POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: Automatyka i Robotyka (AIR)
SPECJALNOŚĆ: Robotyka (ARR)

PRACA DYPLOMOWA INŻYNIERSKA

Optymalizacja trajektorii przejazdu trasy robota klasy linefollower

Optimal line tracking methods for linefollower robots

AUTOR:
Jędrzej Stolarz

PROWADZĄCY PRACĘ:
dr inż. Robert Muszyński

OCENA PRACY:

WROCŁAW 2015

Spis treści

Spis rysunków

Rozdział 1

Wstęp

Roboty klasy linefollower buduje się w celu udziału w zawodach polegających na jak najszybszym przejechaniu trasy wyznaczonej czarną linią na białym podłożu. Najczęściej są to konstrukcje amatorskie, roboty mobilne klasy (2,0), wyposażone w kilka czujników linii i sterowane przy pomocy prostego regulatora PID. Niestety takie rozwiązanie nie pozwala na pokonanie trasy w sposób optymalny, ponieważ robot sterowany w ten sposób, znając jedynie przebieg trasy bezpośrednio pod sobą nie jest w stanie wyznaczyć maksymalnej prędkości z jaką mógłby się poruszać na danym odcinku. W związku z tym jego konstruktor dostrajając regulator, musi ograniczyć prędkość maksymalną tak, by robot nie wypadł z żadnego zakrętu. Taka sytuacja może powodować duże straty w czasie przejazdu, zwłaszcza gdy na trasie znajdują się ciasne zakręty (po których robot musi jechać bardzo powoli) i długie proste (na których robot mógłby osiągnąć dużą prędkość).

Jedną z metod pozwalających na poprawę opisanej powyżej sytuacji jest możliwość zastosowania turbiny która ma na celu zwiększenie docisku robota do podłoża. Niestety nie jest to najlepsze rozwiązanie ponieważ pomimo tego, że wyposażone w nią roboty osiągają wysokie prędkości przejazdu, to nie zawsze radzą sobie z pokonaniem trasy, często na przykład nie zauważając zakrętu pod kątem prostym. Dodatkowo na wielu zawodach robotycznych wprowadzono podział robotów LF na kategorię z i bez turbiny. Z tego powodu powstała potrzeba wyznaczenia innej metody pozwalającej na skracanie czasu przejazdu.

Konkurencja wyścigów robotów linefollower często porównywana jest do wyścigów bolidów F1 [1]. W wyścigach tych kierowca, przed rozpoczęciem zawodów, doskonale zna przebieg trasy, dzięki czemu podczas wyścigu wie w jaki sposób i z jaką prędkością może pokonać każdy z zakrętów na torze.

Analogicznie w zawodach robotów linefollower, można zapisać w pamięci sterownika robota przebieg toru, który następnie zostanie zoptymalizowany i dla którego kolejnych odcinków zostaną wyznaczone maksymalne prędkości przejazdu. W ten sposób robot powinien uzyskać minimalny czas przejazdu na jaki pozwalają mu ograniczenia konstrukcyjne i programowe.

Optymalizacja trajektorii w przypadku samochodów wyścigowych najczęściej polega na wyznaczeniu takiego toru ruchu auta, aby krzywizna tego toru była jak najmniejsza, nie zwiększając przy tym znacząco jego długości. Takie postępowanie wynika z założenia, iż prędkość maksymalna pojazdu na zakręcie jest pochodną relacji siły tarcia poprzecznego kół do siły odśrodkowej.

W literaturze znaleźć można prace, w których do rozwiązania wspomnianego problemu optymalizacji zastosowano między innymi algorytmy genetyczne, sztucznej inteligencji, programowania nieliniowego [1, 2]. Jednakże algorytmy te są trudne w implementacji

oraz posiadają bardzo dużą złożoność obliczeniową, przez co nie nadają się do użytku w trakcie trwania zawodów, gdy podczas przejazdów finałowych czas pomiędzy kolejnymi startami jest ograniczony [5].

Innym sposobem wyznaczenia optymalnego przebiegu trasy jest podejście geometryczne, w którym to na podstawie parametrów każdego z zakrętów na torze, możemy zaplanować kształt ścieżki przejazdu tak, by zapewnić możliwość przemieszczania się w sposób najlepiej wykorzystujący własności jezdne pojazdu, zakładając, że ów pojazd porusza się bez poślizgów. Podejście takie jest niestety optymalne jedynie lokalnie, a nie tak jak wyżej wymienione globalnie.

Celem pracy jest skonstruowanie platformy mobilnej pozwalającej na budowę mapy przebiegu trasy i pokonanie jej w sposób optymalny z pomocą zaprojektowanego algorytmu wyznaczającego taką trajektorię przejazdu, która pozwala na pokonanie trasy w jak najkrótszym czasie.

Projekt zakłada, że algorytm optymalizacji powinien mieć taką złożoność obliczeniową, aby jego realizacja była możliwa na sterownikach robotów linefollower w czasie ograniczonym regułami konkurencji. Algorytmem spełniającym ten warunek jest algorytm geometryczny, dlatego został on wybrany do rozwiązania problemu postawionego w tytule pracy.

Rozdział 2

Budowa mapy i optymalizacja trajektorii

Proces wyznaczenia optymalnej trajektorii dla robota można podzielić na dwa etapy, pierwszy z nich to zbudowanie dokładnej mapy przebiegu trasy, a drugi to dokonanie jej optymalizacji. Optymalizacja trajektorii w sposób geometryczny pozwala na wyznaczenie optymalnej ścieżki w sposób lokalny, dla kolejnych odizolowanych grup zakrętów. Prędkość z jaką pokonywane są kolejne zakręty wynika z wartości przyczepności kół i ograniczona jest tym, że siła odśrodkowa nie może być większa od siły tarcia poprzecznego. Zakłada się, że grupa zakrętów jest izolowana wtedy gdy przed i za nią znajdują się wystarczająco długie odcinki prostych.

Następnie, po zoptymalizowaniu każdej grupy zakrętów, możemy zapisać w pamięci robota przebieg optymalnej trajektorii w celu jej późniejszej realizacji.

2.1 Budowa mapy trasy przejazdu

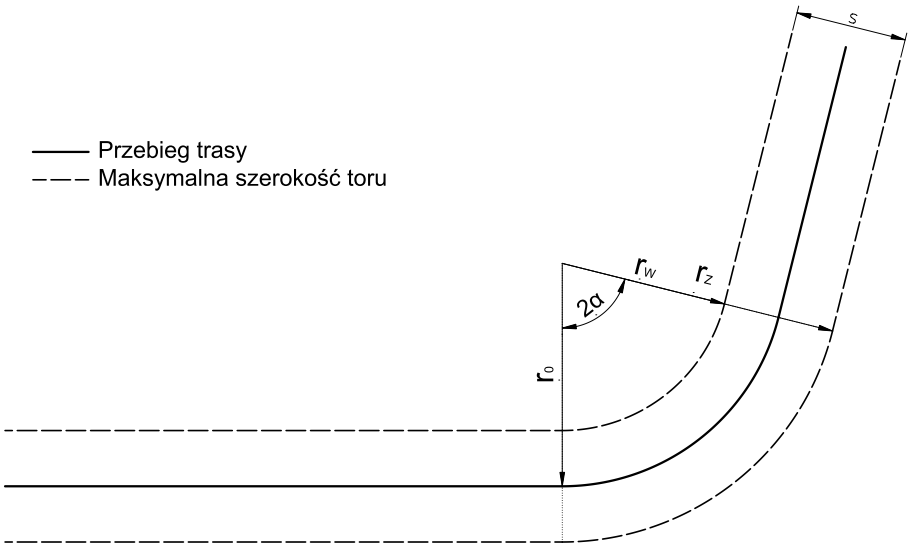
Mapa przebiegu trasy budowana jest na podstawie odczytów z systemu odometrii wykonywanych w trakcie przejazdu, w tym celu posłużyć można się na przykład odczytami z enkoderów, które pozwalają na zmierzenie obrotu każdego z kół robota. Na tej podstawie możliwe jest wyznaczenie pozycji i orientacji robota w każdej próbce, według następującej zależności:

$$\begin{cases} x_i &= x_{i-1} + \frac{1}{2}\cos\theta_{i-1}(\Delta l_r + \Delta l_l) \\ y_i &= y_{i-1} + \frac{1}{2}\sin\theta_{i-1}(\Delta l_r + \Delta l_l) \\ \theta_i &= \theta_{i-1} + \frac{1}{b}(\Delta l_r - \Delta l_l) \end{cases} \tag{2.1}$$

gdzie x_i, y_i, θ_i określają pozycję i orientację w próbce i, b oznacza rozstaw kół robota, a Δl_r i Δl_l określa drogę pokonaną przez każde z kół między kolejnymi próbkami.

Następnie w celu poprawy jakości zbudowanej mapy dane odometryczne filtrowane są przy pomocy odczytów z czujników świata zewnętrznego. Filtracji można dokonać przy pomocy filtru komplementarnego [4], który jest prostym w implementacji filtrem dającym bardzo dobre efekty. Przykładowo dla poprawy jakości odwzorowania zmian orientacji platformy, można wykorzystać do filtrowania dane pochodzące z żyroskopu. W takim przypadku filtr komplementarny przybiera następującą postać

$$\alpha_i = \eta_\omega(\alpha_{i-1} + \omega_i \Delta t) + \eta_\theta \theta_i, \tag{2.2}$$



Rysunek 2.1 Podstawowe parametry zakrętu

gdzie α_i oznacza orientację robota po filtracji, η to wagi filtru, ω_i wartość kolejnych próbek wartości odczytanych z żyroskopu, a Δt to okres między kolejnymi próbkami.

