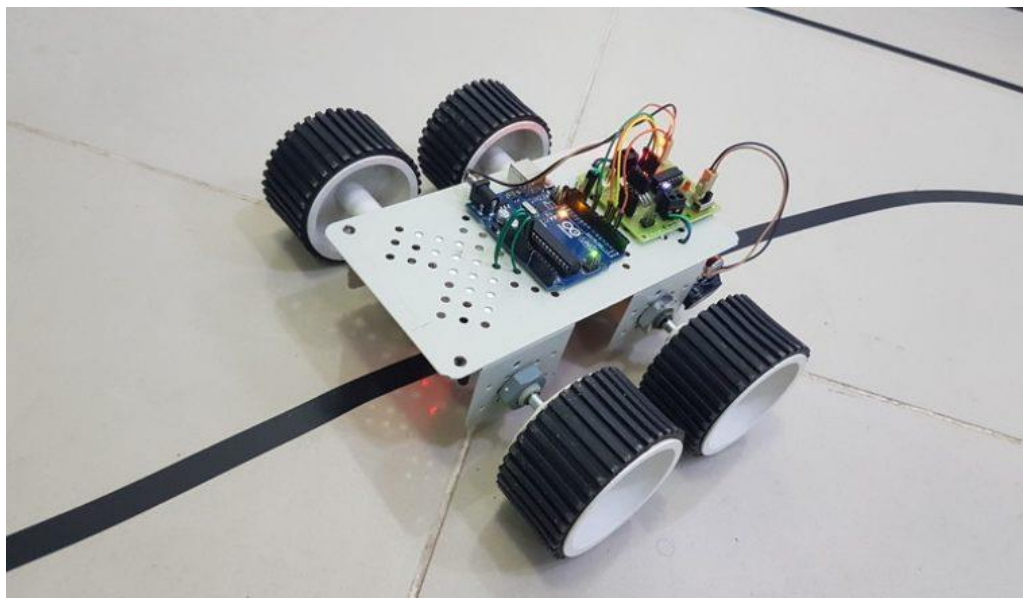
Rysunek 1-2: 'Szybki' Line Follower [7]

‘Mały’ Line Follower

Kolejną grupą są mełe pojazdy charakteryzujące się zdecydowanie mniejszymi prędkościami, poruszające się jednak dokładnie po wyznaczonej linii. Ich zadaniem jest przejechanie nawet bardzo zawiłych tras z jak największą ‘precyzją’. Ich czujniki są wysunięte na przód nieznacznie lub znajdują się pod centralną częścią pojazdu. Przykłady takich konstrukcji przedstawiono na rysunkach 1-3 oraz 1-4.



Rysunek 1-3: 'Mały' Line Follower [8]



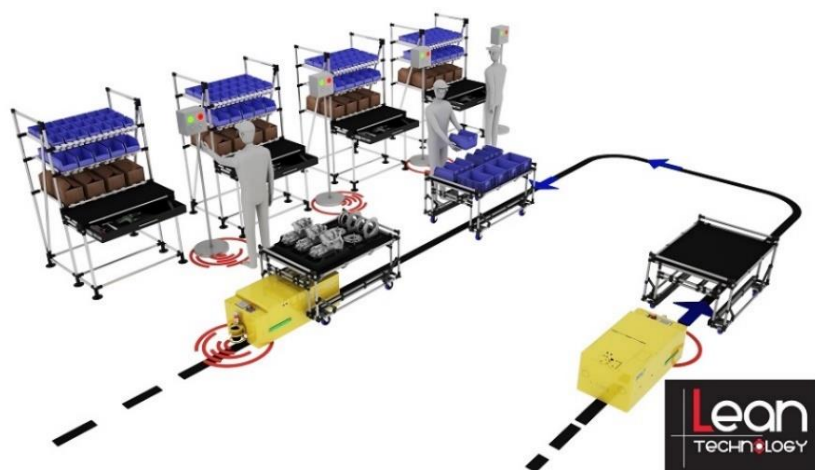
Rysunek 1-4: 'Mały' Line Follower [9]

‘Pożyteczny’ Line Follower – AGV (ang. Automated Guided Vehicles)

Tę grupę pojazdów charakteryzują stosunkowo małe prędkości poruszania się, wynikające bezpośrednio z ich zastosowań. Nie przypominają one szybkich robotów z pierwszej grupy, natomiast pojazdy z drugiej grupy mogą stanowić dla nich pewnego rodzaju prototypy lub modele. Pojazd taki nie ma wysuniętych na przód czujników, jego masa jest zdecydowanie większa, a zastosowane części (między innymi rama, koła, czy same silniki) mają zapewnić możliwość jego długotrwałej eksploatacji.

Line Follower jest robotem stosowanym w przemyśle.

Tego typu roboty mogą być używane do zbierania narzędzi, czy przenoszenia części poruszając się po wyznaczonej ścieżce. Zobrazowanie zastosowania pojazdów śledzących linię przedstawiono na rysunku 1-5, natomiast przykład przemysłowego pojazdu AGV na rysunku 1-6.



Rysunek 1-5: Przykład zastosowania robota avg [10]



Rysunek 1-6: Przemysłowy pojazd typu AGV [11]

1.2. Line Follower - konkurencja

Cel

W kategorii Line Follower można wyróżnić dwa typy tras. W kategorii Line Follower Standard trasa jest kręta, natomiast dla kategorii Line Follower Drag jest to prosta linia. Niezależnie jednak od kategorii głównym celem jest przejechanie trasy w jak najszybszym czasie. Przykładowa trasa pokazana została na rysunku 1-7.



Rysunek 1-7: Przykładowy tor [12]

Wymogi kategorii Line Follower

Najistotniejszymi punktami regulaminu są:

„3. Wymiary Robota nie mogą być większe niż:

- a. 297mm - dotyczy całkowitej długości,
- b. 210mm - dotyczy całkowitej szerokości,
- c. 210mm - dotyczy całkowitej wysokości.

4. Robot musi poruszać się w sposób autonomiczny.

5. Komunikacja z robotem w trakcie Próby jest zabroniona. Dozwolone jest zdalne startowanie oraz zatrzymywanie Robota.” [5]

2. Cel projektu i koncepcja realizacji

Cel projektu

Celem projektu jest stworzenie autonomicznego robota spełniającego warunki zawarte w regulaminie konkursu dla kategorii Line Follower [3]. Docelowo przewidywany jest udział w kolejnej edycji konkursu „ROBO~motion”.

Projekt zakłada zbudowanie robota oraz stworzenie oprogramowania sterującego. Pierwszym krokiem do osiągnięcia postawionego celu było przeprowadzenie przeglądu dostępnych technologii oraz wybór części. Kolejnym krokiem było zbudowanie robota oraz implementacja algorytmów sterowania. Stworzony został również tor, który posłużył do testowania robota oraz skalibrowania wymaganych parametrów.

Sekrety szybkiego Line Followera na podstawie doświadczenia wielokrotnego zwycięzcy w konkurencji Line Follower Standard [4]

- Odpowiednie silniki

Silniki napędzające konstrukcję powinny zapewniać jak największy moment obrotowy przy maksymalnej prędkości obrotowej rzędu 1000 – 3000 RPM. Odpowiedni moment obrotowy jest kluczowy podczas wykonywania przez robota szybkich zwrotów i reakcji na nagłe zmiany linii.

- Proporcje kół

Koła, których średnica jest 1.5 – 2 razy mniejsza od szerokości, najlepiej sprawdzają się podczas gwałtownych zakrętów. Aby robot nie tracił przyczepności ważne jest pokrycie felg odpowiednim materiałem, opony mogą być na przykład odlane z silikonu lub poliuretanu. Dobrze sprawdzają się również koła składające się z felg wykonanych z poliamidu oraz opon Mini-Z [5].

- Liczba i ułożenie czujników

Liczba czujników nie powinna być zbyt duża, jednak gdy jest zbyt mała zwiększa się podatność na błędy. Dobrze sprawdza się na przykład osiem czujników ułożonych w półkołu. Duża liczba czujników zwiększa złożoność obliczeniową oraz powoduje obciążenie przodu konstrukcji, przez co zwiększa jej bezwładność.

