

Robot typu line follower zbudowany
i zaprogramowany w oparciu o zestaw Lego
Mindstorms. Wstęp do Robotyki, sem. 15Z

Adam Kowalewski, Maciej Kłos

28 stycznia 2016

Spis treści

1	Motywacja i ogólna koncepcja	2
2	Konstrukcja robota	3
3	Algorytm śledzenia linii.	6
3.1	Zasada działania algorytmu	6
4	Algorytm omijania przeszkód.	9
4.1	Zasada działa algorytmu	9
4.2	Dobór nastaw regulatora	10
5	Podsumowanie. Wady i zalety.	11

Rozdział 1

Motywacja i ogólna koncepcja

Naszym zadaniem było utworzenie robota, którego głównym celem jest podążanie za czarną linią na białym tle, oraz który potrafi wykryć przeszkodę w postaci prostopadłościanu. Całość miała zostać wykonana w oparciu o udostępnione elementy zestawu *Lego Mindstorms*.

Dziedzina taka jak *Follow the Line* jest bardzo popularną konkurencją turniejową, a co za tym idzie powstał już szereg rozmaitych algorytmów sterowania robota, czy też zasady konstrukcji. Głównym zadaniem robotów jest jak najszybsze pokonanie trasy.

Zawody laboratoryjne składały się z dwóch etapów:

1. Przejechanie po trasie o podwyższonym stopniu skomplikowania wraz z ominięciem przeszkody w celu zaprezentowania jakości algorytmu sterowania. Na tym etapie nie był naliczany czas.
2. Przejechanie po trasie znacznie łatwiejszej, o większej ilości odcinków prostych. Tutaj naliczany był czas.

Wobec tego, cel jaki sobie postawiliśmy, był następujący:

- Stworzyć robota, który przede wszystkim płynnym ruchem śledzi linię,
- Robot miał zapewnić możliwość ominienia przeszkody z obu stron
- Robot miał omijać przeszkodę niezależnie od jej wymiarów, długości i szerokości

To, co udało się nam osiągnąć zostanie przedstawione w następnych rozdziałach.

Rozdział 2

Konstrukcja robota

Zastosowane czujniki

Do poprawnego wykrywania czarnej linii na białym tle potrzebny jest co najmniej jeden czujnik koloru, czy też czujnik światła odbitego. Zastosowaliśmy oba czujniki. Oba zapewniają odświeżanie z częstotliwością 1kHz, co jest bardzo przydatne zwłaszcza w przypadku szybkich robotów. Czujniki umieściliśmy w odległości około 5mm od podłogi na wysięgniku, by nieco oddalić czujniki od osi kół. Wskaźniki są maksymalnie blisko siebie, a co za tym idzie - oba pokrywają linię.

Do wykrywania przeszkody zastosowaliśmy czujnik podczerwieni. Umieściliśmy go możliwie nisko, by wykrywać nawet bardzo niskie przeszkody.

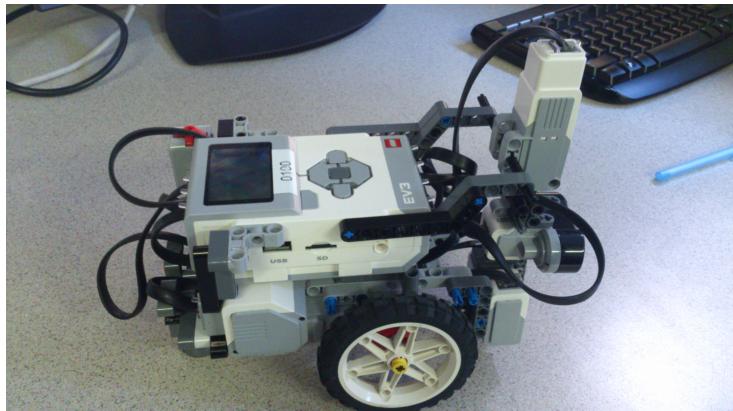
Dodatkowo przydatny okazał się tzw. *push-button* jako źródło sygnału uruchamiającego robota oraz zatrzymującego go w wybranym momencie.