Obliczenia te można wykonać po przesłaniu odczytów z czujników do pamięci komputera i w ten sposób zbudować odwzorowanie trasy. Następnie znając dokładny przebieg trasy, możemy wyznaczyć kolejne grupy zakrętów, dla których będziemy dokonywać optymalizacji, oraz krzywiznę każdego z łuków na trasie.

2.2 Optymalizacja trajektorii

Mapa wyznaczona w podrozdziale 2.1 wraz z parametrami pojazdu takimi jak: rozstaw kół, minimalny promień skrętu, maksymalna wartość przyśpieszenia i opóźnienia, oraz maksymalna możliwa do osiągnięcia prędkość, są niezbędne do wyznaczenie optymalnej trajektorii.

Geometryczna metoda optymalizacji trajektorii przejazdu trasy robota linefollower wymaga zdekomponowania całej trasy na grupy zakrętów odizolowanych od pozostałych, wraz z prostymi odcinkami przed jak i za nimi. W ten sposób uzyskujemy kolejne nakładające się na siebie fragmenty trasy, które poddajemy lokalnej optymalizacji. Szczególnym przypadkiem takiej grupy jest pojedynczy zakręt wraz z przylegającymi do niego prostymi. W niniejszej pracy na potrzeby rozważań teoretycznych zostanie użyta właśnie taka konfiguracja. Schemat takiego zakrętu wraz z jego podstawowymi parametrami przedstawiono na rysunku 2.1. Szerokość toru s jest pochodną szerokości robota, promień r_w jest najmniejszym promieniem po jakim może przejechać robot nie wyjeżdżając swoim obrysem z poza tor, promień ten jest o $\frac{1}{2}s$ mniejszy od zmierzonego promienia śledzonej linii. Analogicznie promień r_z wyznacza zewnętrzną krawędź po której robot może się poruszać nie wyjeżdżając poza linię trasy.

Ścieżka ruchu dla optymalnej trajektorii przebiega w taki sposób, aby była najkrótsza i by robot mógł pokonać ją z największa możliwą prędkością. Oznacza to, że jeżeli robot

może osiągnąć swoją maksymalną prędkość jadąc po wewnętrznej krawędzi toru, co oznacza, że w tej sytuacji jego siła odśrodkowa jest nadal mniejsza od siły tarcia poprzecznego, to powinien pokonać ten zakręt w taki sposób. Dla wyznaczonego promienia R_1 mniejszego od minimalnego promienia skrętu pojazdu, wykonanie optymalnej trajektorii może okazać się niemożliwe, jednakże roboty linefollower są zazwyczaj platformami mobilnymi klasy (2,0), co oznacza że ich minimalny promień skrętu wynosi 0, dlatego zagadnienie to można pominąć w niniejszej pracy. W przeciwnym wypadku, gdy pokonując zakręt po wewnętrznej robot musiałby zwolnić, miejsce wejścia i wyjścia z zakrętu należy przesuwać w kierunku zewnętrznej krawędzi zakrętu, pozostawiając środek zakrętu styczny do jego krawędzi wewnętrznej. Proces ten należy kontynuować do momentu gdy otrzymany zakręt będzie można przejechać z prędkością maksymalną, lub gdy miejsce wejścia i wyjścia osiągnie zewnętrzny skraj toru. Zabieg ten nazywamy rozluźnianiem zakrętu. W praktyce cały ten proces sprowadza się do wyliczenia minimalnego promienia zakrętu, który robot może przejechać z prędkością maksymalną, oraz maksymalnego promienia rozluźnionego zakrętu i wybraniu mniejszego z tych dwóch.

Ogólną ideę optymalizacji trajektorii zakrętu poprzez opisanie na nim łuku o najmniejszej możliwej krzywiźnie przedstawia rysunek 2.2, można na nim zauważyć, że zakręt uległ wydłużeniu na rzecz prostych które do niego przylegają. Długość prostej która potrzebna jest do opisania na torze łuku o większym promieniu jest wyprzedzeniem wejścia w zakręt (na rysunku oznaczona jako l), od niej zależna jest długość prostej która niezbędna jest do odizolowania grup zakrętów w procesie optymalizacji, zarówno przed, jak i za daną grupą zakrętów.

Promień R_1 niezbędny do obliczenia trajektorii przejazdu, może zostać wyznaczony, uwzględniając geometryczne zależności jak na rysunku 2.3. Zależność tą można przedstawić za pomocą równania w następujący sposób:

$$\frac{R_1}{R_1 - s} = \frac{R_1 + s + a}{R_1 + b}, \tag{2.3}$$

gdzie:

$$\begin{aligned} a &= r_z\gamma \\ b &= r_w\gamma \\ \gamma &= \tfrac{1-\cos\alpha}{\cos\alpha} \end{aligned}$$

Długość promienia R_1 wynosi zatem:

$$R_1 = \frac{s(s + r_z\gamma)}{\gamma(r_z - r_w)}. \tag{2.4}$$

Jeżeli minimalny promień zakrętu R_{min} który robot może pokonać z prędkością maksymalną jest mniejszy od R_1 to jego wartość przyjmujemy jako wartość promienia po którym należy pokonać zakręt. Dla takiego promienia szerokość wejścia w zakręt s_{min} wyznaczamy ze wzoru

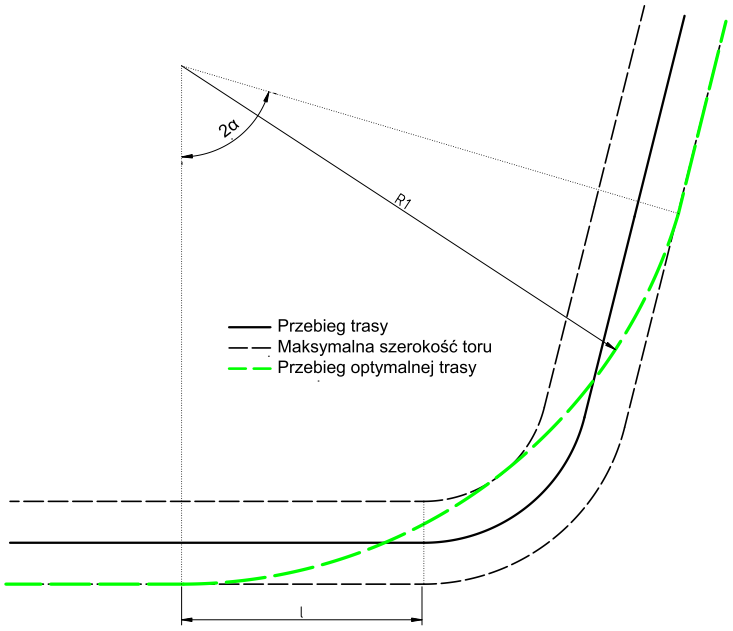
$$s_{min} = \min s, R_{min}(1 - \cos\alpha), \tag{2.5}$$

Odległość wyprzedzenia wejścia w zakręt jest zależna od promienia rozluźnienia zakrętu i można przedstawić ją w następujący sposób:

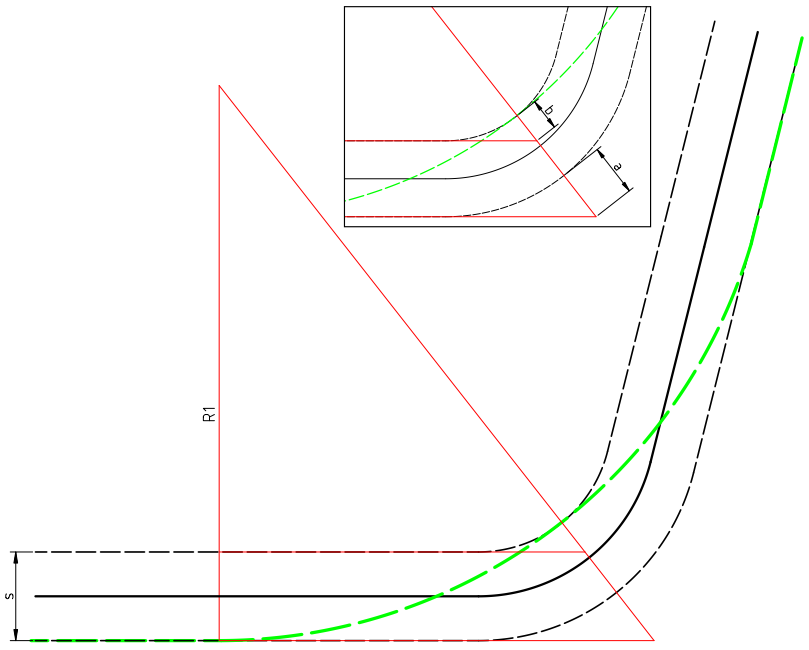
$$l = (R - r_w)\sin\alpha, \tag{2.6}$$

gdzie R jest promieniem pokonywanego zakrętu.

Na podstawie powyższego wzoru możemy dojść do wniosku, że dla zakrętów nierozdzielonych wystarczająco długą prostą również możliwa jest optymalizacja ich pokonywania,



Rysunek 2.2 Schemat optymalizacji zakrętu



Rysunek 2.3 Geometryczne wyznaczenie promienia zakrętu

jednak nie zawsze będzie ona w pełni optymalna. Znając promień maksymalny z jakim możemy pokonać dany zakręt, możliwe jest wyznaczenie prędkości maksymalnej na nim. W tym celu potrzebny jest również współczynnik tarcia poprzecznego który wyznaczyć można w sposób eksperymentalny[1]. Przy znanym współczynniku tarcia poprzecznego kół robota prędkość maksymalna v_r z jaką możliwe jest pokonanie zakrętu o promieniu r opisana jest wzorem

$$v_r = \sqrt{rfg}, \tag{2.7}$$

gdzie f oznacza współczynnik tarcia poprzecznego kół, zaś g przyśpieszenie ziemskie.