- Konstrukcja mechaniczna

Istotna jest waga robota, oraz rozłożenie masy. Robot musi być lekki – redukcja masy wpływa na przyspieszenie oraz prędkość, natomiast środek ciężkości powinien znajdować się jak najbliżej kół – wpływa to na poprawę stabilności.

- Zasilanie

Wybierając akumulator należy mieć na uwadze fakt, że jest to dodatkowe obciążenie całej konstrukcji. Ponieważ przejazd robota trwa zwykle kilkadziesiąt sekund, wystarczający jest mały akumulator (na przykład o pojemności 100-250 mAh). Należy jednak pamiętać, że silniki podczas nagłej zmiany obrotów mogą pobierać duży prąd, a akumulator musi sprostać temu zadaniu.

- Oprogramowanie

Algorytm sterujący nie powinien opierać się na instrukcjach warunkowych, ale powinien zapewniać szybki wybór kierunku jazdy, umożliwiając płynne poruszanie się pojazdu po torze.

- Zmiana ustawień

Zarówno podczas testów, jak i przejazdów turniejowych często istnieje konieczność przeprowadzenia zmian w konfiguracji. Aby przystosować robota do aktualnego wyzwania zmienna może być maksymalna prędkość, prędkość w przypadku wykrycia zakrętu, tryb działania (dostosowanie wykorzystania czujników do skomplikowania trasy). Pomocne jest stworzenie aplikacji umożliwiającej zdalne zmiany konfiguracji.

- Testy

Istotne jest przeprowadzenie testów przy zmiennym oświetleniu, na różnym podłożu oraz przy wielu konfiguracjach trasy.

3. Specyfikacja techniczna – analiza i wybór części

3.1. Jednostka obliczeniowa

Jednostka obliczeniowa musi zapewnić wystarczającą moc obliczeniową dla spełnienia zadania sterowania pojazdem, a więc nie musi być ona bardzo duża. Istotne jest natomiast rozpatrzenie kwestii rozmiarów i wagi samej płytki, ponieważ rozmiar całego robota jest ograniczony, jak również wybór zbyt dużego układu negatywnie wpłynie na uzyskane parametry przejazdu. Po przeprowadzonej analizie wybrano dwa układy, które mogą znaleźć zastosowanie w tworzonym projekcie: *Arduino Uno Rev3*, *Raspberry Pi Zero W*, *Arduino Pro Micro (Leonardo)*.

	<i>Arduino Uno Rev3</i>	<i>Raspberry Pi Zero W</i>	<i>Arduino Pro Micro</i>
Taktowanie	16 MHz	1 GHz	16 MHz
Mikrokontroler / Procesor	ATmega328	Broadcom BCM2835 ARM11	ATmega324 u4
System operacyjny	-	Linux	-
Bluetooth	Nie	Tak	Nie
Wydajność prądowa	40 mA	150 mA	40 mA
Rozmiar	54.3 x 68.6 mm	30 x 65 mm	33 x 18 mm
Waga	25 g	8.8 g	

Wykorzystanie Arduino pozwoli na napisanie bardziej optymalnego i wydajnego algorytmu oraz zapewni większą kontrolę nad jednostką obliczeniową. Taktowanie jest znacznie niższe niż w przypadku Raspberry Pi, powinno być jednak wystarczające na potrzeby działania algorytmu. Raspberry Pi zapewnia środowisko w postaci systemu operacyjnego Linux, co stanowi zaletę w procesie tworzenia oprogramowania, jednak nie jest rozwiązaniem optymalnym dla takiego zastosowania. Ponieważ planowana jest zdalna komunikacja z robotem obecność modułu Bluetooth jest zaletą wykorzystania Raspberry Pi, natomiast w przypadku Arduino konieczne jest dokupienie specjalnego rozszerzenia. Maksymalny pobór prądu znacząco różni się w przypadku analizowanych układów na korzyść Arduino (dla porównania średni pobór prądu przez silniki to maksymalnie 250 mA). Rozmiary układów również znacząco się różnią, natomiast wszystkie mogą być zastosowane bez przekroczenia dopuszczalnych wymiarów robota. Ze względu na niewielki rozmiar oraz wystarczające parametry zdecydowano się na zastosowanie Arduiono Pro Micro.

3.2. Zdalne startowanie i zatrzymywanie

Chociaż komunikacja z robotem podczas próby jest zabroniona, zalecane jest jego zdalne wystartowanie oraz zatrzymanie. Zaplanowana jest również możliwość zdalnej zmiany konfiguracji pojazdu. W związku z tym konieczne jest stworzenie aplikacji umożliwiającej przeprowadzenie tych operacji, ale również posiadanie modułu Bluetooth. Ze względu na brak wbudowanego modułu Bluetooth w wybranym Arduino, dokupiono osobny dedykowany moduł (HC-05).

3.3. Silniki

Zastosowane zostały silniki z przekładnią:

- N20-BT13
- Przekładnia 10:1
- Maksymalna ilość obrotów: 2200 RPM
- Zasilanie: 3 – 9 V
- Moment obrotowy: 0.02 Nm (0.2 kg*cm)
- Średnica wału: 3mm
- Wymiary korpusu: 24 x 10 x 12 mm
- Masa: 10 g
- Pobór prądu bez obciążenia: 70 mA (dla 9V)

Sterowanie silnikami

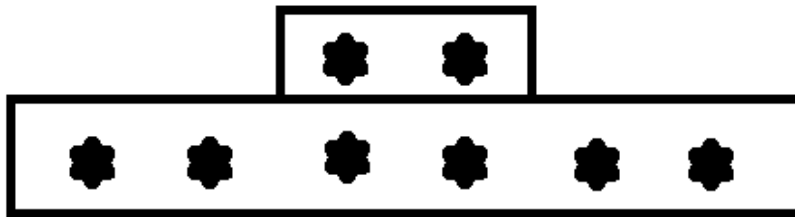
Wykorzystano dwukanałowy sterownik silników (typ: TB6612FNG).

Specyfikacja sterownika:

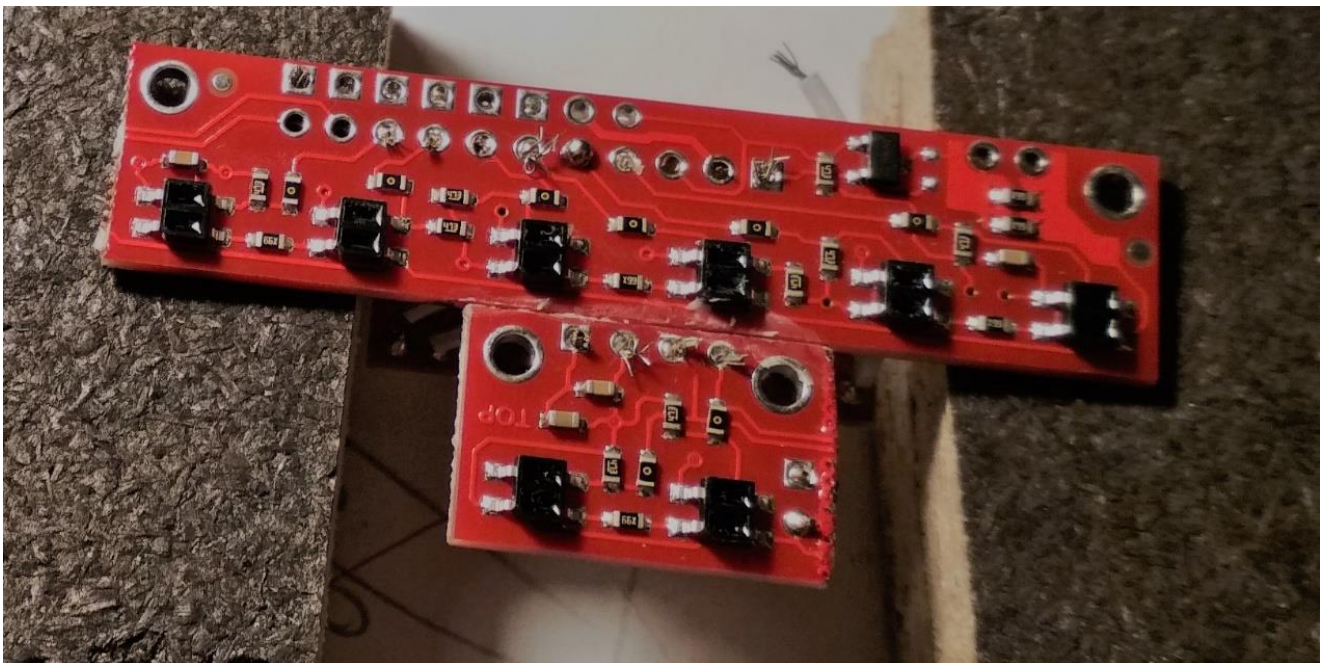
- Zasilanie silników(VMOT): od 4,5 V do 13,5 V
- Zasilanie układu logicznego (VCC): od 2,7 V do 5,5 V
- Maksymalny prąd wyjściowy: 3 A na kanał
- Ciągły prąd wyjściowy: 1 A na kanał
- Przy połączeniu obu kanałów: 2 A
- Maksymalna częstotliwość PWM: 100 kHz

