

POLITECHNIKA WROCŁAWSKA  
WYDZIAŁ ELEKTRONIKI

---

KIERUNEK: AUTOMATYKA I ROBOTYKA  
SPECJALNOŚĆ: ROBOTYKA

PRACA DYPLOMOWA  
INŻYNIERSKA

Projekt robota usługowego do zastosowań  
domowych

Project of a service robot for home purposes

AUTOR:  
Albert Lis

PROWADZĄCY PRACĘ:  
dr inż. Mateusz Cholewiński,  
Wydział Elektroniki, Katedra Cybernetyki i Robotyki

OCENA PRACY:

# Spis treści

1.	Wstęp . . . . .	4
1.1.	Wprowadzenie . . . . .	4
1.2.	Cel i zakres pracy . . . . .	4
2.	Wstęp teoretyczny . . . . .	5
2.1.	Klasyfikacje robotów . . . . .	5
2.2.	Kinematyka robota . . . . .	6
2.3.	Sterowanie silnikiem DC . . . . .	8
2.4.	Optymalizacja kodu . . . . .	9
3.	Konstrukcja . . . . .	10
3.1.	Wybór mikrokontrolera . . . . .	10
3.2.	Pomiar odległości od podłogi . . . . .	11
3.3.	Napęd . . . . .	12
3.4.	Rozpoznawanie pozycji . . . . .	12
3.5.	Zasilanie . . . . .	13
3.6.	Układ pompujący wodę . . . . .	15
3.7.	Pomiar poziomu cieczy . . . . .	15
3.8.	Bezpieczeństwo . . . . .	15
4.	Testy i symulacje . . . . .	17
4.1.	Czas trwania obliczeń . . . . .	17
4.2.	Symulacja działania sterowania kinematycznego . . . . .	18
4.3.	Drgania styków . . . . .	24
4.4.	Filtracja zasilania . . . . .	25
4.5.	Zabezpieczenia tranzystorów . . . . .	26
	Literatura . . . . .	28
	Indeks rzeczowy . . . . .	28

# Skróty

IoT (ang. Internet of Things)

DMA (ang. Direct Memory Acces)

ADC (ang. Analog-to-digital converter)

PLL (ang. Phase-locked loop)

API (ang. Application programming interface)

HAL (ang. Hardware Abstraction Layer)

SPL (ang. Standard Peripheral Libraries)

USB (ang. Universal Serial Bus)

IR (ang. infrared)

RPM (ang. revolutions per minute)

DC (ang. Direct current)

PWM (ang. Pulse-Width Modulation)

CMOS (ang. Complementary Metal-Oxide Semiconductor)

TTL (ang. Transistor-transistor logic)

BMS (ang. Battery management system)

RAM (ang. random-access memory)

(ang. )

(ang. )

# Rozdział 1

## Wstęp

### 1.1. Wprowadzenie

Robotyka jest obecnie prężnie rozwijającą się dziedziną nauki. Niskie ceny mikrokontrolerów oraz duża konkurencyjność firm na rynku powodują przenikanie urządzeń robotycznych z zastosowań specjalnych do życia codziennego. Urządzenia te, mogą oszczędzać zasoby ludzkie w codziennych prostych czynnościach. Dodatkowo projekty takie jak Arduino [2] pozwalają na tworzenie tych urządzeń bez specjalistycznej wiedzy. Kolejnym czynnikiem dynamizującym popularyzację automatyzacji i robotyki jest stopniowe wprowadzanie sieci 5G [7]. Pozwoli ona na wykorzystanie potencjału IoT oraz znaczną automatyzację działania urządzeń robotycznych. W tej pracy skupiono się na budowie urządzenia wspomagającego prace sprzątające.

Na rynku istnieje wiele konstrukcji robotów sprzątających. Skupiają się one głównie na pracy jako odkurzacze. Natomiast liczba autonomicznych robotów myjących podłogi jest zdecydowanie mniejsza. Głównie są to proste roboty mopujące. Robot przedstawiony w pracy ma za zadanie wypełnić lukę między małymi i prostymi urządzeniami, a dużymi do zastosowań profesjonalnych.

### 1.2. Cel i zakres pracy

Celem jest zaprojektowanie, zbudowanie i zaprogramowanie autonomicznego robota myjącego podłogi do zastosowań niekomercyjnych. Robot powinien:

1. Posiadać wirujące szczotki myjące podłogi
2. Posiadać system dozujący wodę z detergentem
3. Posiadać system zbierający zużytą wodę z podłogi
4. Potrafić rozpoznawać i omijać przeszkody
5. Potrafić rozpoznawać niebezpieczne różnice wysokości podłogi
6. Posiadać system jazdny pozwalający łatwo osiągać zadane położenia
7. Posiadać możliwość określania swojej bieżącej pozycji
8. Sygnalizować zdarzenia nadzwyczajne
9. Posiadać system ładowania baterii

## Rozdział 2

# Wstęp teoretyczny

### 2.1. Klasyfikacje robotów

Robot mobilny - robot, który potrafi zmieniać swoje położenie w przestrzeni. Może być robotem autonomicznym, czyli takim który realizując swoje zadanie porusza się bezkolizyjnie w wyznaczonym środowisku oraz robi to bez ingerencji operatora.

Roboty mobilne można podzielić ze względu na ich mobilność:

- kołowe
- kroczące
- latające
- pływające

Z kolei w robotach kołowych możemy rozróżnić następujące klasy:

- (3,0) - robot posiadający 3 koła szwedzkie. Najczęściej spotykany w formie trójkątnej platformy z przymocowanymi kołami do wierzchołków trójkąta. Jego zaletą jest możliwość poruszania się w dowolnym kierunku. Natomiast wadą trudne sterowanie.
- (2,1) - robot posiada 2 koła kastora z tyłu oraz jedno obrotowe, napędowe z przodu. Zaletą jest prostota konstrukcji natomiast wadą trudne sterowanie oraz wrażliwość na nierówności terenu.
- (2,0) - robot tzw. unicycle. Posiada 2 niezależnie napędzane koła, ustawione naprzeciwko siebie. Są one przymocowane na sztywno do ramy. Robot ten musi posiadać także punkt podparcia, którym najczęściej jest koło kastora. Plusami takiego rozwiązania jest prostota sterowania oraz konstrukcji. Dodatkowo robot posiada możliwość obrotu w miejscu. Minusem natomiast jest jego wrażliwość na nierówności.
- (1,2) - robot posiada 2 koła napędowe, skrętne oraz jedno koło kastora służące za podporę. Konstrukcja wymaga zastosowania 3 silników, 2 do zmiany kąta każdego z kół oraz jeden napędowy. Tak więc minusem takiego rozwiązania jest jego skomplikowane sterowanie
- (1,1) - robot nazywany także samochodem kinematycznym, ponieważ zachowuje się podczas sterowania jak zwykły samochód. Jego plusem jest niska wrażliwość na nierówności powierzchni, a jego minusem duży promień skrętu.

Jedynym holonomicznym z wyżej wymienionych klas robotów mobilnych kołowym, jest klasa (3,0). Natomiast po uwzględnieniu wymagań stawianych robotowi zdecydowano się wybrać robota klasy (2,0).

## 2.2. Kinematyka robota

Aby móc sterować robotem należy wyznaczyć kinematykę robota. Korzystając z wniosków zawartych w [10] można przyjąć, że klasa robotów (2,0) jest nieholonomiczna. To znaczy, że ruch takiego robota podlega pewnym ograniczeniom i nie wszystkie trajektorie mogą zostać zrealizowane. Autor wyjaśnia że, takie ograniczenia mogą wynikać z konstrukcji samego robota (wtedy są ograniczeniami wewnętrznymi). Przykładem może być ograniczenie skrętu układu kierowniczego. Ograniczenia mogą pochodzić również od otoczenia w którym robot się porusza. Takie ograniczenia noszą miano ograniczeń holonomicznych. Innym rodzajem ograniczeń są ograniczenia nieholonomiczne. Mamy do czynienia z nimi wtedy, gdy prędkości układu są ograniczone do liczby sterowań przy braku ograniczeń konfiguracji układu. Przykładem może być parkowanie równoległe samochodu. Nie jesteśmy w stanie przesunąć auta prostopadle do drogi (ograniczenie) ale możemy odpowiednio nim sterując zaparkować (osiągnąć dostępną konfigurację).

Zarówno ograniczenia holonomiczne jak i nieholonomiczne można przedstawić w postaci Pfaffa 2.1.

$$A(q)\dot{q} = 0, \quad (2.1)$$

gdzie:

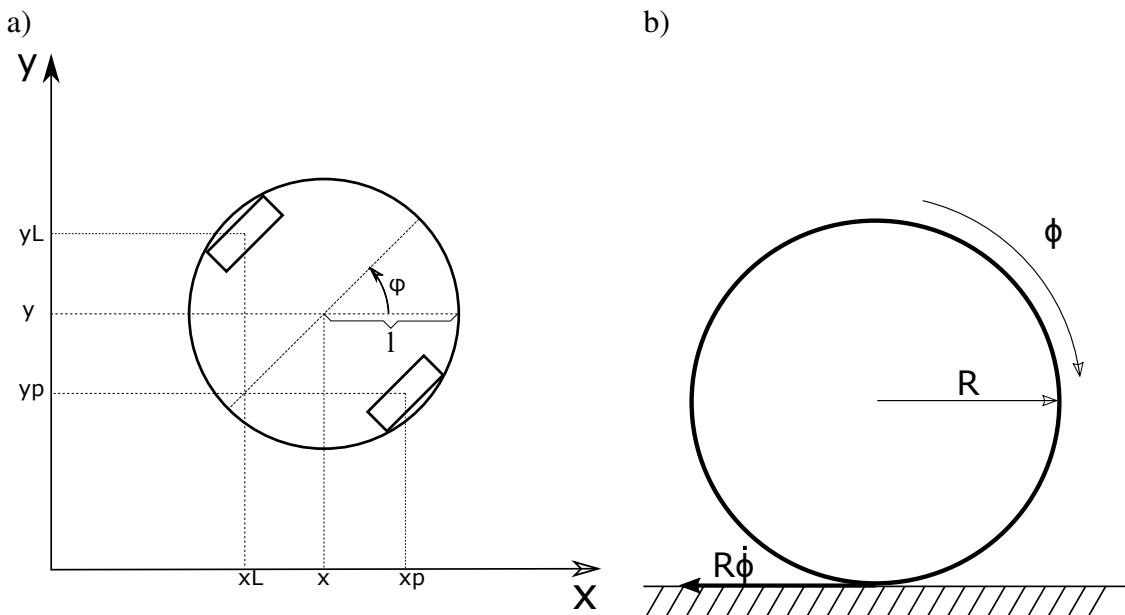
$A(q)$  - macierz pełnego rzędu

$q(t)$  - trajektoria układu

Dla robotów mobilnych kołowych ograniczenia nieholonomiczne w tej postaci wynikają z przyjętych założeń o braku poślizgu kół w miejscu ich styku z podłożem. Opierając obliczenia na [10] i [9] dla monocykla można przyjąć:

- Ograniczenie ruchu poprzecznego (poślizg prostopadły do kierunku ruchu)  
Wyrażone w postaci równania 2.2
- Ograniczenie ruchu wzdłużnego (boksowanie kół)  
Wyrażone osobno dla lewego i prawego koła przedstawiono w postaci równań 2.4 oraz 2.3.