Do wyznaczenia profilu prędkości robota na trasie, który stanowi późniejszą podstawę do realizacji zoptymalizowanej trajektorii, potrzebna jest również znajomość maksymalnych przyśpieszeń jakimi dysponuje pojazd. Przy ich pomocy zostaną obliczone odległości jakie potrzebne są do wyhamowania robota do odpowiedniej prędkości przed zakrętem, oraz do ponownego rozpędzenia po wyjściu z zakrętu.

Znając wszystkie potrzebne parametry tj. krzywiznę i miejsca wejścia w kolejne zakręty, oraz profil prędkości z jaką będziemy pokonywali zoptymalizowaną trasę, możemy wgrać je do pamięci robota który następnie zrealizuje optymalna trajektorię przejazdu trasy.

[1] W celu wyznaczenia tarcia poprzecznego należy ustawić konstrukcję na arkuszu z materiału z którego wykonywane jest podłoże do trasy podczas zawodów, a następnie odchylić arkusz w taki sposób aby robot zaczął się z niego zsuwać w bok, po czym powoli zmniejszać to odchylenie, aż do momentu w którym robot się zatrzyma. Tangens kąta odchylenia arkusza od podłoża jest szukanym współczynnikiem tarcia.

Rozdział 3

Konstrukcja robota

Robot składa się z dwóch modułów połączonych ze sobą przy pomocy taśmy FFC, oraz listwy z włókna węglowego. Dzięki modułowej budowie robota możliwa jest konfiguracja odległości między modułami, jak również wymiana przedniej płytki w razie uszkodzenia, lub w celu zbadania innej konfiguracji ułożenia czujników linii. Gotową konstrukcję robota przedstawia rysunek 3.1, a schematy ideowy i projekt płytki PCB można znaleźć w dodatku A.

3.1 Moduł główny

Moduł główny robota odpowiada za część logiczną, oraz wykonawczą sterowania, jak również za komunikację. W związku z tym na module głównym umieszczono napęd robota, mikrokontroler sterujący całością platformy, oraz pozostałe elementy niezbędne do działania robota takie jak na przykład układ zasilania. Do komunikacji robota z jego użytkownikiem służy przycisk i dwie diody umieszczone na płytce, wyprowadzono również złącza do programowania i debugowania mikrokontrolera poprzez interfejs SWD, oraz złącze do komunikacji w którym możliwe jest umieszczenie modułu Bluetooth.

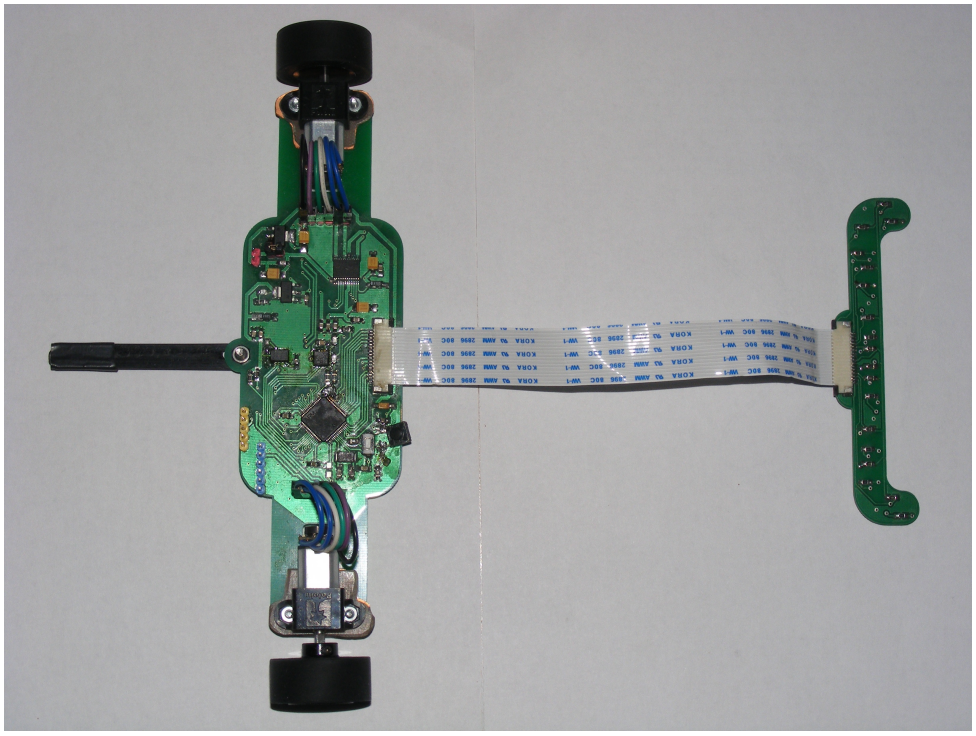
3.1.1 Napęd

Napęd robota bazuje na silnikach firmy Pololu z przekładnią 10:1, zdolnych osiągnąć 3000 obrotów na minutę,a ich moment obrotowy wynosi 0.3 kg*cm. Do sterowania silnikami zastosowano mostek H TB6612FNG, który jest w stanie dostarczyć do pojedynczego silnika prąd o natężeniu do 1A. Napęd przenoszony jest na nawierzchnię przy pomocy kół Solarbotics RW2i o średnicy 31.2 mm.

3.1.2 Mikrokontroler

W robocie zbudowanym na potrzeby pracy dyplomowej został zastosowany mikrokontroler STM32F405RGT6 [6]. Jego główne cechy to:

- 1MB pamięci Flash,
- 192kB pamięci RAM,
- częstotliwość taktowania: 168MHz.
- 14 timerów z możliwością generowania przerwań, sygnału PWM, oraz odczytu wejścia kwadraturowego z enkoderów,



Rysunek 3.1 Gotowa konstrukcja robota

- 12-bitowy przetwornik ADC,
- interfejsy komunikacyjne: I^2C, SPI, UART.

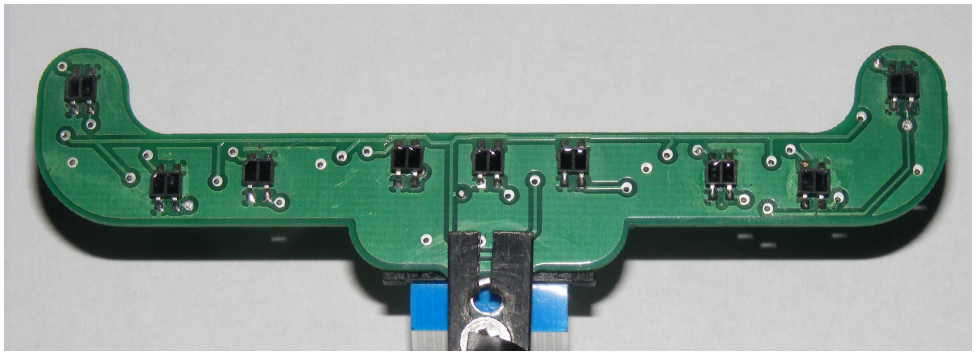
Dzięki tak dużej pamięci RAM możliwe jest zapamiętanie trasy z wysoką rozdzielczością, a duża częstotliwość taktowania procesora umożliwiać będzie w docelowej konstrukcji przeniesienie obliczeń z komputera na mikrokontroler sterujący robotem. Takie wyposażenie mikrokontrolera sprawia, że bardzo dobrze sprawuje się on jako sterownik dla układów wyposażonych w dużą ilość czujników, dla których kluczową wartością jest szybkość i niezawodność. Mimo skomplikowanej architektury, programowanie układów z rodziny STM32 nie sprawia dużych problemów, zawdzięcza się to dostarczonej przez producenta bibliotece HAL [9]. Wykorzystując tę bibliotekę nie musimy martwić się o niskopoziomowe działanie mikrokontrolera, natomiast możemy skupić się na implementacji algorytmów działania.

Powyższe cechy, oraz dostępność na polskim rynku w niewysokiej cenie, przesądziły o tym, aby w projekcie użyć tego układu.

3.1.3 Czujniki

Na module głównym robota umieszczone są czujniki odpowiedzialne za ustalenie pozycji platformy na trasie. Czujnikami tymi są:

- Enkodery magnetyczne firmy Pololu, które przy zastosowanych silnikach z przekładnią 10:1, posiadają rozdzielczość 120 impulsów na jeden obrót koła (dla koła o średnicy 31mm daje to rozdzielczość na poziomie 0.8mm).
- Żyroskop L3GD20 firmy ST, pozwalający na określenie rotacji robota względem pozycji początkowej w trakcie przejazdu [7].



Rysunek 3.2 Przedni moduł z czujnikami linii

- Akcelerometr LSM303DLHC, który może służyć to odczytu przyśpieszeń dzięki którym możemy określić przemieszczenie robota. Dodatkowo, układ LSM303DLHC posiada magnetometr który może być wykorzystany do określenia orientacji robota [8].

Odczyty z wyżej wymienionych czujników posłużą do określenia pozycji i orientacji robota na trasie, co pozwala na stworzenie mapy przebiegu trasy.

3.2 Moduł czujników linii

Na przednim module umieszczone zostały transoptory odbiciowe KTIR0711s, służą one do określenia ułożenia robota względem znajdującej się pod nim linii.

Przedni moduł skonstruowany na potrzeby pracy inżynierskiej posiada 9 transoptorów, ułożonych tak, że skrajne czujniki wysunięte są do przodu względem środkowego (rysunek 3.2). Ma to na celu ułatwienie wykrywania kątów prostych na trasie. Układ rozstał zaprojektowany w taki sposób, żeby uzyskać jak najmniejszą wagę modułu, przy jednoczesnym zachowaniu rozsądnej szerokości rozstawu czujników.