3.4. Czujniki

Zdecydowano się zaprojektować własny, niestandardowy plan rozmieszczenia czujników. Wybrano zestaw ośmiu analogowych czujników odbiciowych (QTR-8A). Do zebrania sygnału z czujników wykorzystano multiplexer analogowo-cyfrowy 74HC4051 (SparkFun). Czujniki zostały rozmieszczone zgodnie ze schematem przedstawionym na rysunku 3-1. Taki układ pozwoli na wprowadzenie pewnych optymalizacji w algorytmie, które pomogą przyspieszyć podjęcie decyzji o korekcie trasy. Rysunek 3-2 przedstawia zdjęcie wykorzystanych czujników.



Rysunek 3-1: Schemat rozmieszczenia czujników



Rysunek 3-2: Wykorzystane czujniki

3.5. Koła

Wybrano koła dedykowane do tego typu zastosowań: Solarbotics RW2i. Koła składają się z dwóch elementów: felg oraz opon wykonanych z wysokiej jakości gumy, co pozwala na zachowanie przyczepności podczas pokonywania zakrętów i przy nagłych zmianach prędkości.

Parametry kół:

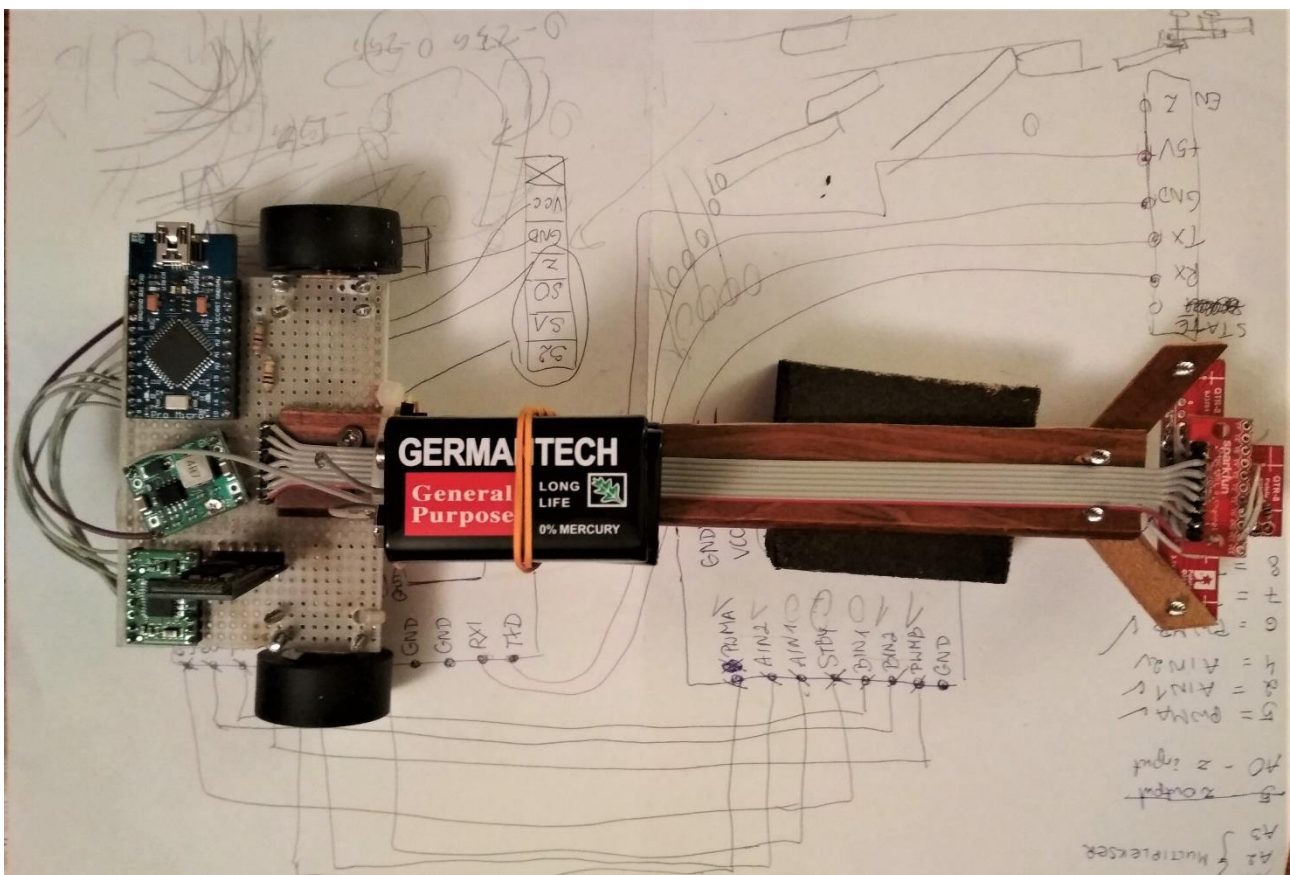
- Wymiary:
 - średnica: 31.2 mm,
 - szerokość: 13.2 mm,
 - średnica otworu: 3 mm (zgodna z wybranymi silnikami)
- Masa: 12 g
- Typ mocowania: wewnętrzne

3.6. Zasilanie

Robot zasilany jest z baterii 9V. Do zasilenia Arduino wymagane jest obniżenie napięcia (9V na 5V), które zrealizowane zostało poprzez wykorzystanie odpowiednio wyskalowanej przetwornicy.

3.7. Podwozie

Jako stelarz dla układów wykorzystana została płytką PCB, natomiast połączenie wysuniętych na przód czujników z płytką zostało zrealizowane przy wykorzystaniu cienkiej drewnianej listewki. Na rysunku 3-3 przedsięwzięte jest zdjęcie złożonego pojazdu.



Rysunek 3-3: Pojazd

4. Specyfikacja oprogramowania

4.1. Architektura

Oprogramowanie pojazdu stworzone zostało w dedykowanym środowisku Arduino IDE. Język programowania jest językiem pochodnym C. Projekt został rozdzielony na kilka plików. W głównym z nich *LineFollower.ino* znajdują się funkcja *setup* oraz główna pętla programu. Pozostałe pliki: *Configuration.ino*, *Controllers.ino*, *Motors.ino* i *Sensors.ino* zawierają implementacje poszczególnych części algorytmu.

4.2. Implementacja funkcjonalności

Configuration.ino

Zawiera między innymi definicje zmiennych wykorzystywanych w algorytmie:

```
int kProviderFront = 50;
int kProviderBack = 300;
int errorMax = 200;
int maxSpeed = 180; //max value is 255
int minSpeed = 10;
float alpha = 5;
```

Opis parametrów:

- *k* jest współczynnikiem regulatora, ze względu na różną interpretację korekty toru jazdy oraz mechanizmu skrętu, wprowadzono dwa różne współczynniki,
- *errorMax* odpowiada za zatrzymanie pojazdu w wypadku zgubienia trasy (wpływa na czas bezwładności algorytmu),
- *maxSpeed* oraz *minSpeed* mają bezpośrednie odzwierciedlenie w napięciu przekazywanym na silniki, co pośrednio przekłada się na prędkość poruszania się pojazdu,
- *alpha* jest parametrem wpływającym na szybkość powracania do prędkości maksymalnej, odpowiednio dobrany pozwala na uzyskanie efektu płynnego przejazdu.