Schematy ideowe zostały pokazane na rysunku 2.1.



Rys. 2.1: Schematy: a) monocykla i b) koła

$$\dot{x} \sin(\varphi) - \dot{y} \cos(\varphi) = 0 \quad (2.2)$$

$$\dot{x}_L \cos(\varphi) + \dot{y}_L \sin(\varphi) - R\dot{\phi}_1 = 0 \quad (2.3)$$

$$\dot{x}_P \cos(\varphi) + \dot{y}_P \sin(\varphi) - R\dot{\phi}_2 = 0 \quad (2.4)$$

Równania 2.4 i 2.3 można przedstawić we współrzędnych globalnych 2.5 i 2.6.

$$\dot{x} \cos(\varphi) + \dot{y} \sin(\varphi) - l\dot{\varphi} - R\dot{\phi}_1 = 0 \quad (2.5)$$

$$\dot{x} \cos(\varphi) + \dot{y} \sin(\varphi) + l\dot{\varphi} - R\dot{\phi}_2 = 0 \quad (2.6)$$

Robota przedstawionego w współrzędnych globalnych można przedstawić za pomocą wektora położeń

$$q = (x \ y \ \varphi \ \phi_1 \ \phi_2)^T \in R^5$$

Który po zróżniczkowaniu daje wektor prędkości 2.7.

$$\dot{q} = (\dot{x} \ \dot{y} \ \dot{\varphi} \ \dot{\phi}_1 \ \dot{\phi}_2)^T \in R^5 \quad (2.7)$$

Z kolei ograniczenia 2.2, 2.5 i 2.6 można wyrazić w postaci macierzy ograniczeń Pfaffa 2.8.

$$A(q) = \begin{bmatrix} \sin(\varphi) & -\cos(\varphi) & 0 & 0 & 0 \\ \cos(\varphi) & \sin(\varphi) & -l & -R & 0 \\ \cos(\varphi) & \sin(\varphi) & l & 0 & -R \end{bmatrix} \quad (2.8)$$

**Szerzej Wyjaśnić pominięcie ograniczenia poprzecznego** Dla robota mobilnego klasy (2,0) można pominąć ograniczenie ruchu poprzecznego 2.2. Wynika to z m.in z faktu, że robot jest w stanie wykonać obrót w miejscu i osiągnąć zadaną pozycję. Wtedy możemy pominąć pierwszy wiersz macierzy 2.8. Przyjmuje ona wtedy postać 2.9.

$$A(q) = \begin{bmatrix} \cos(\varphi) & \sin(\varphi) & -l & -R & 0 \\ \cos(\varphi) & \sin(\varphi) & l & 0 & -R \end{bmatrix} \quad (2.9)$$

Dopuszczalne prędkości można wyrazić jako kombinację wektorów rozpinających jądro macierzy  $A(q)$ . Wektory te dają macierz  $G(q)$ , którą oblicza się z równania 2.10.

$$A(q)G(q) = 0, \quad (2.10)$$

Następnie można zdefinować bezdryfowy układ sterowania w postaci 2.11.

$$\dot{q} = G(q)\eta \quad (2.11)$$

Który po uwzględnieniu ograniczeń 2.8 przyjmuje postać 2.12. **Sprawdzić poprawność macierzy G**

$$\dot{q} = G(q)\eta = \begin{bmatrix} \cos(\varphi) & \cos(\varphi) \\ \sin(\varphi) & \sin(\varphi) \\ \frac{1}{l} & -\frac{1}{l} \\ \frac{2}{R} & 0 \\ 0 & \frac{2}{R} \end{bmatrix} \begin{pmatrix} \eta_1 \\ \eta_2 \end{pmatrix} \quad (2.12)$$

Z dwóch ostatnich równań wynika, że prędkości  $\eta$  są jedynie przeskalowanymi prędkościami kół

$$\dot{\phi}_1 = \frac{2}{R}\eta_1, \quad \dot{\phi}_2 = \frac{2}{R}\eta_2$$

Dzięki temu kinematykę da się przedstawić w analogicznej formie 2.13.

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\varphi} \end{pmatrix} = \begin{bmatrix} \cos(\varphi) & 0 \\ \sin(\varphi) & 0 \\ 0 & 1 \end{bmatrix} \begin{pmatrix} v \\ \omega \end{pmatrix} \quad (2.13)$$

gdzie:

$v = \frac{R}{2}(\dot{\phi}_1 + \dot{\phi}_2)$  – prędkość postępową robota

$\omega = \frac{R}{2l}(\dot{\phi}_1 - \dot{\phi}_2)$  – prędkość kątową robota

Celem robota będzie śledzenie zadanej trajektorii dopuszczalnej, którą dla kinematyki w postaci 2.13 można przedstawić jako 2.14.

$$\begin{pmatrix} \dot{x}_d \\ \dot{y}_d \\ \dot{\varphi}_d \end{pmatrix} = \begin{pmatrix} v_d \cos(\varphi_d) \\ v_d \sin(\varphi_d) \\ \omega_d \end{pmatrix} \quad (2.14)$$

Prędkości liniowa  $v_d$  oraz kątowa  $\omega_d$  zawierają prędkości które powinien mieć układ aby podążał po zadanej trajektorii  $(x_d, y_d, \varphi_d)$ . Następnie, aby móc realizować algorytm sterowania należy zdefiniować referencyjne błędy sterowania 2.15.

$$\begin{pmatrix} x_e \\ y_e \\ \varphi_e \end{pmatrix} = \text{Rot}(Z, -\varphi) \begin{pmatrix} e_x \\ e_y \\ e_\varphi \end{pmatrix} = \begin{bmatrix} \cos(\varphi) & \sin(\varphi) & 0 \\ -\sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x_d - x \\ y_d - y \\ \varphi_d - \varphi \end{pmatrix} \quad (2.15)$$

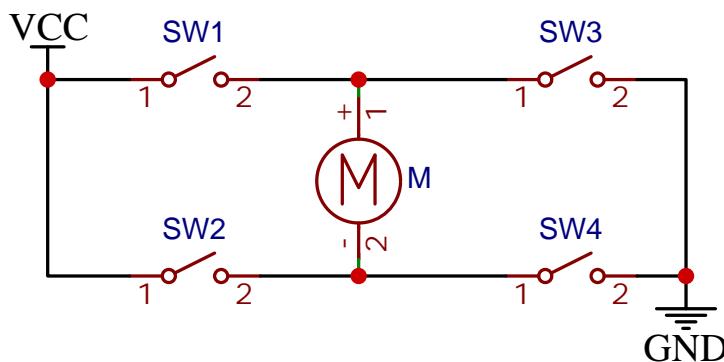
Mając zdefiniowane referencyjne błędy sterowania można przedstawić algorytm sterowania w postaci algorytmu Samsona 2.16.

$$\begin{pmatrix} v_r \\ \omega_r \end{pmatrix} = \begin{pmatrix} k_1 x_e + v_d \cos(\varphi_e) \\ \omega_d + k_2 \varphi_e + v_d y_e \frac{\sin(\varphi_e)}{\varphi_e} \end{pmatrix} \quad (2.16)$$

Za  $\frac{\sin(\varphi_e)}{\varphi_e}$  przyjęto wartość 1.

## 2.3. Sterowanie silnikiem DC

Do sterowania kierunkiem obrotów silnika szczotkowego DC wykorzystuje się układ zwany mostkiem H. Jest on przedstawiony schematycznie na rysunku 2.2.



Rys. 2.2: Schemat mostka H

Styki oznaczone literą S i silnik oznaczony literą M przypominają kształt dużej litery H. Jego działanie polega na zwieraniu odpowiednio styków S1 oraz S4, aby uzyskać kierunek obrotów w jedną stronę. Chcąc uzyskać obroty w drugą stronę należy zewrzeć styki S2 i S3. Należy mieć na uwadze odpowiednie sterowanie, ponieważ załączenie w jednym czasie styków S1 i S3 lub S2 i S4 spowoduje zwarcie.

## 2.4. Optymalizacja kodu

W projekcie wykorzystany został kompilator GNU. Posiada on możliwości optymalizacji kodu wynikowego. Przykładowe flagi optymalizacji przedstawione zostały w tabeli 2.1. Często sto-

Flaga	Działanie
-O0	brak optymalizacji
-Os	optymalizacja rozmiaru kodu
-O1	optymalizacja czasu wykonywania
-O2	większa optymalizacja czasu wykonywania (zawiera -O1)
-O3	jeszcze większa optymalizacja czasu wykonywania (zawiera -O2)
-Ofast	optymalizacja z pominięciem ścisłych standardów (zawiera -O3)

Tab. 2.1: Flagi optymalizacji kodu kompilatora GNU

sowaną flagą optymalizacji na mikrokontrolerów jest -Os. Jednak procesor użyty w tym projekcie posiada wystarczającą ilość pamięci, dlatego zdecydowano się użyć opcji -O3. Pozwoli to zoptymalizować czas obliczeń i skrócić czas wykonywania pętli programu. Szczegółowy opis wymienionych opcji znajduje się na stronie [4].