Rozdział 4

Badania symulacyjne i eksperymentalne

W rozdziale tym opisano badania, które zostały wykonane w celu sprawdzenia działania, oraz jakości algorytmu geometrycznej optymalizacji trajektorii. Przetestowano funkcjonalność budowania mapy trasy, oraz eksperymentalnie i symulacyjnie sprawdzono poprawność optymalizacji. Na potrzeby badań eksperymentalnych została wykorzystana platforma mobilna przedstawiona w rozdziale 3, a do badań symulacyjnych założono parametry robota takie jak tej platformy. Podstawowe właściwości robota, które są niezbędne w procesie optymalizacji trajektorii to m.in. szerokość robota równa $0.2m$, współczynnik tarcia poprzecznego 0.4, maksymalne przyśpieszenie na poziomie $4\frac{m}{s^2}$, oraz prędkość maksymalna wynosząca $4\frac{m}{s}$.

4.1 Badanie funkcji budowy mapy

W celu przetestowania algorytmu stworzono dwa odcinki testowe:

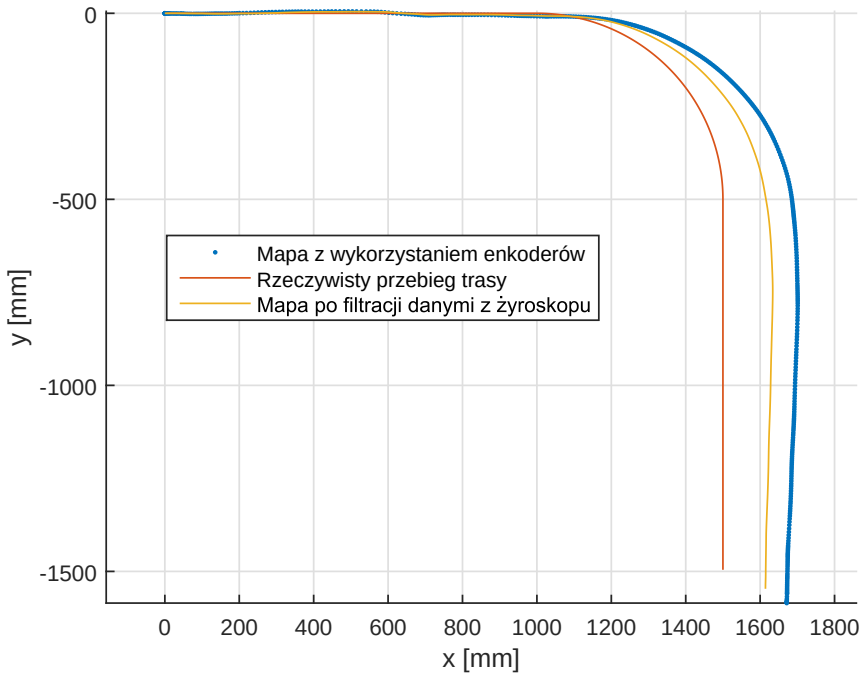
1. składający się z prostej o długości 1m, zakrętu o promieniu 0.5m i kącie $\frac{\pi}{2}$, oraz drugiej prostej o długości 1m. Przebieg przedstawiono na rysunku 4.1,

2. składający się z prostej o długości 0.5m, kąta prostego, prostej o długości 0.3m, łuku o promieniu 0.3m i kącie π, oraz prostej o długości 0.5m. Przebieg przedstawiono na rysunku 4.2.

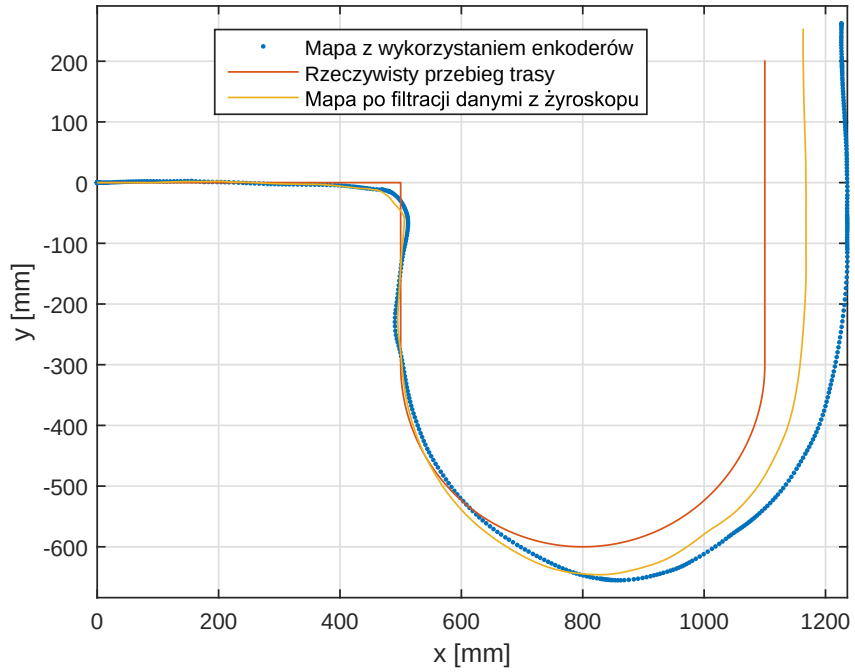
Dla obu odcinków wykonano przejazd mający na celu zbudowanie mapy trasy, następnie za pomocą wzorów 2.1 i 2.2 obliczono przebiegi obu tras. Wyniki zostały przedstawione na rysunkach 4.1, oraz 4.2.

Jak widać na rysunkach 4.1 oraz 4.2 odwzorowanie to nie jest bardzo wysokiej jakości, aczkolwiek można je znacząco polepszyć dzięki fuzji danych z pozostałych czujników. Przy odpowiednim doborze parametrów filtru komplementarnego możliwe jest nawet dwukrotne zmniejszenie błędu pozycji. Można również zauważyć, że błąd pozycji narasta przede wszystkim na długich zakrętach, jest to spowodowane prawdopodobnie nierównomiernym poślizgom wzdłużnym podczas ich pokonywania.

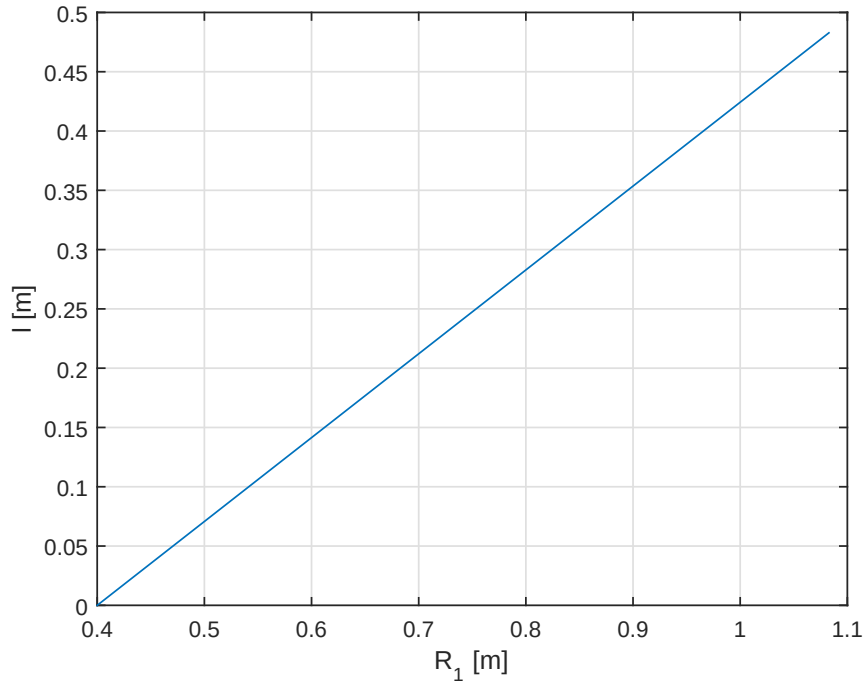
Rozwiązaniem powyższych problemów mogłoby być zaprojektowanie filtru o lepszych właściwościach, np. filtru Kalmana, oraz zmiana materiału z którego wykonane są opony, na taki, który charakteryzuje się wyższą lepkością.



Rysunek 4.1 Pierwszy odcinek testowy



Rysunek 4.2 Drugi odcinek testowy



Rysunek 4.3 Odległość l w zależności od stopnia rozluźnienia zakrętu

4.2 Badanie optymalizacji trajektorii

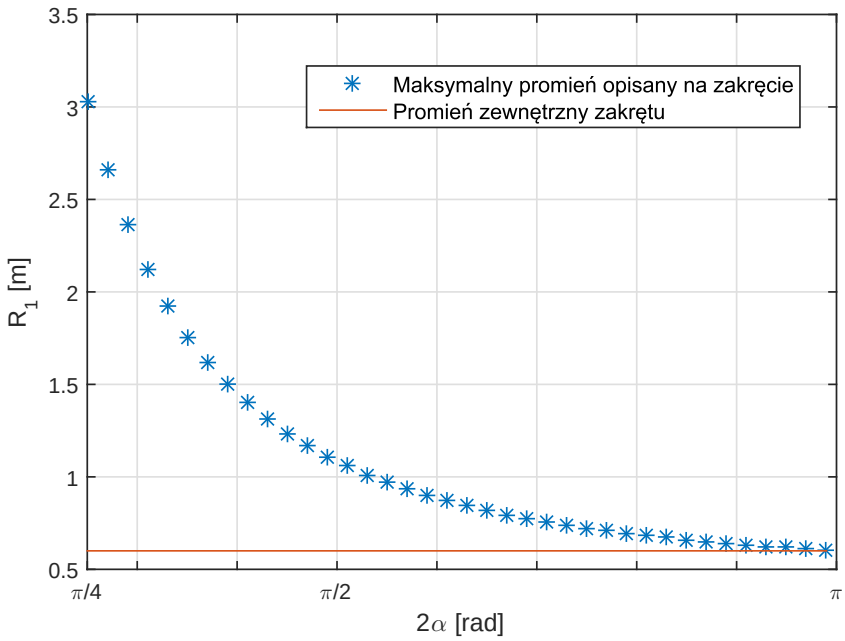
4.2.1 Analiza zachowań algorytmu

Dla sprawdzenia poprawności założeń została wykonana analiza zachowań algorytmu. Sprawdzono jak wielkość rozluźnienia łuku zakrętu wpływa na wydłużenie odległości wyprzedzenia wejścia i wyjścia z zakrętu, jak zależy maksymalny rozluźniony promień zakrętu od jego kąta, oraz jak rozluźnianie wpływa na zmianę odległości jaką trzeba pokonać przy przejeżdżaniu przez zakręt wraz z przylegającymi do niego prostymi. Analizę przeprowadzona dla przykładowego zakrętu o promieniu $r_w = 40cm$, $r_z = 60cm$ i kącie $2\alpha = \frac{\pi}{2}$.