Sensors.ino

Wykorzystano bibliotekę QTRSensors.h dostarczoną przez producenta czujników. Pierwszym etapem pracy z czujnikami jest kalibracja biblioteki.

W pliku zawarto również funkcje odpowiedzialną za kalibrację kolorów, zrealizowaną autorskim algorytmem. Taka kalibracja pozwala na ustalenie zakresów odczytów czujników w celu odpowiedniej interpretacji kolorów – rozróżnienie między planszą a linią. Kalibracja jest

wykonywana dla przednich czujników osobno od skrajnych tylnych czujników, dlatego pierwszy rząd sensorów musi znajdować się na linii. Poprzez uśrednienie dziesięciu odczytów wyliczane są wartości średnie dla bieli i czerni, na które nakładana jest procentowa poprawka umożliwiająca poprawną interpretację w różnych warunkach (oświetlenie, materiał planszy).

Biblioteka zawiera metody pozwalające na odczyt wartości z poszczególnych sensorów. Do odczytywania danych z czujników wykorzystywany jest multiplekser. Zaimplementowano metodę pomocniczą umożliwiającą dokonanie odczytu z wybranego czujnika.

Motors.ino

Na początku pracy z silnikami przeprowadzana jest ustawienie odpowiednich pinów.

Najważniejszą funkcją zawartą w pliku jest funkcja wykorzystywana do kontroli przyspieszenia dla poszczególnych kół. Realizowane jest obliczenie napięcia na podstawieadanego przez regulator przyspieszenia. Pomocnicza metoda *backToDefault()* pozwala na płynny powrót do prędkości maksymalnej (wykorzystuje parametr *alpha*) i jest wywoływana gdy pojazd znajduje się na linii. Zostały wprowadzone również warunki brzegowe dla wyliczanych prędkości.

```
void setMotorSpeeds(){

    if(accelerationMotorA == 0 && accelerationMotorB==0){
        motorASpeedFloat=backToDefault(0);
        motorBSpeedFloat=backToDefault(1);
    }else{
        motorASpeedFloat += ((float)accelerationMotorA * (float)deltaTime * 0.001);
        motorBSpeedFloat += ((float)accelerationMotorB * (float)deltaTime *0.001);
    }

    accelerationA = (motorASpeedFloat + motorBSpeedFloat)*0.5;

    if(motorASpeedFloat < minSpeed)
        motorASpeedFloat=minSpeed;
    else if(motorASpeedFloat > maxSpeed)
        motorASpeedFloat=maxSpeed;

    if(motorBSpeedFloat < minSpeed)
        motorBSpeedFloat=minSpeed;
    else if(motorBSpeedFloat > maxSpeed)
        motorBSpeedFloat=maxSpeed;

    move(1, (int)motorASpeedFloat, motorADirection);
    move(0, (int)motorBSpeedFloat, motorBDirection);
}
```

Sama kontrola napięcia realizowana jest w funkcji *move()*, polega ona na ustawieniu odpowiedniego stanu pinów. Napięcie sterowane jest pinem PWM i przyjmuje wartości od 0 do 255. Realizacja zatrzymania pojazdu zawarta jest w funkcji *stop()*, polega ona na ustawieniu na pinie STBY stanu niskiego.

Controllers.ino

W pliku znajduje się kontroler komunikacji z robotem. Możliwe jest przesyłanie za pomocą komunikacji Bluetooth konkretnych komunikatów w celu modyfikacji paramentów oraz wydawania komend:

- start i stop
- włączenie trybu debuggowania
- modyfikacja parametrów regulatora
- modyfikacja minimalnej i maksymalnej prędkości
- kalibracja stopnia przyspieszenia
- kalibracja reakcji na błędy.

Funkcja pozwala również pozwala odczytać aktualne parametry.

Umieszczono tu również funkcję *raceControl()*, w której znajduje się główny algorytm odpowiedzialny za podejmowanie decyzji o kierunku jazdy.

```
void raceControl(){
    getFrontSensorsValues();
    if(!frontSensors[0] && !frontSensors[1]){
        getBackSensorsValues();
        _e = eFunctionBack();
        if(_e == 0){
            _countError++;
            _e = _prevE;
        }else{
            _countError = 0;
            _prevE = _e;
        }
        _k = kProviderBack;
    }else{
        _countError = 0;
        _e = eFunctionFront();
        _k = kProviderFront;
    }
    if(_countError > errorMax)
        _isError = true;
    _du = _k * _e;
    accelerationMotorA = _du ;
    accelerationMotorB = -_du;
}
```


Algorytm realizuje dwie możliwości: korekta trasy, przy której wykorzystywane są tylko dwa przednie czujniki oraz mechanizm skrętu, do którego wykorzystany jest odczyt ze wszystkich czujników. Przytoczone poniżej funkcje umożliwiają odczytanie w sposób pośredni pozycje pojazdu względem trasy.

```
int eFunctionFront(){
    if(frontSensors[0] &&frontSensors[1])
        return 0;
    else if (!frontSensors[0])
        return -1;
    return 1;
}
int eFunctionBack(){
    for( int k = 0; k < 3;k++){
        if(backSensors[k])
            return 3-k;
        else if(backSensors[5-k])
            return k-3;
    }
    return 0;
}
```

LineFollower.ino

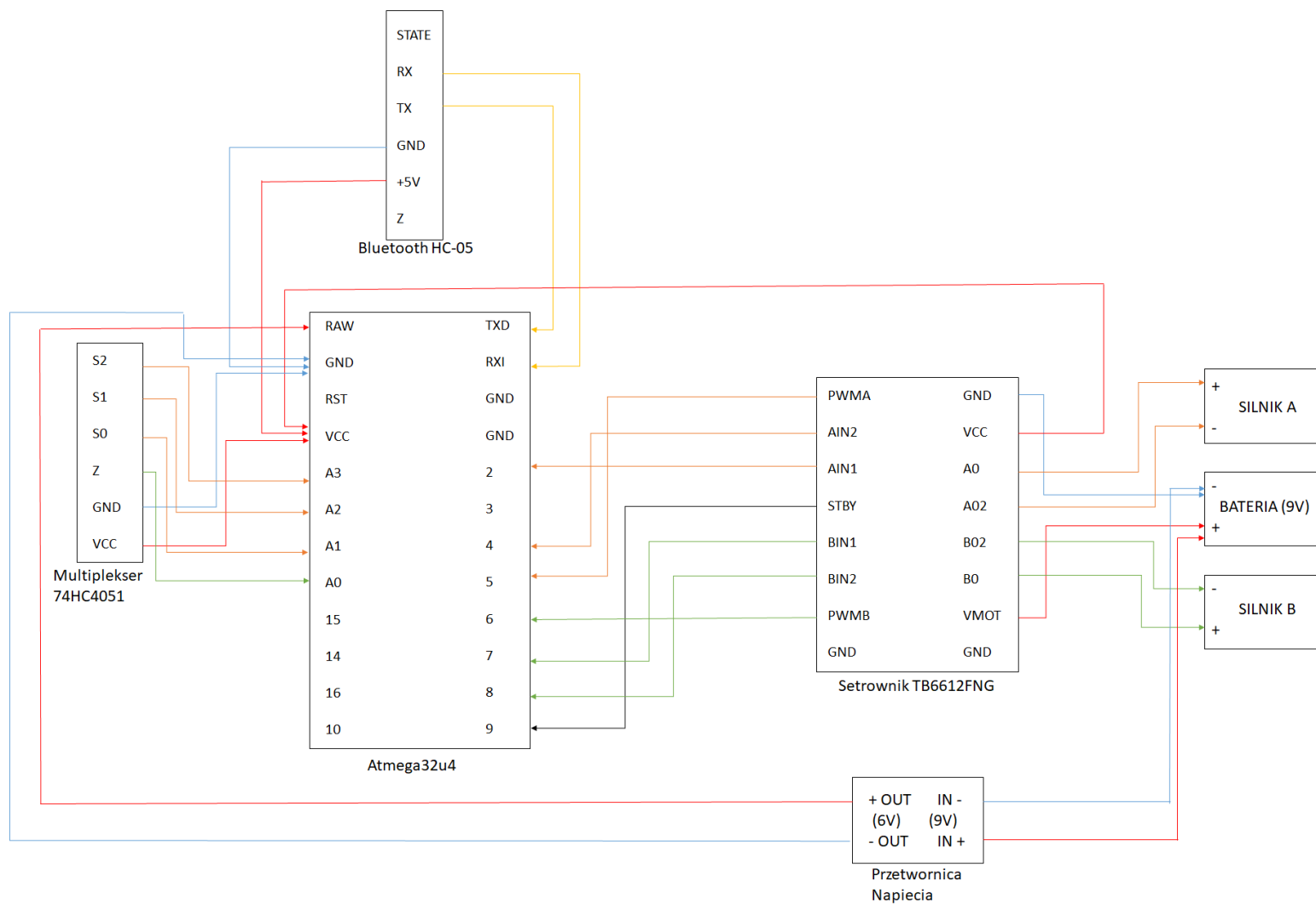
W tym pliku umieszczono funkcję *setup()*, która uruchamiana jest bezpośrednio po uruchomieniu robota. W ramach funkcji przeprowadzane jest ustawienie odpowiednich pinów dla multipleksera, sterownika i czujników oraz wstępna kalibracja czujników.