# Rozdział 3

## Konstrukcja

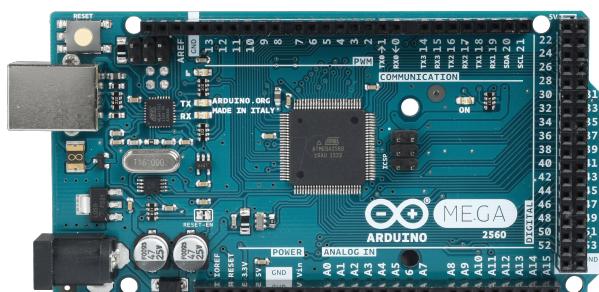
### 3.1. Wybór mikrokontrolera

W przypadku robota autonomicznego istotną jego częścią jest jednostka logiczna, która nim steruje. Powinna być wystarczająco wydajna, aby umożliwić szybkie podejmowanie decyzji na podstawie odczytów z czujników oraz stanu wewnętrznego robota. W przypadku braku zewnętrznego sterowania przez operatora, robot sam powinien unikać kolizji oraz decydować o kierunku poruszania się.

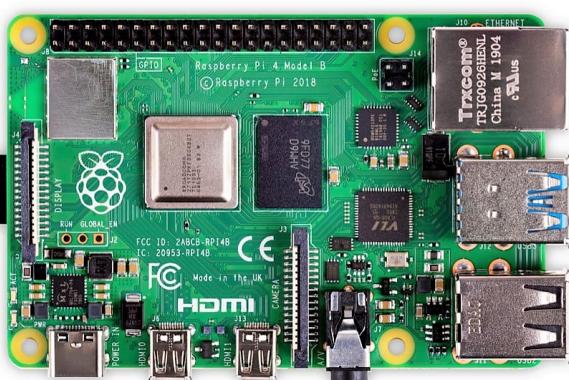
Obecnie dostępnych jest wiele rodzajów mikrokontrolerów, a konkurencyjność zapewnia niskie ceny zakupu. To sprawia, że są one w zasięgu finansowym przeciętnego człowieka. Najbardziej rozpowszechnioną platformą jest seria Arduino [2]. Posiada ona dużą ilość użytkowników i dzięki temu można łatwo uzyskać wsparcie w przypadku problemu z platformą. Przykładową gotową płytą z 8-bitowym procesorem ATmega2560 jest Arduino Mega 3.1. Posiada zegar o maksymalnej częstotliwości 16Mhz, 54 piny cyfrowe (w tym 15 PWM i 6 wspierających przewrącania), 16 pinów analogowych i 6 timerów. Cena w przypadku nieoryginalnej wersji płytki wynosi około 7\$ (28zł)

Kolejnym przykładem rodziny mikrokontrolerów z dużym wsparciem użytkowników jest Raspberry Pi. Przykładową płytą jest Raspberry Pi 4 model B 3.1. Posiada 4-rdzeniowy, 64-bitowy procesor o taktowaniu GHz, od 1 do 4GB pamięci RAM oraz 40 pinów cyfrowych. Zaletą tej płytki jest jej wysoka wydajność i możliwość wgrania pełnoprawnego systemu operacyjnego. Przykładem może być system operacyjny Raspbian [8]. Jest to wersja Linuxa podobna do systemu Debian. Minusem jest cena, która dla najnowszego modelu wynosi około 50\$ (200zł) natomiast dla starszej wersji około 40\$ (160zł).

a)

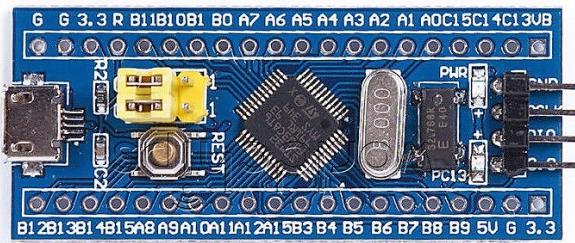


b)



Rys. 3.1: Mikrokontrolery: a) Arduino Mega, b) Raspberry Pi

Trzecią alternatywą łączącą pozytywy obu wymienionych wcześniej platform są mikrokontrolery STM32. Przykładową płytka jest STM32F103 Blue Pill 3.2. Posiada 32 konfigurowalne piny, 16 może obsługiwać zewnętrzne przerwania, 10 pinów połączonych z przetwornikiem ADC. Dodatkowo 18 może pracować z napięciem 5V (sam procesor pracuje na napięciu 3.3V) co może być przydatne zważywszy na fakt, że większa część gotowych modułów pracuje w logice 5V. Procesor wyposażony jest również w kontroler DMA, który pozwala na pomiary ADC oraz komunikację bez użycia procesora. Maksymalna nominalna częstotliwość taktowania wynosi 72MHz i dzięki pętli PLL może być łatwo konfigurowana. W przypadku niewielkiego braku mocy obliczeniowej istnieje możliwość łatwego overclockingu do 128MHz kosztem braku komunikacji przez USB. Procesor wyposażony jest także w 4 timery 16-bitowe, 4-kanałowe. Posiada także kilka możliwości wyboru API. Od wysokopoziomowego STM32duino, opartego na wspomnianym wcześniej Arduino, przez HAL oraz starsze SPL kończąc na systemie czasu rzeczywistego FreeRTOS [3]. Ogromnym plusem jest niska cena. Płytkę kosztuje około 1.5\$ (6zł).

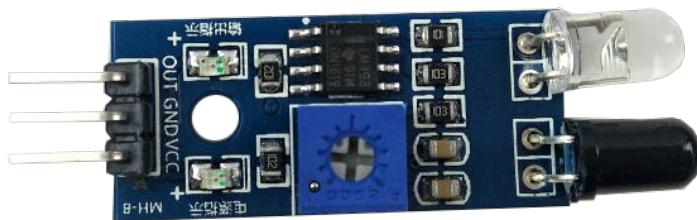


Rys. 3.2: Mikrokontroler STM32F103 Blue Pill

Uwzględniając wskazane powyżej informacje wybrano płytę STM32F103 Blue Pill. Zapewnia dobry bilans między ilością peryferiów i wydajnością w porównaniu do pozostałych możliwości. Dodatkowo cechuje się najniższą ceną zakupu.

## 3.2. Pomiar odległości od podłogi

Aby uniemożliwić robotowi wjazd w niebezpieczny obszar i tym samym zapobiec m.in. spadnięciu ze schodów. Zastosowano czujniki odbiciowe IR mierzące aktualną odległość od podłogi Rys 3.3. Czujnik ten ma wbudowany komparator LM393 regulowany potencjometrem. W przypadku detekcji zbyt dużej odległości na pin sygnałowy wystawiany jest odpowiedni stan logiczny. Dzięki temu nie ma potrzeby robienia pomiarów ADC oraz obsługi sprzętowej. Pozwala to dzięki mechanizmowi przerwań mikrokontrolera na praktycznie natychmiastową reakcję w przypadku wykrycia niebezpieczeństwa.



Rys. 3.3: Odbiciowy czujnik IR

### 3.3. Napęd

Jako napęd robota wykorzystano silniki szczotkowe DC z wbudowaną przekładnią. Parametry znamionowe silnika przedstawione są w tabeli 3.1.

Parametr	Wartość
Napięcie zasilania	3 – 6 V
Prędkość obrotowa po redukcji (6 V)	100 RPM
Pobór prądu pod obciążeniem (6 V)	350 mA
Pobór prądu bez obciążenia (6 V)	170 mA
Siła ciągu na przekładni (6 V)	5.5 kg/cm

Tab. 3.1: Parametry znamionowe silników napędowych

Do sterowania silnikami wykorzystano mostek H zbudowany w oparciu o układ L9110S. Pozwala on na sterowanie zarówno prędkością jak i kierunkiem obrotów silników niezależnie od siebie. Parametry znamionowe mostka przedstawione są w tabeli 3.2.

Parametr	Wartość
Napięcie zasilania	5 – 12 V
Maksymalne napięcie zasilania silników	12 V
Maksymalny prąd na kanał	800 mA
Ilość kanałów	2
Poziom sygnałów sterujących mostkiem	CMOS/TTL

Tab. 3.2: Parametry znamionowe mostka H

Sygnały jakie trzeba podać na wejście sterownika, aby uzyskać pożądane sterowanie zostały przedstawione w tabeli 3.3.

1A	1B	Stan silnika
0	0	Wyłączenie zasilania
1	0	Do przodu
0	1	Do tyłu
1	1	Hamowanie silnikiem?

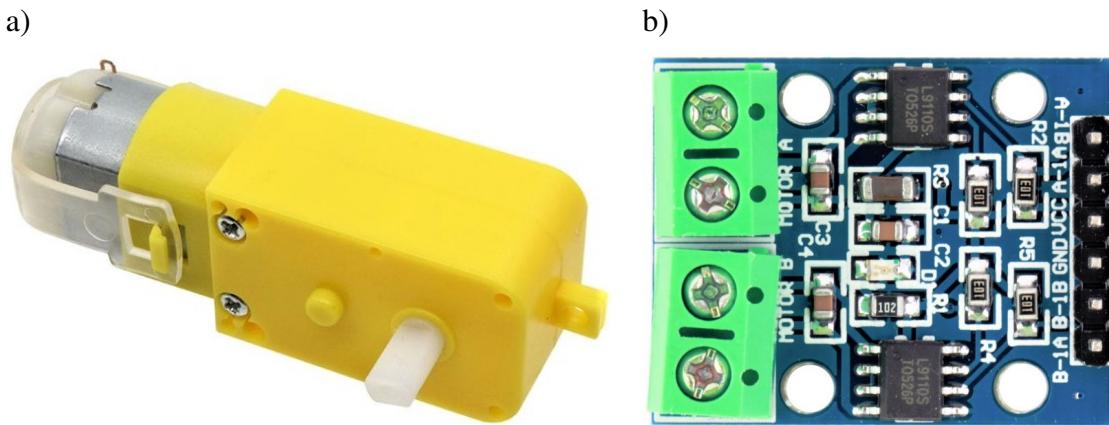
Tab. 3.3: Tabela prawdy mostka H

Silnik oraz mostek H przedstawione są na rysunku 3.4

Do przeniesienia napędu wykorzystane zostały koła o średnicy 68 mm. Przy maksymalnej prędkości obrotowej silnika robot osiąga prędkość wynoszącą w przybliżeniu 42.7 cm/s.