Rysunek 4.3 przedstawia wpływ rozluźniania zakrętu na odległość wejścia w ten zakręt. Jak widać w zależności od tego jak bardzo zwiększony zostanie promień łuku po którym zostanie pokonany zakręt, tak zwiększy się długość prostych które muszą znaleźć się przed i za faktycznym zakrętem w celu pokonania go w sposób zoptymalizowany. Oznacza to, że możliwe jest takie manipulowanie stopniem rozluźnienia zakrętu, aby możliwe było wejście i wyjście z niego nawet w przypadku gdy następny zakręt lub ich grupa znajduje się bardzo blisko tego, który poddawany jest lokalnej optymalizacji.

Następnie zmieniając parametr 2α sprawdzono jak zależy maksymalny promień zakrętu po rozluźnieniu od jego kąta. Rysunek 4.4 przedstawia tą zależność. Zauważyć można że dla zakrętu o kącie π, największy możliwy do opisania na zakręcie promień zbiega do r_z, natomiast dla zakrętów o małym kącie osiąga on znacznie większe długości, dążąc do nieskończoności dla odcinków prostych. Porównując te dane z danymi z rysunku 4.3 można wywnioskować, że dla zakrętów o małym kącie 2α, nie zawsze możliwe będzie całkowite rozluźnienie zakrętu, ze względu na niewystarczająco długie odcinki pomiędzy kolejnymi zakrętami.

Rysunek 4.5 przedstawia profil prędkości dla robota pokonującego zakręt po wewnętrz-



Rysunek 4.4 Maksymalny promień zakrętu po jego rozluźnieniu w funkcji kąta zakrętu

nej krawędzi trasy (Trajektoria I), oraz po trajektorii optymalnej (Trajektoria II). Założono, że prędkość na początku i końcu odcinka testowego jest równa prędkości maksymalnej jaką może uzyskać robot (w tym przypadku $4\frac{m}{s}$). Na wykresie tym można zauważyć również wpływ jaki ma zwiększenie promienia łuku po jakim pokonywany jest zakręt na całkowitą długość trasy robota. Przy maksymalnym rozluźnieniu zakrętu długość ta nieznacznie wzrasta, lecz wzrost ten kompensowany jest tym, że zakręt pokonywany jest ze znacznie większą szybkością.

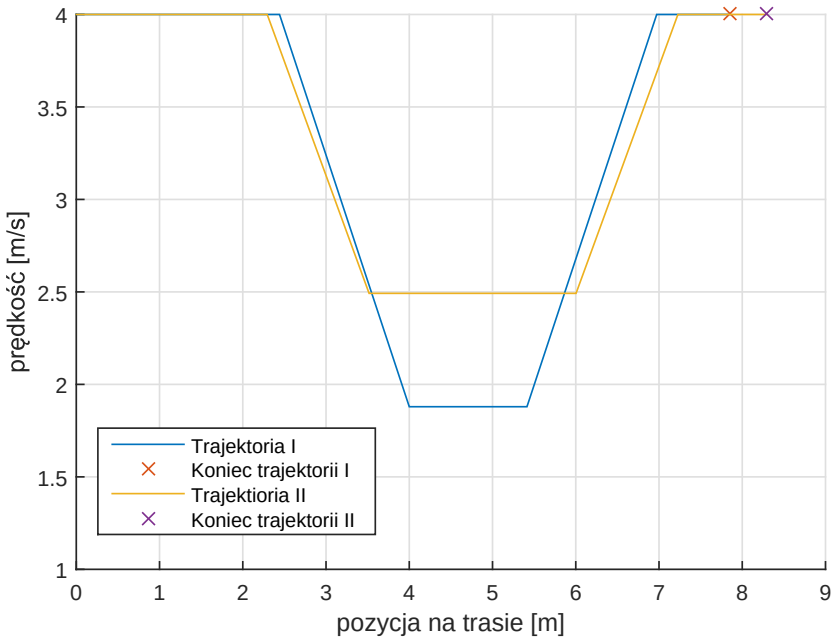
Powyższa analiza wskazała, że założenia i oczekiwania postawione względem opisywanej metody optymalizacji trajektorii przejazdu trasy, są możliwe do spełnienia.

4.2.2 Symulacyjne badanie jakości optymalizacji

Pomiar jakości geometrycznej metody optymalizacji polegał na wyznaczeniu czasu przejazdu przez testowy odcinek, rozpoczynając od przejazdu po wewnętrznej krawędzi zakrętu a następnie zmniejszając jego krzywiznę, w taki sposób, że punkt wejścia i wyjścia z zakrętu zbliżano do zewnętrznej krawędzi. Założono, że prędkość na początku i końcu odcinka testowego powinna być równa prędkości maksymalnej robota, uzyskując profile prędkości podobne do tych jak na rysunku 4.5.

Analizę przeprowadzano przy założeniu, że zakręt ma strukturę jak na rysunku 2.1, jego kąt wynosi $\frac{\pi}{2}$, promień $r_w = 0.8m$, promień $r_z = 1m$, a długość prostych przed i za zakrętem równa się $4m$. Parametry symulowanego robota wyznaczono na podstawie rzeczywistej konstrukcji przedstawionej w rozdziale 3. Prędkość maksymalna, jaką może osiągnąć robot ma znaczący wpływ na jakość optymalizacji, z tego powodu przeprowadzono kilka symulacji dla różnych prędkości maksymalnych możliwych do osiągnięcia przez robota.

Pokonując powyżej opisany zakręt po wewnętrznej krawędzi, robot nie wpadając w poślizg może osiągnąć prędkość $1.78\frac{m}{s}$, natomiast pokonując zakręt ten po łuku o największym możliwym promieniu może osiągnąć na nim prędkość $2.41\frac{m}{s}$.



Rysunek 4.5 Profil prędkości dla minimalnego i maksymalnego promienia

Badania czasu przejazdu przeprowadzono dla następujących prędkości maksymalnych:

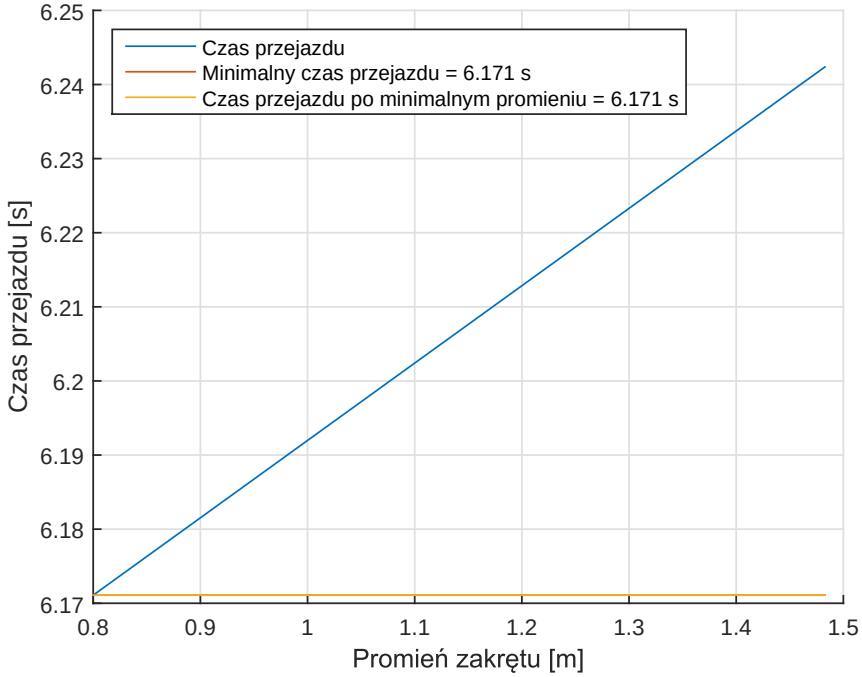
1. $v_{max} = 1.5\frac{m}{s}$
2. $v_{max} = 1.8\frac{m}{s}$
3. $v_{max} = 2.2\frac{m}{s}$
4. $v_{max} = 2.5\frac{m}{s}$
5. $v_{max} = 4\frac{m}{s}$

Na rysunku 4.6 przedstawiono czas przejazdu odcinka testowego w zależności od stopnia rozluźnienia łuku po jakim go pokonujemy, dla prędkości maksymalnej, mniejszej od prędkości z jaką robot może pokonać zakręt po wewnętrznej jego krawędzi. Jak łatwo zauważyć dla takiego przypadku, optymalizacja przez powiększanie promienia skrętu nie sprawdza się, ze względu na wydłużający się tor jazdy, oraz niedostatecznie dużą prędkość jaką możemy na nim osiągnąć. W tym przypadku trasę należy pokonywać po wewnętrznej stronie zakrętu.