Kolejną funkcją jest *loop()*, stanowiąca główną pętlę zdarzeń. Na początku obliczana jest różnica czasowa wykorzystywana do wyznaczania prędkości na podstawie przyspieszeń. Kolejnym etapem jest sprawdzenie stanu linii komunikacyjnej. Jeżeli robot jest w trakcie przejazdu, to wykonywane są kolejne dwa etapy: podjęcie decyzji o parametrach dalszej przejazdu (*raceControl()*) oraz sprawdzenie flagi błędu. Jeżeli nie ma przeciwwskazań, wytyczne dotyczące dalszego przejazdu są aplikowane w postaci sygnału do sterownika silników. W przeciwnym wypadku, silniki są zatrzymywane.

5. Szczegóły techniczne i algorytmiczne

Połączenie układów

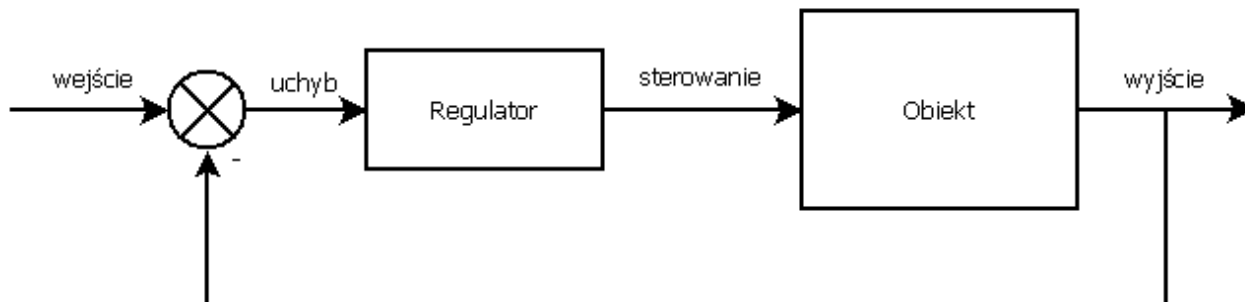
Połączenie głównych układów zostało przedstawione na rysunku 5-1.



Rysunek 5-1: Schemat połączeń układów

Charakterystyka regulatora

Głównym elementem odpowiedzialnym za sterowanie pojazdem jest część algorytmu będąca regulatorem. Schemat regulatora zaprezentowano na rysunku 5-2.



Rysunek 5-2: Schemat regulatora

Standardowo na wejściu algorytmu podawana jest zadana wartość, która w analizowanym przypadku stanowi zadane położenie poszczególnych czujników na linii lub na planszy, oraz informacja zwrotna układu, czyli odczyty zebrane z poszczególnych czujników (po wprowadzeniu uprzednio zadanej korekty). Następnym krokiem jest wyliczenie uchybu. Wyznaczany jest on z wykorzystaniem wag przypisanych poszczególnym czujnikom, które odzwierciedlają ich położenie względem linii. Uchyb stanowi bezpośrednie wejście regulatora. Wyjściem regulatora jest przyspieszenie dla poszczególnych silników, na podstawie którego wyliczana jest prędkość w postaci napięcia, które podawane jest na silnik. Wartość sterująca wyliczana jest jako iloczyn uchybu i parametru k . Takie podejście gwarantuje uzyskanie tym większej korekty, im dalej od linii wyjechał pojazd – wpływ uchybu. Jednak to właśnie parametr k jest tutaj najbardziej istotny, a zarazem najtrudniejszy do wyznaczenia, ponieważ jego wartość musi zostać wyznaczona eksperymentalnie. Jedną z możliwości jest zadanie jednej, stałej wartości dla regulatora, co przy zmiennych parametrach przejazdu nie jest rozwiązaniem optymalnym. Innym rozwiązaniem może być wyznaczenie funkcji $k(V)$, która pozwoli uwzględnić wpływ prędkości w mechanizmie sterowania.




W stworzonym projekcie wprowadzono rozdzielenie sterowania: w momencie gdy uzyskany zostanie odczyt linii na przednich czujnikach prowadzona jest drobna korekta, dla której wartość współczynnika k jest znacznie mniejsza niż w drugim przypadku, gdy przednie czujniki „zgubią” linię i trzeba wprowadzić większą korektę na podstawie odczytów z drugiego rzędu czujników. Takie rozwiązanie pozwala praktycznie wyeliminować sytuacje, w której przód pojazdu będzie „odbijał” raz na prawą, raz na lewą stronę linii w sposób gwałtowny. Wprowadza natomiast efekt płynnego manewrowania.

W przypadku braku odczytów linii na wszystkich czujnikach następuje zmiana działania algorytmu i zamiast standardowej procedury podawania odczytów czujników na sprzężeniu zwrotnym układu i wyliczania uchybu, zapamiętywany jest ostatni odczyt wskazujący linię. Pozwala to na kontynuację skrętu w poprawnym kierunku, a w momencie uzyskania kolejnych odczytów linii – powrót do standardowej procedury sterowania.

Podjmowanie decyzji

Dla zobrazowania działania algorytmu przedstawiono kilka możliwych sytuacji, które robot napotyka podczas pokonywania przejazdu. Trasa przedstawiona jest za pomocą niebieskich prostokątów, a odczyty czujników pokolorowane zostały zgodnie z legendą przedstawioną na rysunku 5-3.

Pierwszym etapem działania algorytmu jest odczytanie wartości na dwóch przednich czujnikach. Wystarczy, że jeden z nich znajduje się na śledzonej linii, aby robot podjął decyzję o korekcie trasy.

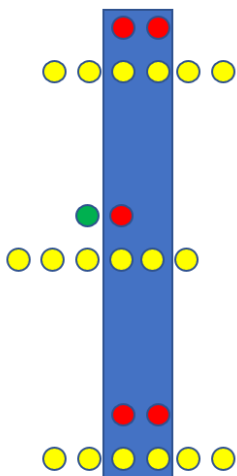
-  Brak wykrytej linii
-  Wykryta linia
-  Brak odczytu

Rysunek 5-3:Legenda

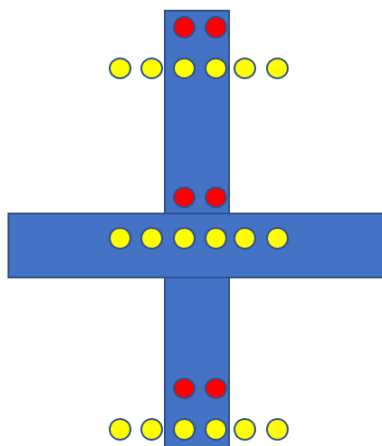
Wyróżnane są trzy możliwe rozwiązania etapu korekty:

- Korekta pozycji w lewo (jeśli lewy czujnik znajduje się na linii),
- Korekta pozycji w prawo (jeśli prawy czujnik znajduje się na linii),

- Płynne wyrównanie prędkości do wartości początkowej (jeśli oba czujniki znajdują się na linii – jest to faza przyspieszania).