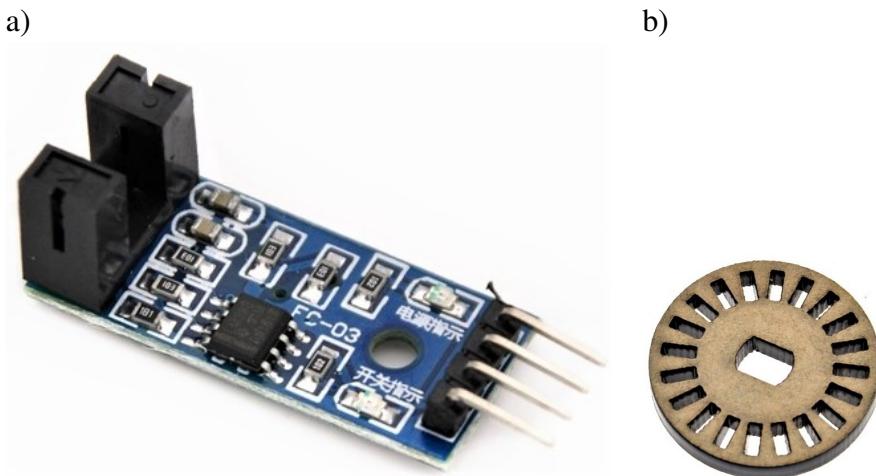
### 3.4. Rozpoznawanie pozycji

Do obliczania aktualnej pozycji i orientacji robota wykorzystanie zostały czujniki szczelinowe w połączeniu z płytami szczelinowymi. Czujnik optyczny posiada wbudowany komparator LM393 dzięki czemu istnieje prosta możliwość detekcji czy sygnał optyczny jest odbierany czy nie. W połączeniu z płytą szczelinową mocowaną na wale silnika daje to prosty enkoder impulsowy. Płyta szczelinowa posiada rozdzielcość 20 linii na obrót. Obsługując impulsy w przewraniach mikrokontrolera oraz wyzwalając przerwanie zarówno na zboczu narastającym jak i



Rys. 3.4: Zdjęcia: a) Silnika DC, b) Mostka H

opadającym można osiągnąć rozdzielcość 40 impulsów na obrót. Uwzględniając średnicę koła niepewność pozycjonowania wynosi w przybliżeniu 5.34 mm. Jest ona zbyt duża, aby zapewnić precyzyjne sterowanie robotem. Dlatego zdecydowano się na zbudowanie przekładni zwiększającej ilość obrotów płytki enkodera. Przekładnia jest dwustopniowa z 5-krotnym wzmacnieniem w każdym ze stopni. Ostatecznie osiągamy 25-krotne zwiększenie prędkości obrotowej enkodera. Daje to nam 1000 impulsów na obrót i niepewność pozycjonowania wynoszącą w przybliżeniu 0.21 mm. Czujnik szczelinowy wraz z płytą enkodera zostały przedstawione na rysunku 3.5.



Rys. 3.5: Zdjęcia: a) Czujnik szczelinowy, b) Płytką szczelinowa

## 3.5. Zasilanie

Jako zasilanie wykorzystanie 2 akumulatory Li-Ion 18650 połączone szeregowo. Napięcie nominalne takiego pakietu wynosi 7.4 V. Przewagą takiego rozwiązania względem połączenia równoległego jest zmniejszenie prądów płynących w przewodach zasilających. Przyczyni się to do mniejszych strat na rezystancjach połączeń oraz zwiększy żywotność przełącznika bistabilnego On/Off. Aby dopasować napięcie do zasilania mikrokontrolera oraz peryferiów wykorzystano 2 przetwornice obniżające napięcie zbudowane w oparciu o układ MP2307. Jedna została wyregulowana na napięcie 3.3 V, natomiast druga na 5 V. Parametry techniczne przetwornic zostały przedstawione w tabeli 3.4.

Parametr	Wartość
Napięcie wejściowe	4.75 – 23 V
Napięcie wyjściowe	1 – 17 V
Maksymalny prąd wyjściowy	Szczytowy 3 A, ciągły 1.8 A
Maksymalna sprawność konwersji	98 %
Częstotliwość przełączania	340 kHz

Tab. 3.4: Parametry znamionowe przetwornicy obniżającej napięcie

Baterie Li-Ion są bardzo wrażliwe zarówno na przeładowywanie [1] jak i nadmierne rozładowanie [5, 6]. Najniższe napięcie rozładowania jakiego nie zaleca się przekraczać wynosi 2.4 V, natomiast typową wartością eksploracji jest 3 V [6]. Z drugiej strony nadmierne przeładowywanie również drastycznie wpływa na żywotność akumulatorów. Nominalnym napięciem ładowania jest 4.2 V. Jednak można ładować ogniwą do niższego napięcia zmniejsza się wtedy ich pojemność, ale wzrasta żywotność. Wpływ końcowego napięcia ładowania na ogniwą przedstawiono w tabeli 3.5 w oparciu o informacje ze źródła [1]

Napięcie końcowe	Ilość cykli rozładowywania	Pojemność ogniw
4.30 V	150 – 200	110 – 115 %
4.25 V	200 – 350	105 – 110 %
<b>4.20 V</b>	300 – 500	<b>100 %</b>
4.15 V	400 – 700	90 – 95 %
4.10 V	600 – 1000	85 – 90 %
4.05 V	850 – 1500	80 – 85 %
4.00 V	1200 – 2000	70 – 75 %

Tab. 3.5: Wpływ napięcia końcowego na żywotność akumulatorów Li-Ion

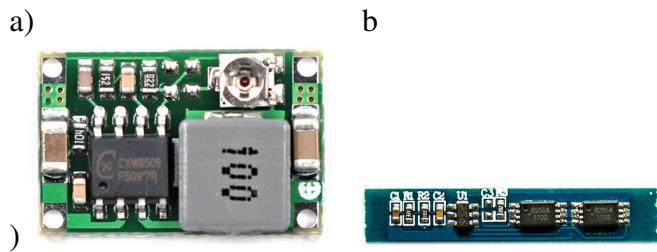
Jak wynika z powyższej tabeli każda różnica 0.1 V powoduje zmianę żywotności o połowę oraz pojemności o około 15 %. W celu ochrony ogniw przed niepoprawną eksploracją zdecydowano się dołączyć układ BMS. Jego parametry techniczne zostały przedstawione w tabeli 3.6. Natomiast przetwornica wraz z modułem BMS zostały przedstawione na rysunku 3.6.

Parametr	Wartość
Zabezpieczenie przed nadmiernym ładowaniem	4.25 – 4.35 V $\pm 0.05$ V
Zabezpieczenie przed nadmiernym rozładowaniem	2.3 – 3.0 V $\pm 0.05$ V
Maksymalny prąd ciągły	5 A
Maksymalny prąd szczytowy	7 A
Opór	<45 mΩ

Tab. 3.6: Parametry techniczne modułu BMS

Układ ten posiada zadowalający poziom zabezpieczeń przed nadmiernym rozładowaniem. Jednak wykorzystanie go bezpośrednio do ładowania ogniw mogłoby przyczynić się do znacznego skrócenia żywotności ze względu na wysokie napięcie końcowe ładowania.

Celem pracy nie jest budowa układu ładowającego baterię dlatego zdecydowano się wykorzystać do tego celu zewnętrzną ładowarkę.



Rys. 3.6: Zdjęcia: a) Przetwornica obniżająca napięcie, b) moduł BMS

## 3.6. Układ pompujący wodę

Do pobierania wody ze zbiornika i podawania jej na szczotki została wykorzystana mała pompa odśrodkowa. Jej parametry przedstawione zostały w tabeli 3.7. Sterowanie wydajnością pompy zostało zrealizowane poprzez sygnał PWM pochodzący z mikrokontrolera.

Parametr	Wartość
Napięcie zasilania	3 – 6 V
Prąd maksymalny	140 mA
Wydajność maksymalna	100 L/h
Hałas maksymalny	40 dB
Stopień ochrony	IP67

Tab. 3.7: Parametry techniczne przetwornicy podnoszącej napięcie

## 3.7. Pomiar poziomu cieczy

Pomiar poziomu cieczy odbywa się w pojemniku z wodą czystą. W pojemniku z wodą brudną nie jest konieczny, ponieważ będzie się w nim znajdować taka sama lub mniejsza ilość wody niż w pojemniku czystym. Pomiar zrealizowany jest za pomocą pary przewodów miedzianych połączonych z pinami mikrokontrolera. Wykorzystywany jest tutaj fakt, że woda kranowa posiada rozpuszczone sole mineralne i w efekcie przewodzi prąd. Autor zakłada, że w robocie nie będzie używana woda destylowana. W dalszej części zostanie poruszona kwestia bezpieczeństwa takiego rozwiązania. **Dodać zdjęcia i pomiary prądu płynącego w wodzie**

## 3.8. Bezpieczeństwo

W tabeli 3.8 przedstawiono bezpieczne poziomy napięć w zależności od środowiska pracy. Wy-

Warunki	Napięcie przemienne [V]	Napięcie stałe [V]
Normalne (suche)	50	120
Zwiększonego zagrożenia (wilgotne)	25	60
Ekstremalnego zagrożenia (mokre)	12	30

Tab. 3.8: Wartości bezpieczne poziomów napięć w zależności od warunków

nika z niej że w warunkach mokrych dopuszczalne napięcie uznawane za bezpieczne wynosi 30 V. Pin mikrokontrolera w stanie wysokim posiada potencjał 3.3 V. Natomiast maksymalne

napięcie w układzie wynosi 8.4 V. Nawet w przypadku uszkodzonego sprzętu i pojawiienia się napięcia zasilania w nieuwzględnionym miejscu nie przekroczy ono poziomu w pełni bezpiecznego dla człowieka.