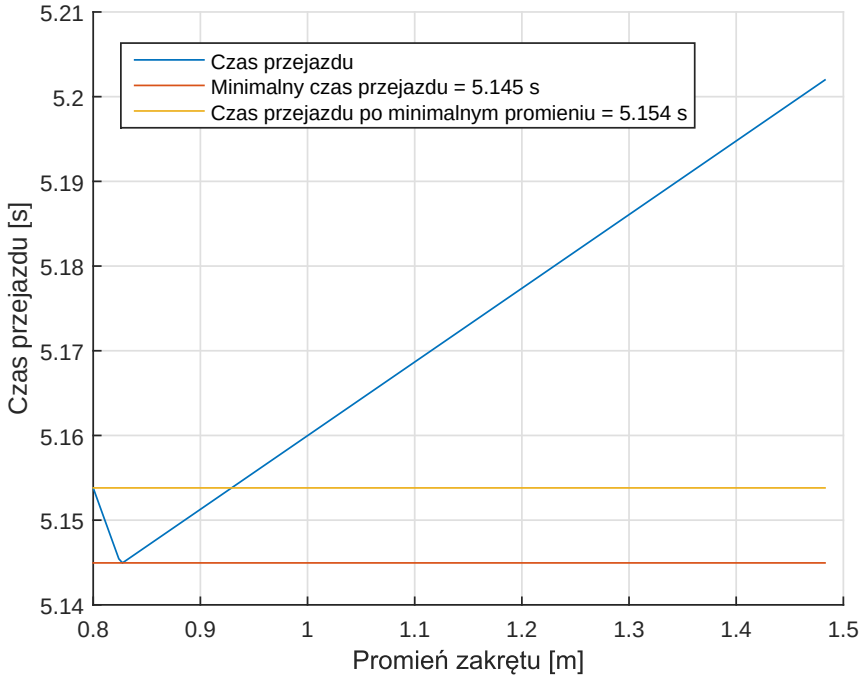
Optymalizacja trajektorii przejazdu przez zakręt, ma zatem sens dopiero wtedy gdy robot jest w stanie osiągnąć prędkość, większą od tej z jaką może pokonać dany zakręt po jego wewnętrznej krawędzi. Sytuację taką przedstawiają rysunki 4.7 i 4.8.

Jak widać, w takich przypadkach optymalną ścieżką robota nie jest ta maksymalnie rozluźniona, lecz taka po której prędkość robota nie ogranicza prędkości pokonywania zakrętu.

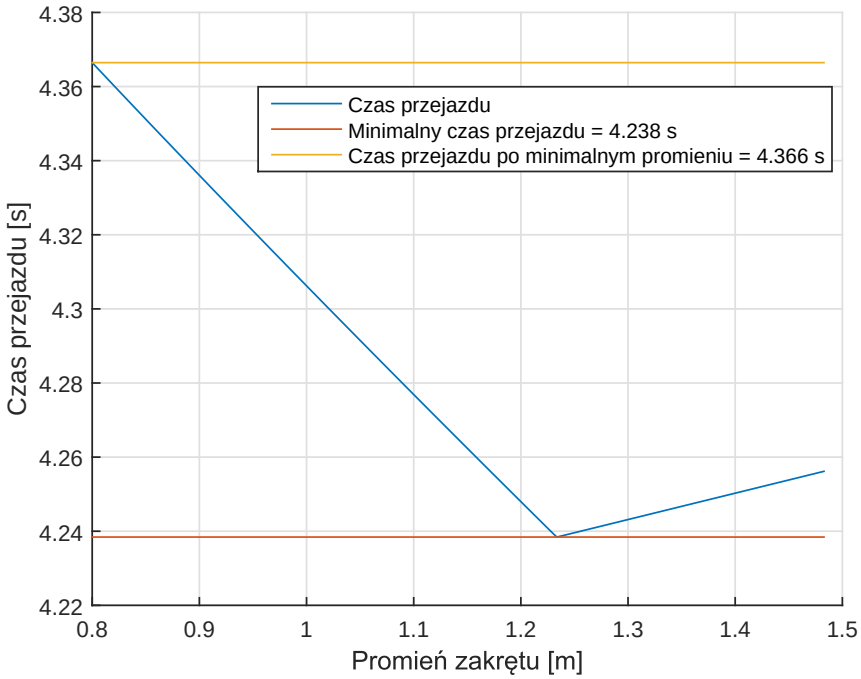
Dopiero zwiększając prędkość maksymalną osiąganą przez pojazd, powyżej progu prędkości z jaką możemy pokonać zakręt po łuku o największym możliwym promieniu jaki mamy możliwość wpisać w dany zakręt, optymalną trajektorią staje się ta która wynika z największego rozluźnienia krzywizny zakrętu. Dokładnie taką sytuację przedstawia rysunek 4.9. Najlepszy czas przejazdu w tym przypadku ($3.70s$), jest osiągany podczas



Rysunek 4.6 Czas przejazdu dla $v_{max} = 1.5\frac{m}{s}$



Rysunek 4.7 Czas przejazdu dla $v_{max} = 1.8\frac{m}{s}$



Rysunek 4.8 Czas przejazdu dla $v_{max} = 2.2\frac{m}{s}$

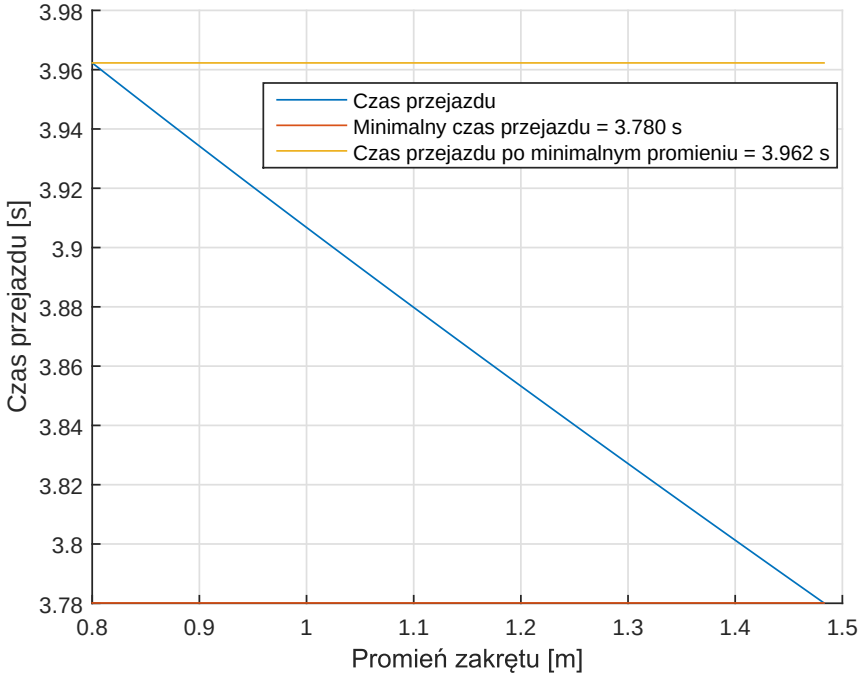
jazdy po łuku o największym promieniu. Rzeczywista konstrukcja przedstawiona w pracy, ma możliwość rozwinięcia prędkości na poziomie $4\frac{m}{s}$. Symulację dla takiej prędkości przedstawia rysunek 4.10.

Jakość zadania optymalizacji trajektorii przejazdu trasy robota linefollower możemy wyznaczyć poprzez stosunek różnicy czasu osiąganego przejeżdżając optymalną trajektorią i czasu przejazdu najkrótszą trajektorią do czasu przejazdu po wewnętrznej krawędzi toru. Dla ustalonej prędkości maksymalnej równej $4\frac{m}{s}$ i nie ograniczającej prędkości pokonywania zakrętu(rys. 4.10), współczynnik ten wynosi 4.6%. Dla wolniejszych robotów, czyli takich które nie są w stanie osiągnąć maksymalnej prędkości z jaką mogłyby przejechać dany zakręt, współczynnik jakości optymalizacji ma wartość mniejszą. I tak na przykład, dla robota z symulacji przedstawionej na rysunku 4.8 wynosi on 2.9%.

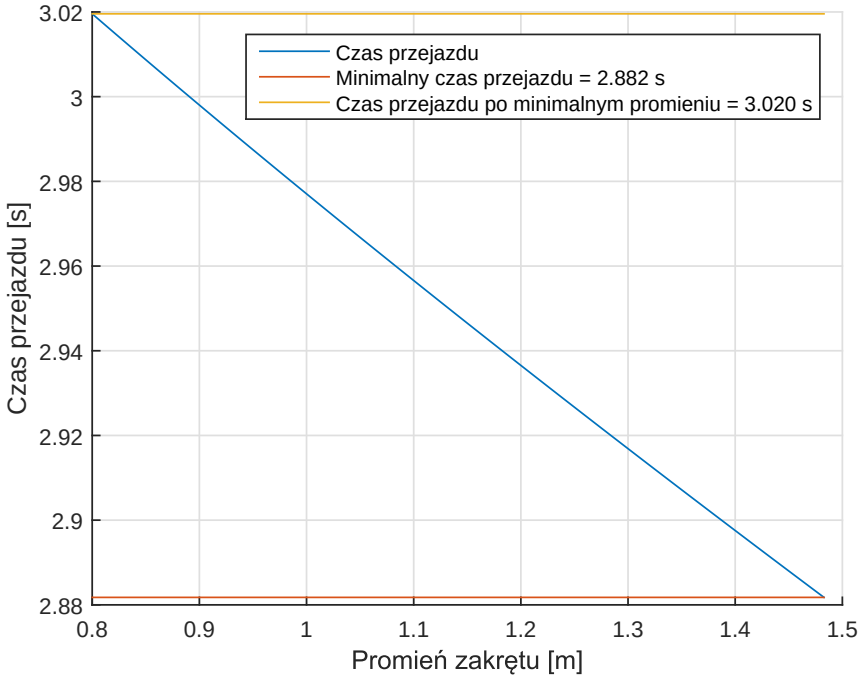
Nie jest to jednak ostateczny zysk uzyskany dzięki zastosowaniu zapamiętania i optymalizacji przebiegu trasy. Roboty nie mające możliwości zapamiętania trasy mają ustawianą prędkość maksymalną na stałej wartości, takiej aby pojazd nie wypadł z trasy podczas pokonywania najwęższego zakrętu na trasie po jego osi, czyli tak aby linia znajdowała się pod środkiem robota. Dla zakrętu wykorzystywanego w powyższych badaniach prędkość ta wynosi $1.88\frac{m}{s}$, co daje czas przejazdu równy $5.01s$. Wynik ten jest o 2.13 sekundy gorszy od wyniku dla zoptymalizowanej trajektorii, zatem ostateczny wskaźnik optymalizacji jest na poziomie 43%.