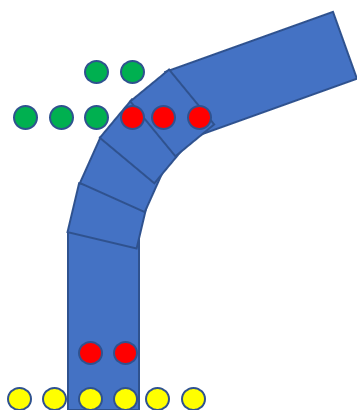
Rysunek 5-4: Korekta trasy



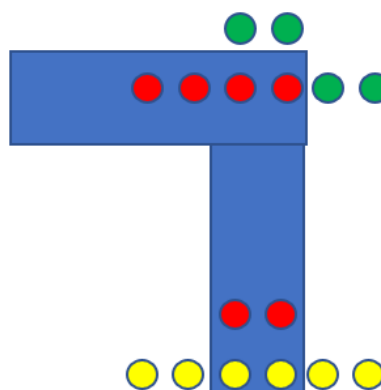
Rysunek 5-5: Skrzyżowanie

Faza korekty nie wymaga pobrania informacji z tylnych czujników, co przekłada się na szybsze podejmowanie decyzji, gdyż pominięte jest oczekiwanie zebranie odpowiedzi wszystkich sensorów. Skrzyżowania (Rys 5-4) są rozwiązywane przez decyzje podejmowane w fazie korekty. W teorii podczas przejazdu przez skrzyżowanie robot powinien ciągle odczytywać linie na przednich sensorach, w związku z czym powinien przejechać zgodnie z wyznaczoną trasą.

Jeżeli robot nie będzie w stanie przejść do fazy korekty, przechodzi do fazy skrętu. W jej pierwszym etapie pobierane są dane z tylnych czujników, a następnie na podstawie wyliczeń regulatora robot skręca z proporcjonalną prędkością do pokonywanego zakrętu.

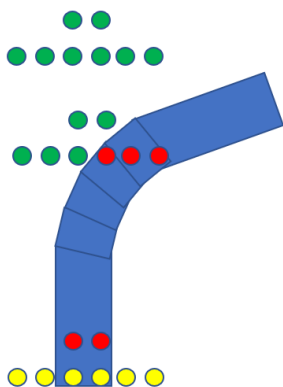


Rysunek 5-6: Zakręt w prawo



Rysunek 5-7: Ostry zakręt w lewo

Oba przedstawione zakręty (rysunki 5-6 oraz 5-7) są wykrywane przez algorytm jako ostre zakręty w prawo i w lewo, gdyż linia wykrywana jest na skrajnych czujnikach, czyli na maksymalnym wychyleniu od osi robota. W związku z czym regulator wyznaczy maksymalne przyspieszenie/spowolnienie dla obu kół, aby dokonać szybkiej korekty.



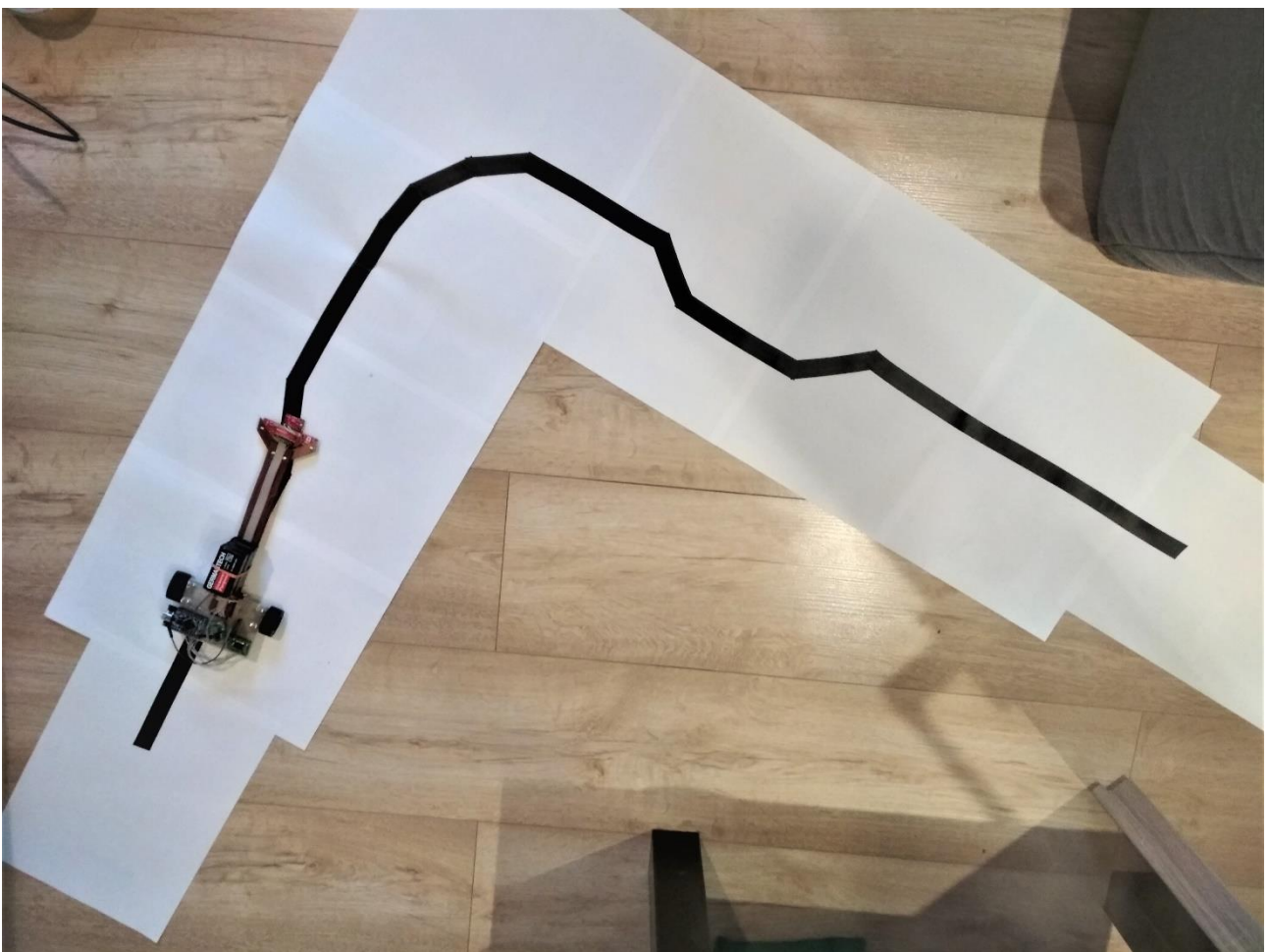
Rysunek 5-8: Ostry zakręt w prawo

Algorytm rozpatruje również sytuacje, gdy wszystkie czujniki nie są w stanie wyznaczyć pozycji linii (rysunek 5-8). Robot w następującej sytuacji kontynuuje skręt zgodnie z ostatnim odczytanym stanem linii, czyli dla przedstawionej sytuacji kontynuowane będzie skręcanie z maksymalną prędkością w prawo. Dodatkowo robot zlicza pustę odczyty, gdy otrzymana wartość przekroczy dopuszczalne normy (parametr określony przez użytkownika), algorytm decyduje o odcięciu zasilania od silników, co powoduje zatrzymaniem się pojazdu.