## Rozdział 4

# Testy i symulacje

### 4.1. Czas trwania obliczeń

W trakcie projektowania sterowania robota warto uwzględnić wydajność obliczeniową mikrokontrolera. Do obliczenia sterowania wymagana jest duża ilość obliczeń zmiennoprzecinkowych. Obliczenia te pochłaniają większą ilość cykli procesora niż obliczenia całkowitoliczbowe. Aby mieć pewność, że obliczenia zostaną wykonane na czas zdecydowano się porównać różnice wynikające z tego faktu. Wzorzec funkcji testującej został przedstawiony w 4.1.

Listing 4.1: Wzorzec funkcji testującej czas wykonywania obliczeń

```
template <typename T>
u32 calculate(T nr1, T nr2) {
    T temp;
    u32 timerStart = micros();
    for(u32 counter{0}; counter < 1000000; ++counter) {
        temp = nr1 * nr2;
        nr1 += temp;
        nr2 += temp + nr1;
    }
    return micros() - timerStart;
}
```

Test przeprowadzono dla wbudowanych typów `int`, `float` oraz `double`. Każdy test został wykonany 5 razy. Wykonano testy z wyłączoną optymalizacją oraz optymalizacją `-O3`. Wyniki dla flag `-O0` przedstawiono w tabeli 4.1.

Numer testu	Czas trwania obliczeń [ms]		
	int	float	double
1	529,446	3482,773	4332,720
2	529,470	3482,749	4332,743
3	529,467	3482,749	4332,743
4	529,469	3482,751	4332,743
5	529,467	3482,749	4332,744

Tab. 4.1: Czas trwania obliczeń z wyłączoną optymalizacją (flaga O0)

Niestety dla optymalizacji prędkości działania za pomocą flagi `-O3` nie udało się otrzymać wiarygodnych wyników przedstawioną funkcją. Ze względu na uproszczenia jakich dokonał kompilator, niezależnie od ilości powtórzeń pętli otrzymywany wynik wynosi 0 µs. Jednak po dodaniu generatora liczb losowych w postaci modyfikacji kodu z

```

temp = nr1 * nr2;
na
temp = nr1 * nr2 * static_cast<T>(random(100000));
dla wszystkich testów osiągnięto wyniki rzędu 1180 ms. Co pozwala twierdzić, że przy użytej
optymalizacji kompilator powinien wystarczająco zoptymalizować kod, tak aby nie wynikały z
tego powodu znaczne opóźnienia.

```

Następne porównanie przeprowadzono po implementacji symulacji sterownika kinematycznego w mikrokontrolerze. Porównano maksymalny czas wykonywania jednej pętli dla flag -O0, -Os i -O3. Wszystkie zmienne, które zawierają części ułamkowe zostały zdefiniowane jako double. Wyniki przedstawiono w tabeli 4.2.

Czas [μs]			
-O0	-Os	-O3	-Ofast
844	452	450	414

Tab. 4.2: Czas wykonywania jednej pętli w zależności od optymalizacji

Dodatkowo porównano wykorzystanie pamięci przez program. Rozmiar pamięci RAM mikrokontrolera to 20 480 B, natomiast Flash 65 536 B.

Pamięć	Wykorzystanie pamięci			
	-O0	-Os	-O3	-Ofast
RAM	22.9 % (4696 B)	22.9 % (4680 B)	22.9 % (4680 B)	22.9 % (4680 B)
Flash	84.6 % (55 424 B)	59.0 % (38 688 B)	62.2 % (40 736 B)	62.2 % (40 736 B)

Tab. 4.3: Wykorzystanie pamięci w zależności od optymalizacji

## 4.2. Symulacja działania sterowania kinematycznego

Przed właściwą implementacją sterowania kinematycznego zdecydowano się sprawdzić jego zachowanie za pomocą symulacji komputerowej. Do symulacji wykorzystano oprogramowanie Matlab wraz z pakietem Simulink. Wynik działania dla trajektorii kołowej pokazano na rysunku 4.1. Parametry  $k_1$  i  $k_2$  to odpowiednio 0.1 oraz 1.

Następnie zasymulowano dwa różne sposoby generowania trajektorii poruszania się po pomieszczeniu. Parametry  $k_1$  i  $k_2$  pozostały bez zmian. Wyniki przedstawiono na rysunku 4.2. Wynika z nich, że robot ma tendencję do zaokrąglania gwałtownych zmian ruchu. A w przypadku 4.2 b) co czwarty zakręt następują załamania rzeczywistego ruchu robota. Po zakończeniu generowania trajektorii robot od razu zatrzymuje się w miejscu bez przeregulowań.

Następnie zdecydowano się na porównać wpływ parametrów  $k_1$  i  $k_2$  na jakość sterowania. Wyniki dla zmian parametru  $k_1$ , przy stałym  $k_2 = 1$  pokazano na rysunku 4.3. Można z nich wnioskować, że parametr ten ma wpływ na przeregulowania podczas zmian trajektorii. Gdy jest zbyt mały, robot skraca zakręty. Natomiast wraz z jego wzrostem, robot ma tendencję do coraz większego zaokrąglania zakrętów. Przy dużych wartościach, powstaje także przeregulowanie na prostym odcinku drogi.

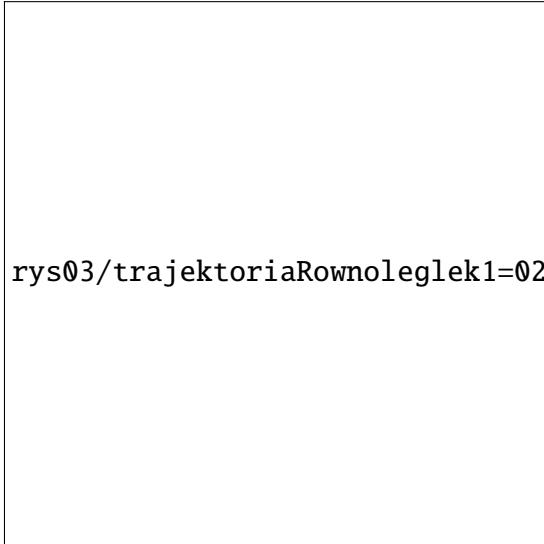
Wyniki dla zmian parametru  $k_2$ , przy stałym  $k_1 = 0.1$  zostały przedstawione na rysunku 4.4. Zwiększenie tego parametru powoduje poprawę jakości podążania za zadaną trajektorią. Gdy jest zbyt mały, robot zbyt wcześnie zwraca i następują załamania trajektorii w przypadku trajektorii równoległej. Natomiast w przypadku trajektorii do środka, następuje zaokrąglenie zakrętów.



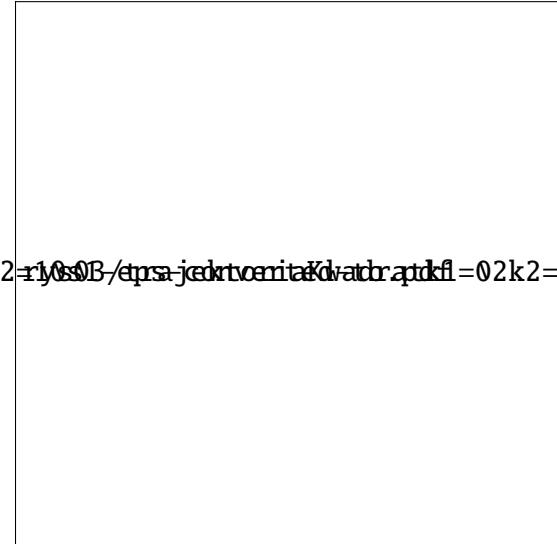
rys03/trajektoriaKoloWykres-eps-converted-to.pdf

Rys. 4.1: Porównanie trajektorii pochodzącej z generatora oraz robota

a)



b)

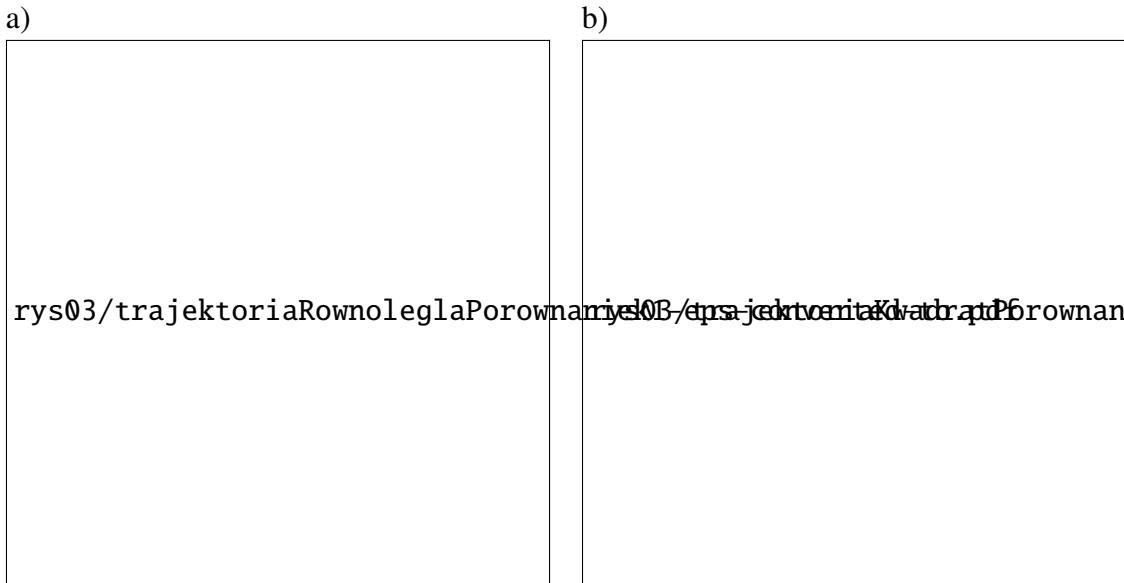


Rys. 4.2: Porównanie dwóch sposobów generowania trajektorii: a) równoległa b) do środka