4.2.3 Badania eksperymentalne

Na podstawie przeprowadzonych symulacji przeprowadzono badania eksperymentalne na rzeczywistej konstrukcji, oraz trasie takiej jak została przedstawiona na rysunku 4.1. Współczynnik tarcia poprzecznego pomiędzy robotem a podłożem był równy około 0.4, co dla zakrętu o promieniu 0.5m daje prędkość maksymalną przejazdu przez łuk na poziomie $1.4\frac{m}{s}$.



Rysunek 4.9 Czas przejazdu dla $v_{max} = 2.5\frac{m}{s}$



Rysunek 4.10 Czas przejazdu dla $v_{max} = 4\frac{m}{s}$

Wykonano serię przejazdów najpierw dla zwykłej metody pokonywania trasy(I), czyli takiej gdzie prędkość maksymalna na całej trasie jest stała, a następnie dla metody przedstawionej w niniejszej pracy(II). Dla pierwszej metody prędkość przejazdu była równa prędkości przejazdu po łuki i wynosiła $1.4\frac{m}{s}$.

Aby robot mógł pozycjonować się względem linii przez cały czas, należało przyjąć za szerokość trasy rozstaw pomiędzy czujnikami linii poprzedzającymi skrajne czujniki, który wynosi 70mm. W związku z tym maksymalny promień łuku po jakim robot mógł pokonać dany zakręt ma długość 70cm, dla takiego promienia prędkość przejazdu wynosi $1.67\frac{m}{s}$. Prędkość maksymalna na prostej została ograniczona do $2.0\frac{m}{s}$, ponieważ powyżej tej prędkości robot w znacznym stopniu tracił przyczepność, co bardzo utrudniało sterowanie nim. Wyniki przeprowadzonych badań przedstawiono w tabeli 4.1.

Metoda	Czas przejazdu							Średnia
I	1.958	1.980	2.056	1.956	2.008	2.114	1.989	2.009
II	1.416	1.541	1.497	1.619	1.672	1.624	1.571	1.563

Tabela. 4.1 Czas pokonania trasy testowej [s]

Można zauważyć, że średni czas przejazdu dzięki zastosowaniu optymalizacji trajektorii przejazdu trasy, polepszył się o 0.46 sekundy, co jest znaczącą poprawą względem czasu przejazdu wykonanego metodą nie używającą mapy trasy.

Rezultaty wykonanych badań pokazują, że optymalizacja trajektorii spełnia pokładane w niej nadzieje, a robot z zaimplementowanym algorytmem pozwalającym na skorzystanie z tej metody ma możliwość uzyskania znacznie lepszego czasu przejazdu od robotów sterowanych standardowo wykorzystywanymi algorytmami.

Rozdział 5

Podsumowanie

W niniejszej pracy przedstawiono jedną z metod optymalizacji trajektorii przejazdu trasy. Metoda ta jest łatwa w implementacji, a przede wszystkim w prosty sposób pozwala na poprawienie wyników uzyskiwanych przez roboty na których jest ona zaimplementowana.

Ważnym aspektem który należy wziąć pod uwagę przy projektowaniu platformy mobilnej na której ma być zaimplementowany algorytm optymalizacji trajektorii, jest to aby była ona odpowiednio szeroka tak, aby szerokość toru jazdy była jak największa. W przeciwnym wypadku optymalizacja nie daje widocznych rezultatów.

Istotną rolę przy przejazdach z dużą prędkością i przyśpieszeniami, odgrywa przyczepność robota względem podłoża. Zapewnia ona możliwość szybszego pokonywania zakrętów oraz zmniejsza drogę hamowania przed nimi.

Spełniając powyższe warunki robot mobilny typu linefollower, będący w stanie przejechać trasę w sposób optymalny, tj. pokonać zakręty po łuku o jak największym promieniu, a proste z jak największą szybkością, wykonuje zadanie to w sposób dający zauważalne rezultaty. Zaproponowana metoda optymalizacji wykazuje duży potencjał, oraz daje możliwości rozwoju. Dalsze prace nad rozwojem implementacji tego algorytmu zdecydowanie powinny przynieść wymierne korzyści w postaci osiągania lepszych wyników.

Dodatkowym atutem zapamiętywania trasy przejazdu może być możliwość udziału w konkurencji lineFollower Enhanced, czyli takiej odmianie zawodów LF, podczas której na trasie umieszczone są przeszkody. Robot po pierwszym przejeździe, znając rozmieszczenie przeszkód, również taką trasę może pokonać w krótszym czasie.

W pracy przyjęto uproszczenie modelu robota, do postaci kinematycznej, oraz sama optymalizacja została przeprowadzony w sposób lokalny. Dalsza rozwój projektu będzie polegać na optymalnym połączeniu kolejnych odcinków trasy, oraz próbie uwzględnienia dynamiki konstrukcji w procesie wyznaczania optymalnej trajektorii.

Kolejnym elementem który zostanie opracowany to lepsze dostrojenie regulatora tak aby robot pokonywał trasę z większą dokładnością, ponieważ oscylując wzdłuż linii zmniejsza on swoją prędkość, a przede wszystkim jakoś odwzorowania optymalnej trajektorii na rzeczywistej trasie.

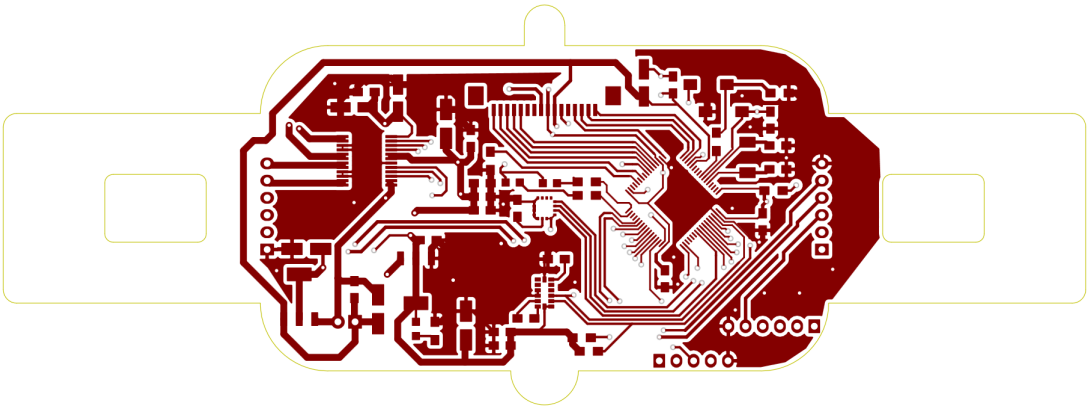
W celu poprawy odwzorowania trasy będą przetestowane inne rodzaje materiałów wykorzystywanych do produkcji opon i moduły czujników linii z innym rozmieszczeniem ich na płytce. Zostanie uruchomiony magnetometr znajdujący się w układzie LSM303DLHC, oraz zaprojektowany filtr Kalmana.

Projekt w celu realizacji optymalnej trasy w sposób globalny, wymaga również dopracowania algorytmu lokalizacji robota względem trajektorii optymalnej i podążania za nią.

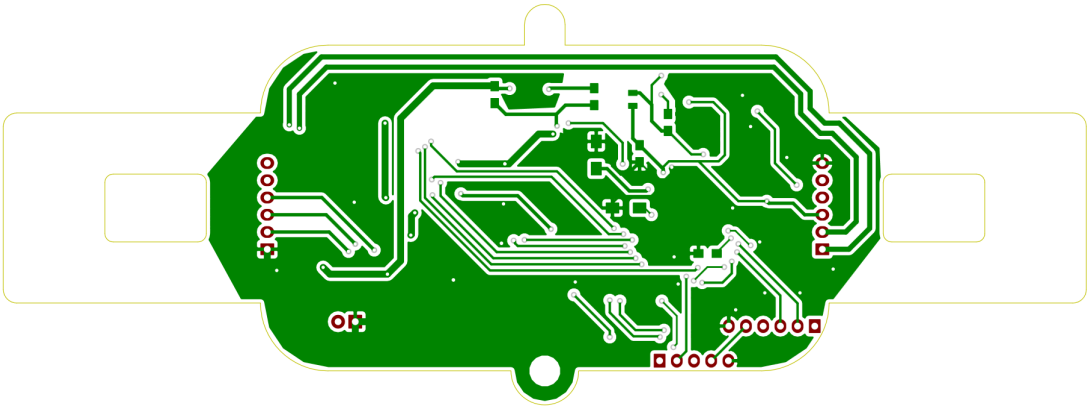
Dodatek A

Schematy elektryczne

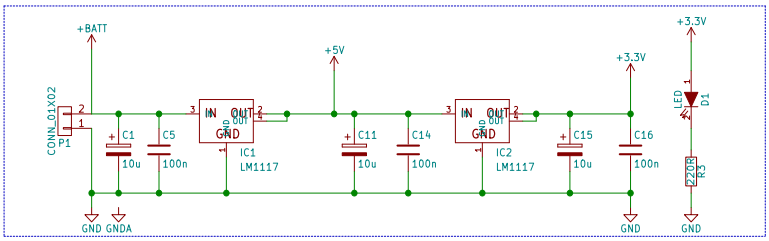
Dodatek przedstawia schemat PCB oraz schemat ideowy układu, który jest modułem głównym, platformy mobilnej skonstruowanej w ramach pracy inżynierskiej.