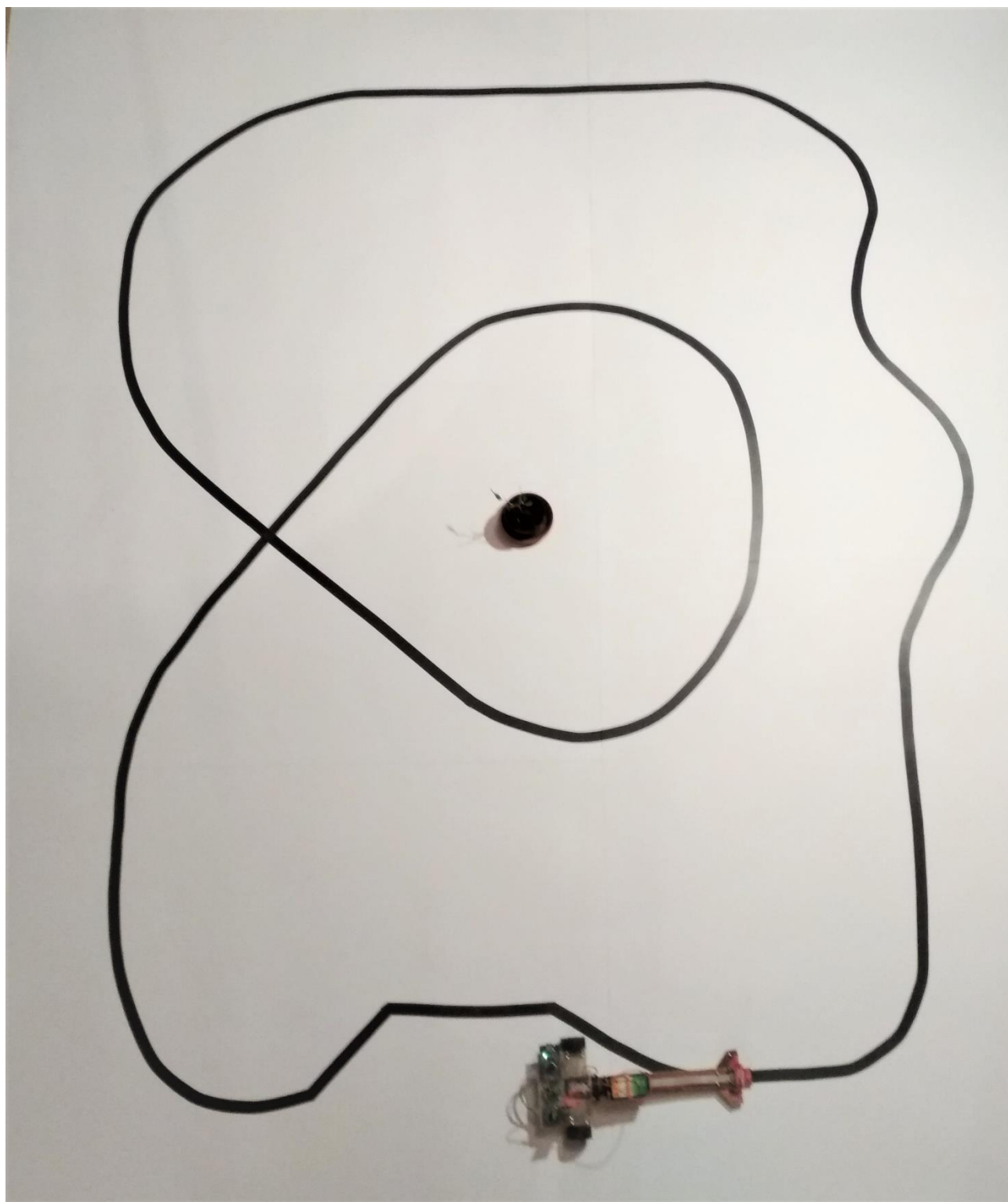
6. Testy

Przygotowanie toru

Do stworzenia toru wykorzystano biały bristol. Rzeczywiste plansze mogą być jednak wykonane z różnych materiałów, takich jak płyty meblowe MDF/HDF, czy białe ‘banery’ (wyklejenie lub wydruk linii na materiale, z którego wykonane są np wielkoformatowe reklamy). Niezależnie jednak od rodzaju podłoża, linię (w tym wypadku czarną) należy wykleić taśmą izolacyjną, która mając odpowiednie wymiary, sprawdza się idealnie do tego zastosowania. Rysunek 6-1 przedstawia zdjęcie pierwszego toru testowego, natomiast 6-2 bardziej skomplikowany tor umożliwiający przeprowadzenie testów niezbędnych do ustawienia parametrów regulatora, ograniczeń prędkości i wykrywania ‘sytuacji awaryjnych’.