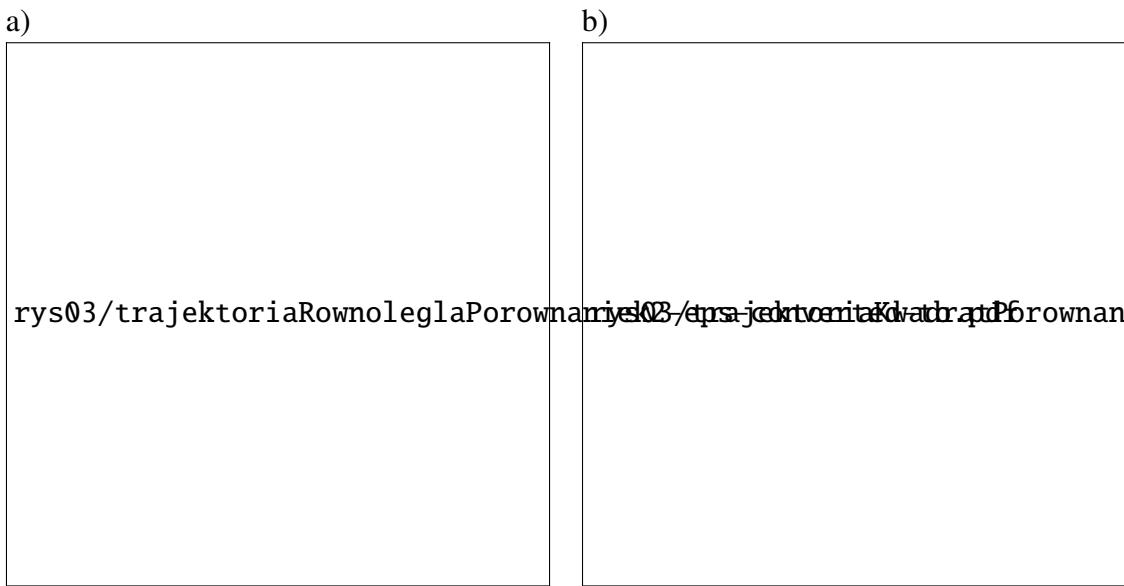
Znając wpływ parametrów  $k_1$  i  $k_2$  można dobrać ich wartości do robota. Aby było to możliwe należy sprawdzić prędkości obrotowe kół. Powinny być możliwe do zrealizowania przez silniki. Jako parametry testowe wybrano  $k_1 = 1$  oraz  $k_2 = 10$ . Prędkość postępowa robota wynosi 1 m/s. Dla obu trajektorii 4.5 i 4.6 prędkości obrotowe kół dla odcinków prostych wynoszą  $\approx 4.7$  obr/s. Natomiast zastosowane silniki według dokumentacji pozwalają na osiągnięcie 100 obr/min =  $1\frac{2}{3}$  obr/s. Oznacza to, że robot nie jest w stanie poruszać się z taką prędkością. Dodatkowo na zakrętach odnotowywane są skoki prędkości obrotowych. W przypadku 4.6b) co czwarty zakręt dochodzi do przeregulowań i gwałtownych skoków prędkości obrotowych.

W pierwszej kolejności zdecydowano się zmniejszyć prędkość postępową robota do 10 cm/s. Zmodyfikowano także trajektorię, tak aby odpowiadała szerokości szczotek myjących podłogę.

Jak pokazuje rysunek 4.7 i 4.8. Prędkość obrotowa na odcinkach prostych zgadnie z oczekiwaniami spadła dziesięciokrotnie do poziomu  $\approx 0.45$  obr/s i mieści się w możliwym do zrealizowania przez silniki przedziale. Natomiast osłabieniu nie uległy prędkości obrotowe kół podczas załamania trajektorii i nadal znacznie przekraczają dopuszczalne przez silniki prędkości. Z tego powodu postanowiono ustawić limity prędkości obrotowych kół w modelu i sprawdzić



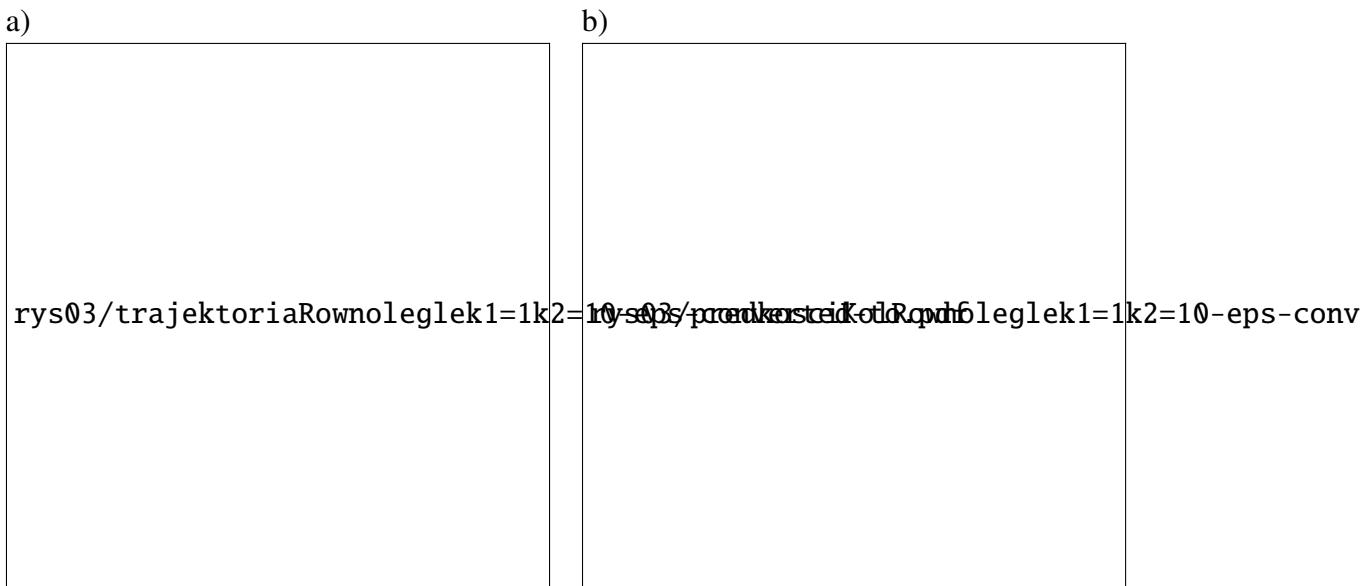
Rys. 4.3: Porównanie wpływu parametru  $k_1$  na trajektorię: a) równoległą b) do środka



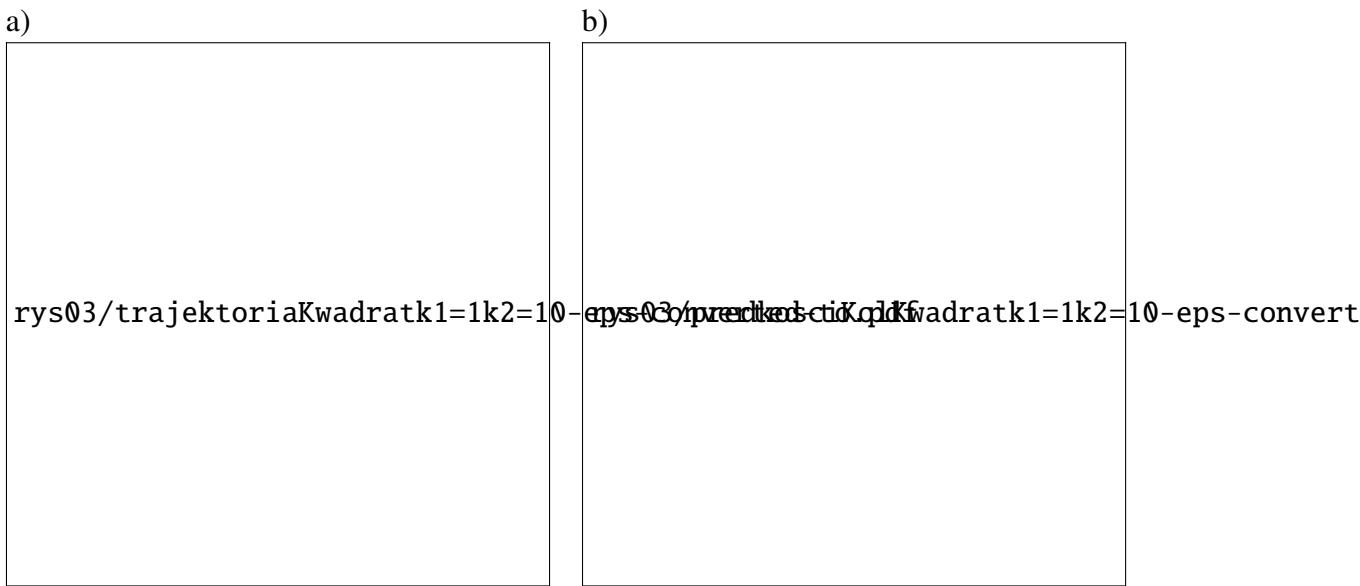
Rys. 4.4: Porównanie wpływu parametru  $k_2$  na trajektorię: a) równoległą b) do środka

ich wpływ na zachowanie robota. Wyniki przedstawiają rysunki 4.9 i 4.10. Widać na nich pogorszenie jakości śledzenia trajektorii. Natomiast maksymalne prędkości kół mieszczą się w zakresie realizowanym przez silniki.