Elementy wykonawcze

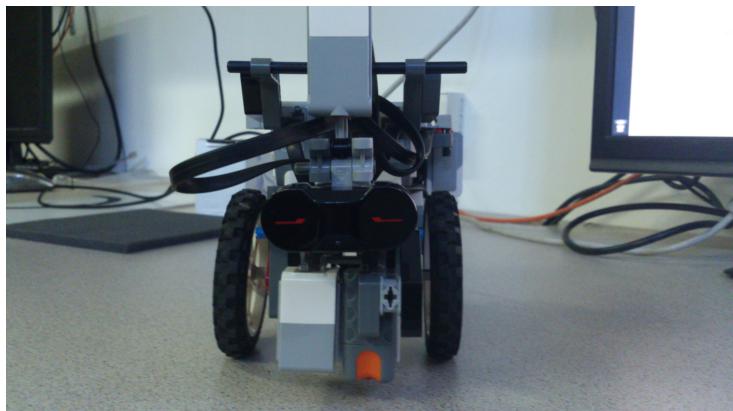
Zbudowany robot to przykład maszyny o napędzie różnicowym z jednym punktem podparcia w postaci koła sferycznego. Do jego konstrukcji użyliśmy dwa duże serwomechanizmy oraz dwa koła o największej średnicy. Aktualne położenie koła odczytywaliśmy wprost z biblioteki *EV3dev*.

Dodatkowo, chcąc zapewnić łatwo realizowalne omijanie przeszkody z obu stron, zdecydowaliśmy się na zamontowanie czujnika odległości w średnim serwomechanizmie, dzięki czemu mogliśmy go obracać w zakresie ($-90^\circ, 90^\circ$), co dawało nam możliwość dokładnego ocenienia, w jakim miejscu znajduje się przeszkoda.

Kulkę podporową umieściliśmy z tyłu pojazdu, solidnie mocując do podwozia robota. Komputer usadowiliśmy możliwie daleko, tak by środek ciężkości



Rysunek 2.1: Widok z boku. Na zdjęciu widać dokładnie mikrokomputer oraz serwomechanizmy.

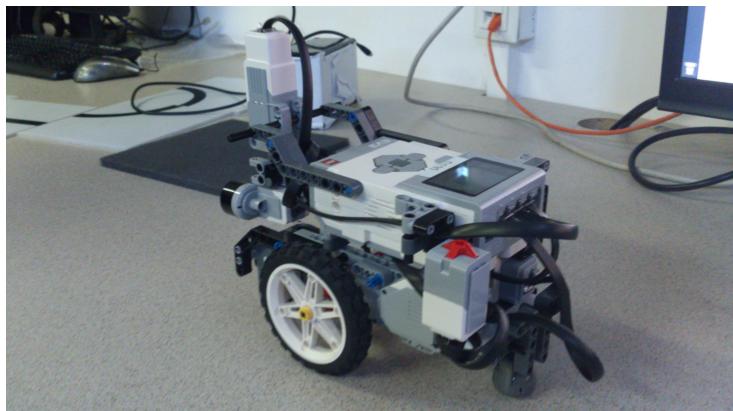


Rysunek 2.2: Przód robota. Czujniki podczerwieni, a nad nimi czujnik odległości.

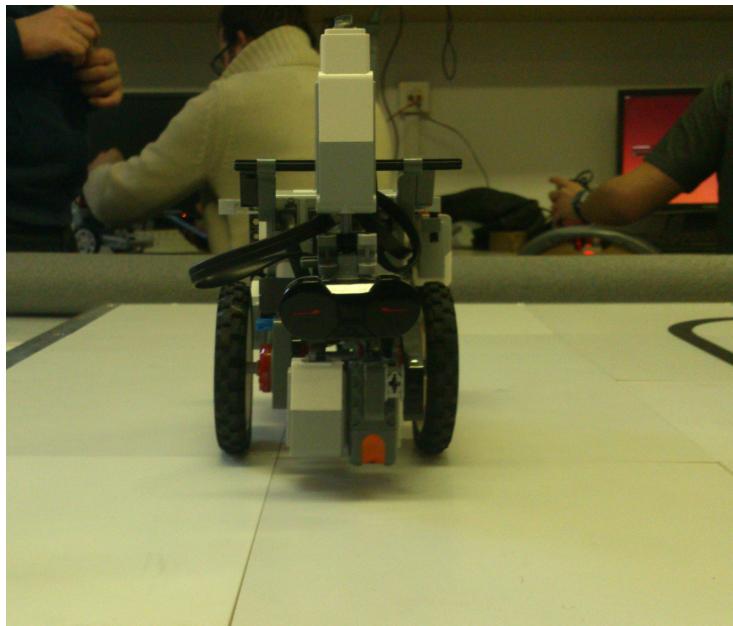
był przesunięty do tyłu, co pomagało uniknąć szarpiię podczas gwałtownego przyspieszania i hamowania.