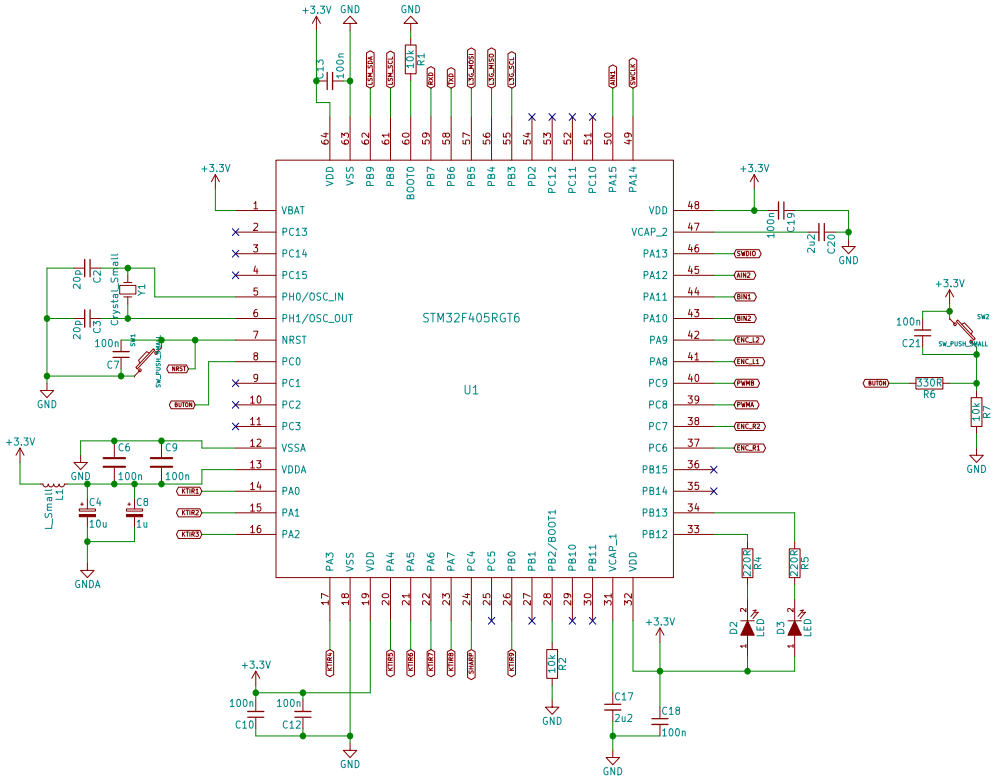


Rysunek A.1 Schemat górnej warstwy płytki PCB

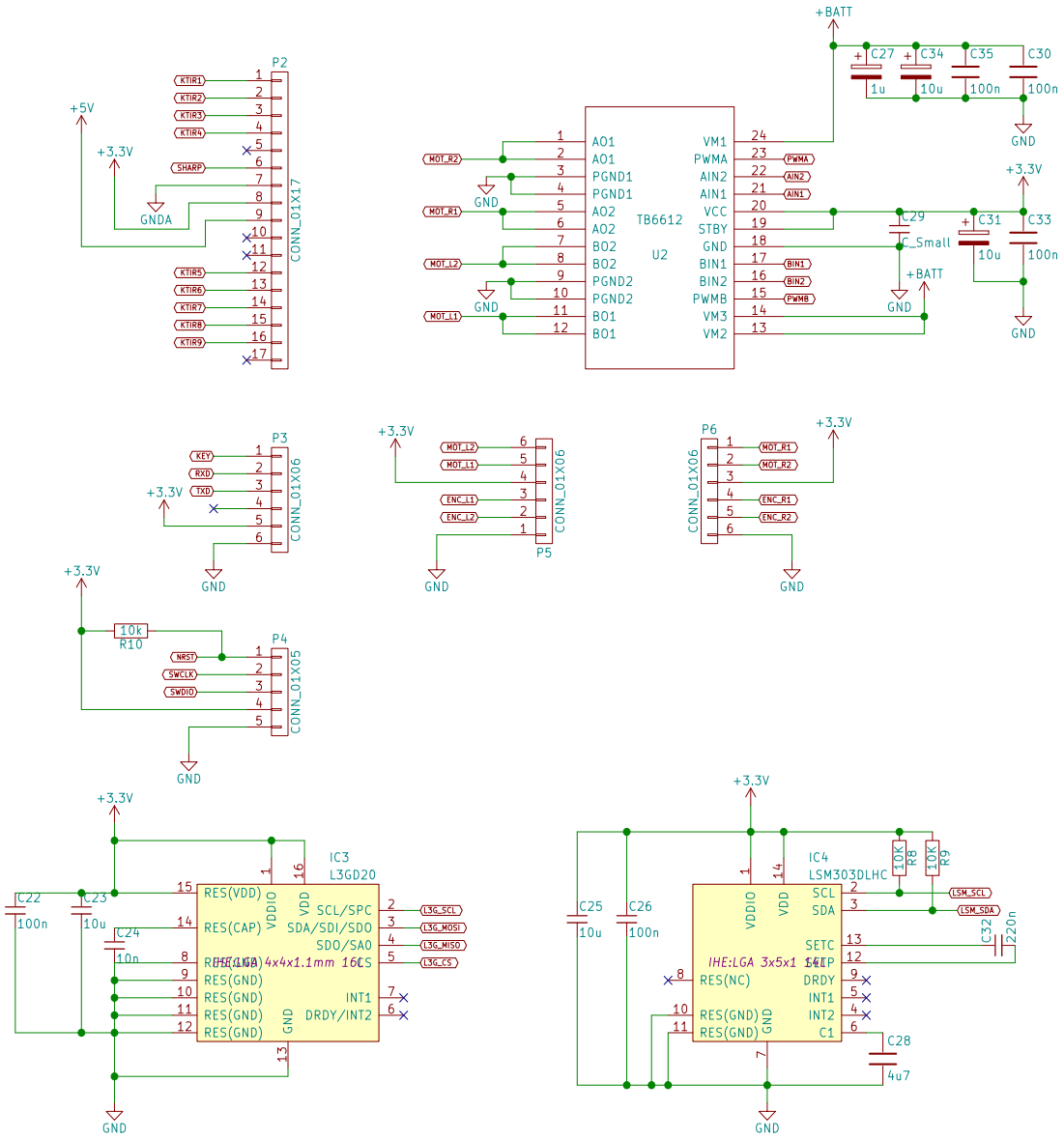


Rysunek A.2 Schemat dolnej warstwy płytki PCB





Rysunek A.3 Schemat ideowy układu elektronicznego cz.1



Rysunek A.4 Schemat ideowy układu elektronicznego cz.2

Bibliografia

[1] Xiong Ying, "Racing Line Optimization". *Master of Science Thesis*, 2010.

[2] Thomas Gustafsson, "Computing The Ideal Racing Line Using Optimal Control", 2008. *Praca dyplomowa*, http://www.vehicular.isy.liu.se/en/Publications/MSc/08_EX_4074_TG.pdf, [Dostęp 1-Grudnia-2015]

[3] Dariusz Pazderski, "Odometria jako względna metoda lokalizacji robotów kołowych". *Materiały do wykładów*, http://etacar.put.poznan.pl/dariusz.pazderski/data/RM_naw2.pdf, [Dostęp 1-Grudnia-2015].

[4] Pieter-Jan Van de Maele, "Reading a IMU Without Kalman: The Complementary Filter". *Online*, http://www.pieter-jan.com/node/11, [Dostęp 1-Grudnia-2015].

[5] Koło Naukowe Robotyków "KoNaR", "Regulamin robotycznych zawodów Robotic Arena 2015, Kategoria LineFollower Light". *Online*, http://konar.pwr.edu.pl/index.php/component/phocadownload/category/15-ra-regulaminy?download=161:linefollower-light, [Dostęp 1-Grudnia-2015]

[6] Nota katalogowa mikrokontrolera STM32F405, http://www.st.com/web/en/resource/technical/document/datasheet/DM00037051.pdf [Dostęp 1-Grudnia-2015]

[7] Nota katalogowa żyroskopu L3GD20, http://www.st.com/web/en/resource/technical/document/datasheet/DM00036465.pdf [Dostęp 1-Grudnia-2015]

[8] Nota katalogowa akcelerometru LSM303DLHC, http://www.st.com/web/en/resource/technical/document/datasheet/DM00027543.pdf [Dostęp 1-Grudnia-2015]

[9] Dokumentacja biblioteki HAL dla mikrokontrolerów rodziny STM32 F4, http://www.st.com/st-web-ui/static/active/jp/resource/technical/document/user_manual/DM00105879.pdf [Dostęp 1-Grudnia-2015]

[10] Dokumentacja programu KiCad, http://kicad-pcb.org/help/documentation/ [Dostęp 1-Grudnia-2015]