Uwzględniając wszystkie badania można stwierdzić, że kluczowym problemem wprowadzającym błędy do układu są gwałtowne zmiany trajektorii. Dlatego na zakrętach postanowiono wygenerować trajektorię po łuku i sprawdzić wpływ tej zmiany na jakość śledzenia trajektorii. Zmniejszenie gwałtownych zmian trajektorii pozwala także dobrać wyższe wartości parametrów  $k_1$  i  $k_2$ . Wyniki dla  $k_1 = 10, k_2 = 100$  przedstawiają rysunki 4.11 oraz 4.12. Po wprowadzonych zmianach w przypadku trajektorii równoległej jakość realizowanej przez robota trajektorii jest zadowalająca, a prędkości obrotowe kół maksymalnie wynoszą  $\approx 1 \text{ obr/s}$  i mieszczą się w zakresie realizowanym przez silniki z dodatkowym marginesem błędu. Natomiast w przypadku trajektorii do środka rezultat jest nieco gorszy, ale maksymalne prędkości obrotowe kół są o połowę mniejsze niż w trajektorii równoległej.

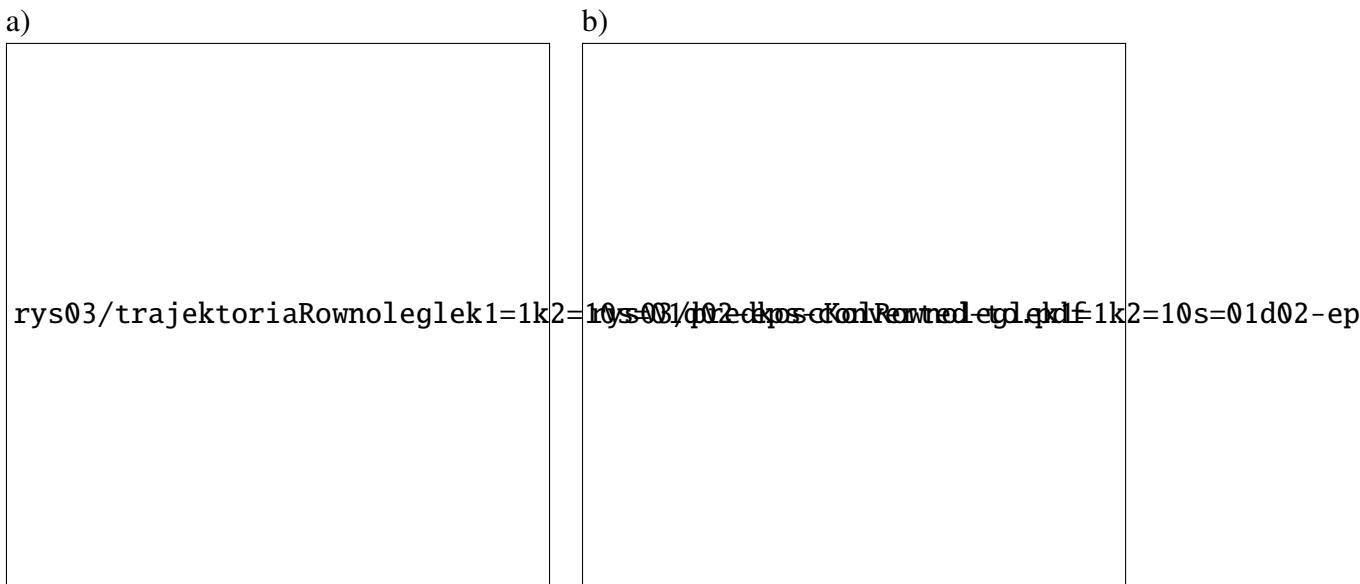


Rys. 4.5: Zestawienie: a) Trajektoria równoległa b) Prędkości kół

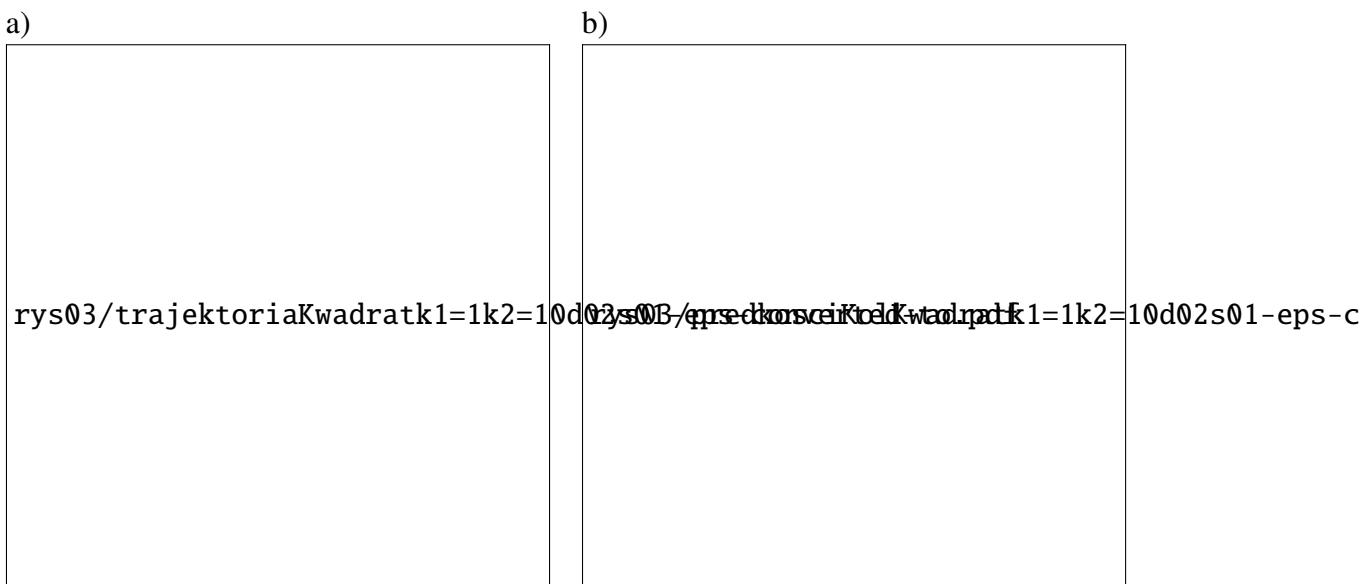


Rys. 4.6: Zestawienie: a) Trajektoria do środka b) Prędkości kół

Na wszystkich symulacjach dla trajektorii do środka można zauważać charakterystyczne przeregulowania podczas zmiany kąta obrotu z  $\frac{3}{2}\pi$  do  $2\pi$ . Aby zbadać to zjawisko postanowiono porównać wartości kątów pochodzących z generatora oraz obiektu. Porównanie zostało pokazane na rysunku 4.13. Można z niego wnioskować, że robot zamiast osiągnąć pełny kąt  $2\pi$  czyli wartość początkową 0, obraca się w drugą stronę i tak osiąga ten kąt. To w połączeniu z ograniczeniami prędkości kół powoduje przedstawiony efekt. Aby rozwiązać ten problem postanowiono zmienić zakres matematycznego zapisu kąta obrotu robota. Zakres  $(0, 2\pi)$  rozszerzono do  $(-\infty, +\infty)$ . Pozwoli to przechowywać pełną informację o ilości obrotów robota i rozwiąże problem powracania do pozycji początkowej. W rzeczywistej implementacji robota zakres ten będzie ograniczony do wartości brzegowych zmiennych w zależności od użytego typu. Trajektoria po wprowadzonej zmianie, wraz z prędkosciami kół przedstawiona na rys 4.14. Widać na nich, że nie występują już charakterystyczne przeregulowania.

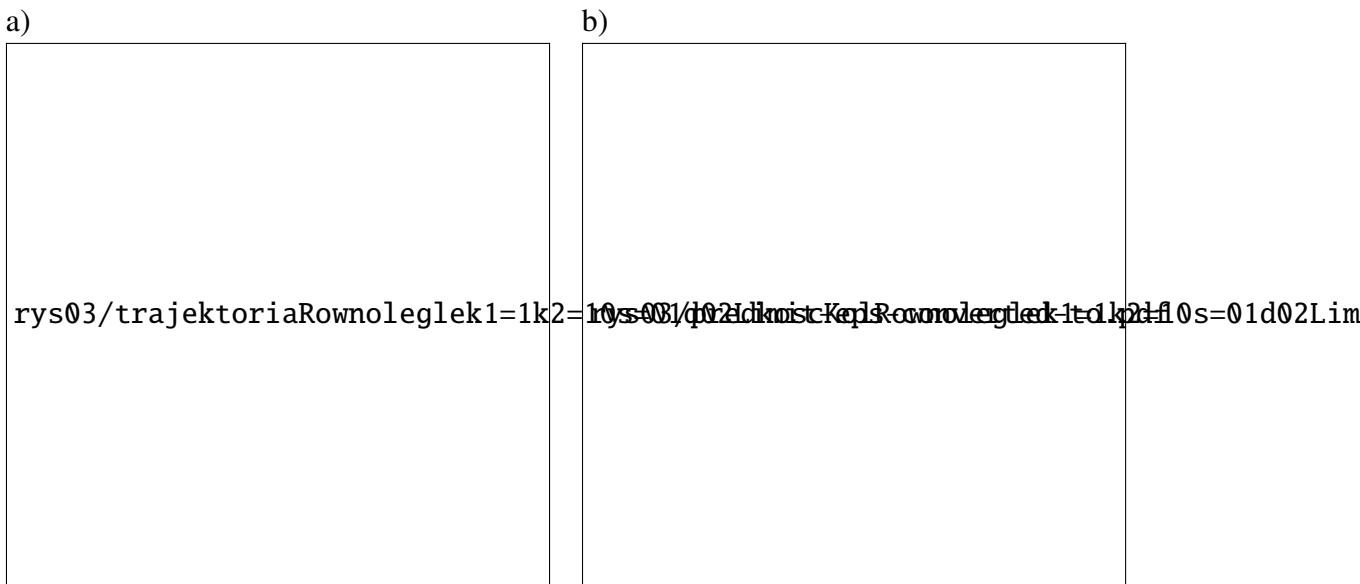


Rys. 4.7: Zestawienie dla prędkości 10 cm/s: a) Trajektoria równoległa b) Prędkości kół