Godny uwagi jest także sposób upakowania kabli w robocie, tuż pod kostką, tak by zredukować bezwładność robota.

Zdjęcia poniżej dokładniej przedstawią konstrukcję robota:



Rysunek 2.3: Tył robota. Widoczny *ballcaster* oraz *pushbutton*



Rysunek 2.4: Robot na planszy do FTL

Rozdział 3

Algorytm śledzenia linii.

Tutaj zostanie przedstawiony sposób, w jaki robot porusza się po czarnej linii na białym tle.

3.1 Zasada działania algorytmu

Z racji tego, że oba czujniki pokrywają linię, musielibyśmy podzielić program na dwie zasadnicze części :

- Sterowanie automatyczne, regulatorem PID,
- Sterowanie manualne

Sterowanie automatyczne. Regulator PID.

Sposób działania

Zastosowany schemat regulacji to klasyczny regulator proporcjonalno-całkującoc różniczkujący w wersji dyskretnej. W związku z tym, całka zamienia się w sumę, a różniczka - w różnicę. Dodatkowo zastosowaliśmy pewne usprawnienie, mające na celu proporcjonalnie zmniejszać prędkość na zakrętach zgodnie z ustaloną przez nas funkcją liniową. Bardzo to pomogło regulatorowi PID, zwłaszcza pokonywać zakręty z łukami, przez co częściej program pracuje w tym trybie, zamiast w manualnym.

Zastosowaliśmy dwa regulatory PID, których działanie sumuje się na obu silnikach. Dwa, ponieważ mamy dwa czujniki, w dodatku o różnych wskazaniach. Tak otrzymane sterowanie wstawiamy do wspomnianej funkcji ograniczania prędkości. Wszystko na koniec sumujemy i wystawiamy na sterowniki silników.

A poniżej wycinek kodu obrazujący działanie regulatora :

```
# kalkulacje dla regulatora PID
lerror = lMaxBlack - lval
rerror = rMaxBlack - rval

lintegral = 0.5 * lintegral + lerror
rintegral = 0.5 * rintegral + rerror

lderivative = lerror - lastErrorL
rderivative = rerror - lastErrorR

lastErrorL = lerror
lastErrorR = rerror

lcontrol = int((2.6 * lerror + 1 * lintegral + 2.5 * lderivative))
rcontrol = int((2.6 * rerror + 1 * rintegral + 2.5 * rderivative))

...
# Dodatkowe ograniczanie prędkości
predkosc = 500 + int((-1) * (abs(rcontrol - lcontrol)))
# Ustawienie sterowania na silniki
lmotor.run_forever(speed_sp = predkosc + (rcontrol-lcontrol)/2)
rmotor.run_forever(speed_sp = predkosc + (lcontrol-rcontrol)/2)
```

Dobór nastaw regulatora

Nastawy regulatora PID dobieraliśmy doświadczalnie. Zastosowane nastawy nie zapewniają co prawda bardzo dynamicznej odpowiedzi na skokowe zmiany w czujnikach, ale gwarantuje gładką jazdę na prostych i lżejszych łukach.

Podobnie z parametrem funkcji liniowej ograniczającej pęd robota. Współczynnik -1 uznaliśmy za odpowiednią szybkość zmniejszania prędkości robota.