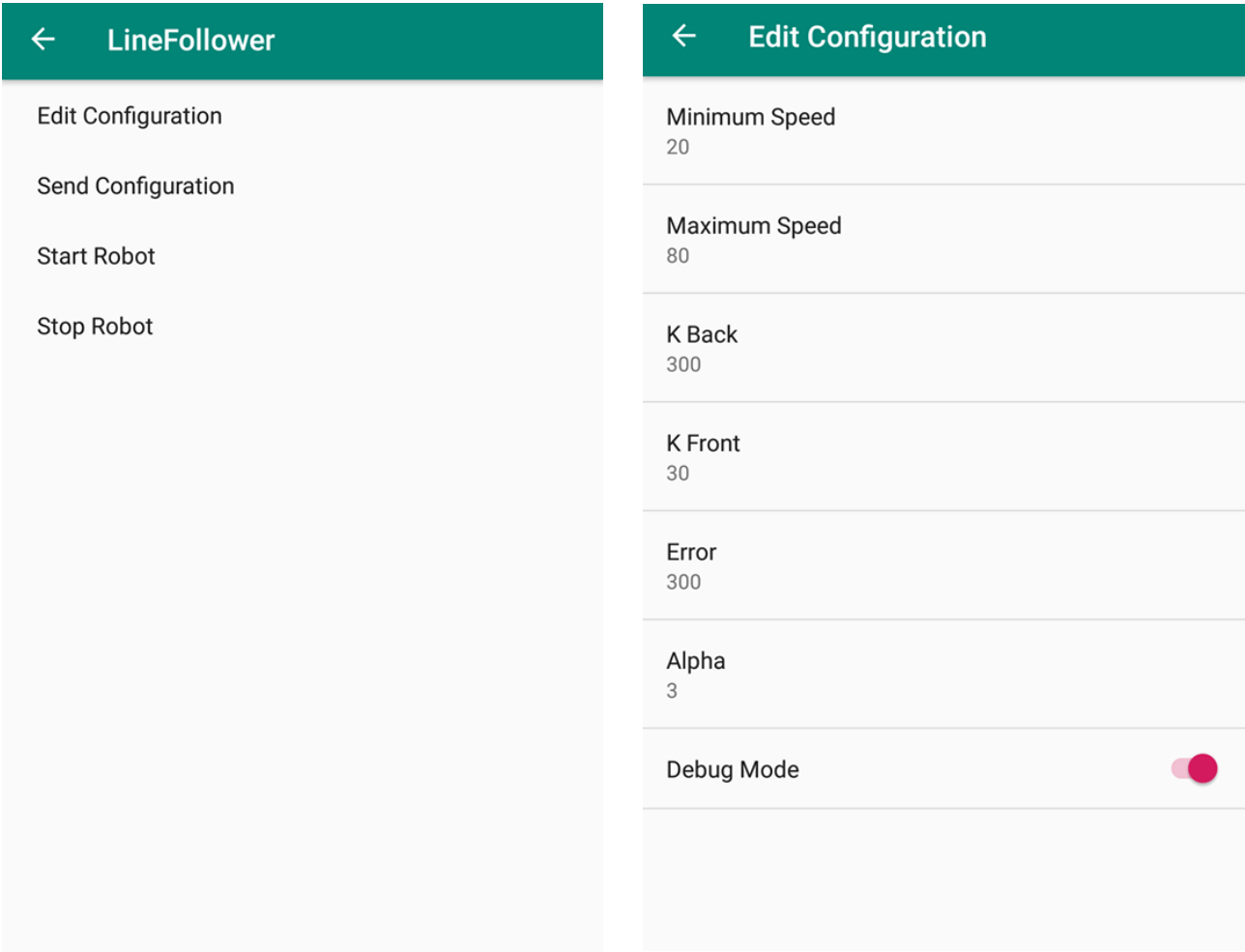
Rysunek 6-1: Pierwszy tor



Rysunek 6-2: Bardziej skomplikowany tor

Przygotowanie aplikacji

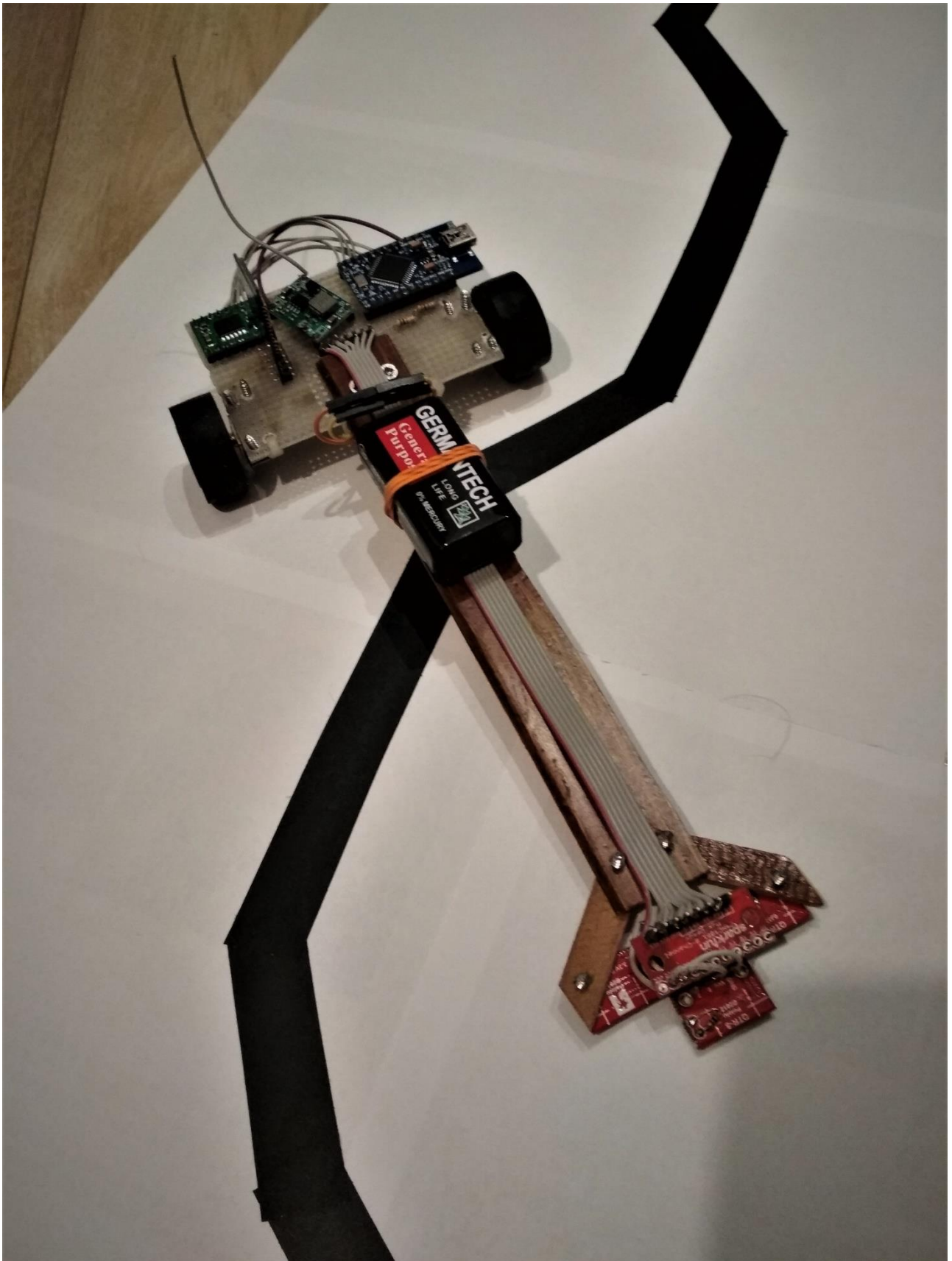
Podczas testowych przejazdów dobierano odpowiednio wartości poszczególnych współczynników. Aby przeprowadzać to w sposób efektywny, stworzona została dedykowana aplikacja umożliwiająca zdalne startowanie i zatrzymywanie robota, jak również zmianę parametrów. Ekran aplikacji znajdują się na rysunku 5-3.



Rysunek 5-3: Aplikacja

7. Podsumowanie

- W ramach projektu przeprowadzono przegląd dostępnych technologii, zaprojektowano i zbudowano konstrukcję robota oraz stworzono autorskie algorytmy sterujące robotem.
- Konstrukcja autonomicznego robota spełnia wymagania określone regulaminem zawodów, które odzwierciedlają standardowe założenia dotyczące budowy tego typu pojazdów.
- Napędem robota są dwa silniki, a sterowanie odbywa się na zasadzie różnicy prędkości obrotowej obu kół. Wysoka przyczepność została zapewniona poprzez odpowiedni dobór kształtu oraz materiału, z jakiego wykonane zostały opony. Jednostką sterującą jest Arduino Pro Micro (ATmega324 u4).
- Zaimplementowano algorytm umożliwiający autonomiczne poruszanie się pojazdu po linii. Istotną częścią algorytmu jest sposób kalibracji odczytów z czujników, w celu umożliwienia uniwersalnego rozróżniania linii i planszy. Sterowanie oparte jest na regulatorze PD – analizowane są stany poszczególnych czujników oraz przypisanych im wag, na tej podstawie wyznaczana jest korekta, która w postaci zmiany napięcia podawana jest do sterownika silników. W przypadku wyjazdu poza trasę zapamiętywane są ostatnie odczyty, co umożliwia powrót robota na linię, jednak ze względów bezpieczeństwa zbyt duża liczba odczytów niewykrywających linii prowadzi do zatrzymania się robota.
- Podczas przejazdu prędkość robota zmienia się w zależności od aktualnej sytuacji na trasie (nieznaczne obniżenie prędkości przy drobnych korektach trasy, szybszy spadek prędkości w przypadku wykrycia zakrętu oraz stopniowy powrót do prędkości maksymalnej po wymaganej korekcie położenia względem linii), a zaimplementowane algorytmy pozwalają na uzyskanie efektu ‘płynnej’ jazdy.
- Stworzona aplikacja pozwala na łatwą konfigurację robota, może posłużyć zarówno do dalszych testów i prac nad algorytmem, jak również do modyfikacji poszczególnych parametrów na zawodach podczas przejazdów próbnych.
- **W ramach projektu samodzielnie stworzono cały projekt pojazdu, połączono poszczególne komponenty i układy elektroniczne, zbudowano konstrukcję oraz zaimplementowano algorytmy sterujące pojazdem i aplikację mobilną umożliwiającą konfigurację i zdalne startowanie i zatrzymywanie robota.**



Rysunek 7-1: Line Follower "Burek"

8. Bibliografia:

- [1] <http://www.robomotion.pl/konkurencje.htm>, 19.12.2018
- [2] <https://leksykon.forbot.pl/LineFollower,4.htm>, 19.12.2018
- [3] http://www.robomotion.pl/Regulaminy_PURT_lf.pdf, Regulamin zawodów ROBO~motion, 5.10.2018
- [4] <https://forbot.pl/blog/poznaj-8-sekretow-szybkiego-linefollowera-id5482>, Sekrety szybkiego Line Followera, 5.10.2018
- [5] „Line Follower – autonomiczny robot śledzący trasę”, B. Derkacz, S. Mońka
- [6] <https://forbot.pl/blog/kurs-budowy-robotow-line-follower-czyli-bolid-f1-id19363>, 14.12.2018
- [7] https://www.google.com/imgres?imgurl=http%3A%2F%2Fwww.par.pl%2Fvar%2Faol%2Fstorage%2Fimages%2Fmedia%2Fpar%2F01_12%2Ffm%2F4.jpg%2F79864-1-pol-PL%2F4.jpg.jpg&imgrefurl=http%3A%2F%2Fwww.par.pl%2FForum-mlodych%2FStudenckie-Kolo-Naukowe-Inzynierii-Mechatronicznej%2Fline-follower-autonomiczny-robot-sledzacy-trase&docid=9RE6euvZdFhR6M&tbnid=re8Q_qzVI88zCM%3A&vet=1&w=487&h=320&bih=706&biw=1536&ved=2ahUKEwjdpvOlaDfAhWStIsKHeySAYgQxiAoAXoECAEQEg&iact=c&ictx=1, 14.12.2018
- [8] <https://www.pantechsolutions.net/line-follower-robot>, 14.12.2018
- [9] <https://www.skyfilabs.com/project-ideas/line-follower-with-arm-robot>, 14.12.2018
- [10] <http://lean-technology.pl/wozki-samojezdne-agv/>, 14.12.2018
- [11] <https://automatykab2b.pl/temat-miesiaca/48213-nowoczesne-roboty-przejmujac-coraz-wiecej-zadan>, 14.12.2018
- [12] https://bionik.ia.pw.edu.pl/projects/ftl_board/, 14.12.2018