Sterowanie manualne Powstało, by zaprogramować pokonywanie zakrętów pod większym kątem niż 45 stopni. W normalnej sytuacji, regulator przez konstrukcję robota jest zbyt słaby, by móc pokonywać np. kąty proste. W wyniku tego robot po prostu wylatuje z trasy i jedzie w nieskończoność prosto (bo sterowanie się w sumie zeruje na białym tle). Rozwiązałyśmy to w następujący sposób :

1. Wykryj, gdy robot całkowicie zjechał z linii,
2. Obróć się w przybliżeniu o 135 stopni w jedną stronę. Obrót w miejscu
3. Jeśli nie wykryto ponownie linii, Skręć z jeszcze większą prędkością, z tym że teraz w drugą stronę.
4. Skręcaj tak długo aż wykryjesz linię.

Metoda ta jest kołem ratunkowym dla robota po wypadnięciu z trasy, a przy tym jest pewna. W końcu trafi na linię. By zredukować przypadkowość wyboru kierunku skrętu zaprogramowaliśmy pewną logikę. Program po prostu zapamiętuje, w którym kierunku skręcał ostatnio i dobiera tak nowy kierunek skrętu. Z obserwacji wynikło, że znacznie to poprawiło jazdę po kątach prostych.

Rozdział 4

Algorytm omijania przeszkód.

Tutaj zostanie omówiony sposób, w jaki robot wykrywa i omija przeszkody w postaci prostopadłościanu.

4.1 Zasada działa algorytmu

Algorytm omijania przeszkody załącza się dopiero wtedy, gdy robot, jeżdżąc po linii, wykryje zmianę wskazań czujnika odległości, czyli pojawienie się w bliskiej odległości przeszkody. Wygląda on następująco :

1. Zatrzymaj pojazd w sposób łagodny, unikając szarpnięć,
2. Obróć robota o 90° lub o -90° , w zależności od wyboru podjętego przez programistę,
3. Obróć czujnik odległości o ten sam kąt, ale z przeciwnym znakiem, tak by "patrzył" prostopadle na przeszkodę,
4. Pobierz wskazanie czujnika, by znać odległość, w jakiej jesteśmy od przeszkody,
5. Jedź tak długo, aż z czujnika zniknie przeszkoda, po czym wykonaj pojazdem skręt powrotny (czujnik pozostaje nadal prostopadle do przeszkody),
6. Robot jest w pewnej odległości za przeszkodą. Pobierz wartość czujnika, by móc później wykryć ponownie przeszkodę,
7. Jedź tak długo, aż w czujniku pojawi się przeszkoda,
8. Jedź tak długo, aż z czujnika zniknie obiekt,

9. Wróć na linię i kontynuuj program.

4.2 Dobór nastaw regulatora

Jak widać, zastosowany algorytm jest elastyczny, Robot nie ma zapisane na stałe, jak długo ma jechać dookoła przeszkody, ale w "inteligentny" sposób potrafi wykryć pojawienie się jej w czujniku. Prędkość ustawiona jest jako stała, ale może być zmodyfikowana w łatwy sposób. Wartość początkowego wykrycia przeszkody dobrano doświadczalnie, podobnie współczynnik wykrycia, kiedy przeszkoda znów się pojawia, albo znika z pola czujnika.

Rozdział 5

Podsumowanie. Wady i zalety.

Podsumowując, udało nam się osiągnąć w przybliżeniu postawione zadanie. Pora usystematyzować, jakimi zaletami i wadami wyróżnia się robot :

Zalety

- Wysoka dokładność ruchu po linii. Bardzo rzadko zdarzają się oscylacje.
- Prosta konstrukcja. W łatwy sposób można zdemontować części i wymienić na inne, czy też szybko wymienić baterie.
- Ruchomy czujnik odległości zapewnia elastyczne sterowanie trajektorią omijania przeszkody, również z dowolnej strony.
- Robot sam się kalibruje przed każdym przejazdem. Wystarczy go tylko postawić na trasie i sam dokona pomiaru.

Wady

- Mimo zastosowanych ulepszeń bezwładność robota nadal jest duża. Przez to nie możemy rozwinąć wysokich prędkości robotem.
- Algorytm nie jest najszybszy, ale stabilny, a przede wszystkim bez oscylacji.