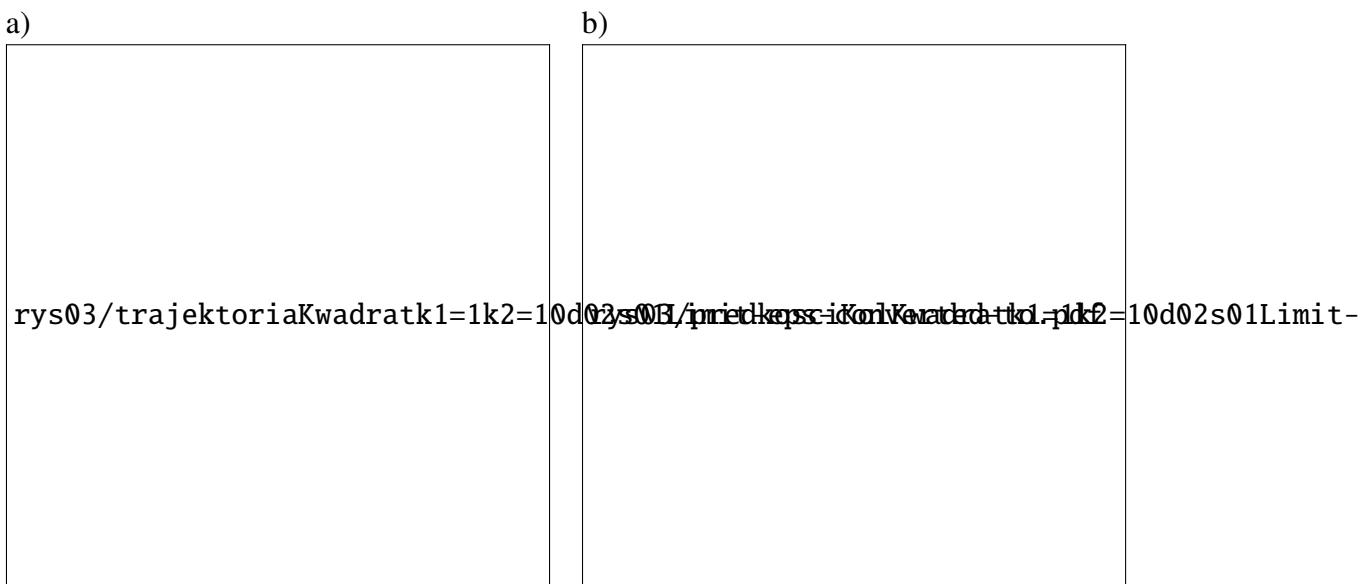


Rys. 4.8: Zestawienie dla prędkości 10 cm/s: a) Trajektoria do środka b) Prędkości kół

Następnie zaimplementowano sterownik kinematyczny bezpośrednio w mikrokontrolerze. Z uwagi na wykorzystany framework `stm32duino` oraz język C++ w standardzie C++11, zdecydowano się na podejście obiektowe. Do celów wizualizacji trajektorii w czasie rzeczywistym postanowiono wykorzystać środowisko `Processing` oparte o język Java. Dodatkowo dane są logowane do pliku `.txt` i wyświetlane za pomocą skryptu w języku Python oraz biblioteki `Matplotlib`. Co umożliwia późniejszą interpretację danych. Czas powtarzania pętli został ustawiony na 10 ms, co daje aktualizację pozycji z częstotliwością 100 powtórzeń/s. Trajektoria z podglądu w środowisku `Processing` została pokazana na rysunku 4.15. Parametry symulacji ustalono na  $k_1 = 1$ ,  $k_2 = 1$ . Kolor biały reprezentuje trajektorię zadaną, natomiast czerwony rzeczywistą. Następnie wyrysowano szczegółowe informacje zawarte w pliku `.txt`. Wyniki zostały przedstawione na rysunkach 4.16 oraz 4.17. Uzyskane wyniki są podobne do tych ze środowiska `Matlab/Simulink`.



Rys. 4.9: Zestawienie z limitem prędkości kół: a) Trajektoria równoległa b) Prędkości kół



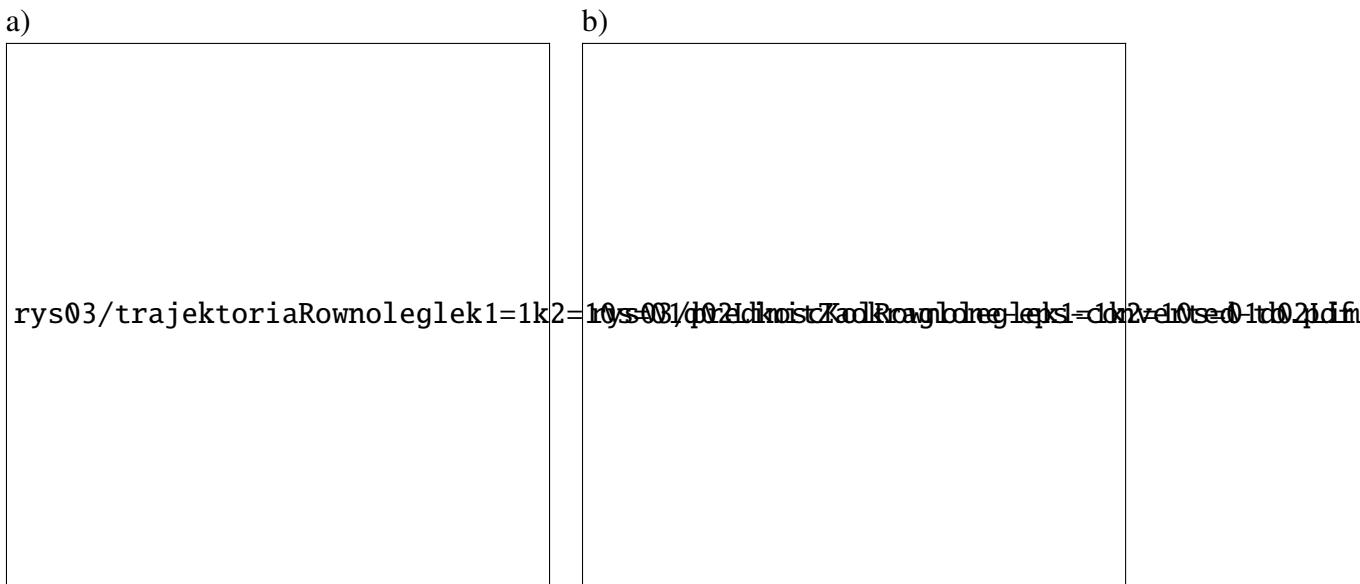
Rys. 4.10: Zestawienie z limitem prędkości kół: a) Trajektoria do środka b) Prędkości kół

### 4.3. Drgania styków

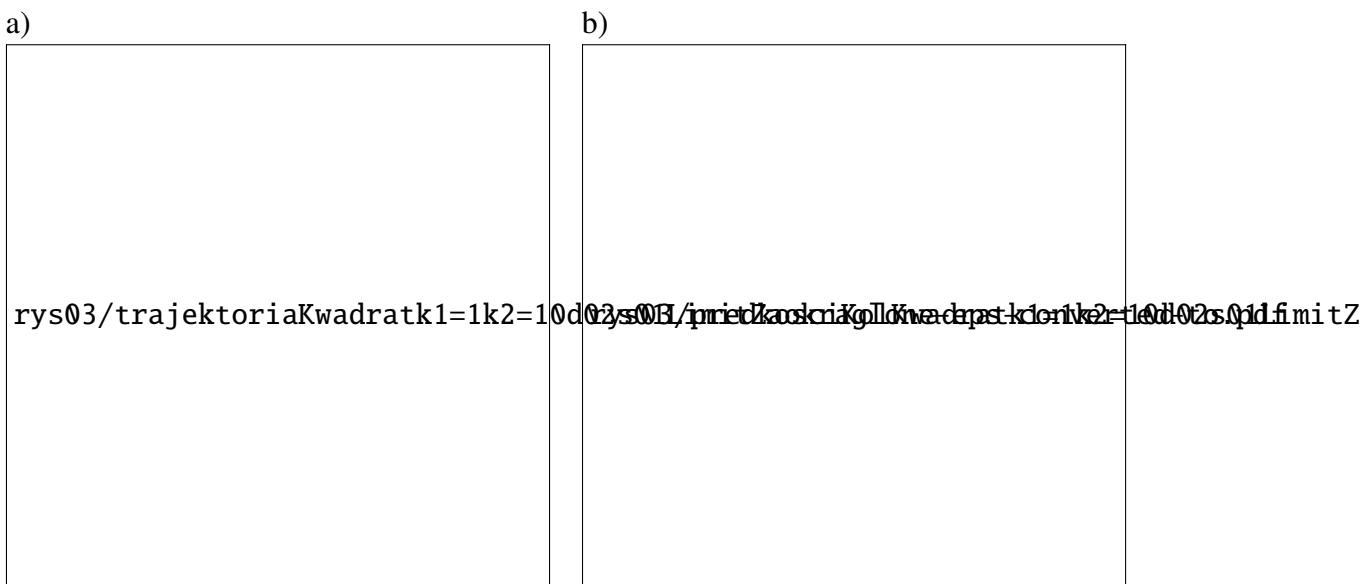
Dla układów stykowych takich jak przyciski, przełączniki etc. istnieje zjawisko drgań styków. Pojawia się ono przy niepełnym kontakcie styków w trakcie włączania/wyłączania i potrafi trwać do 20 ms **dodać źródło**. Zjawisko to powoduje wielokrotne odczyty kliknięć przez mikrokontroler pojedynczej zmiany stanu przycisku. Jednym z rozwiązań sprzętowych jest dodanie filtra RC o odpowiedniej stałej czasowej. Przykład drgań styków oraz zastosowania filtra został przedstawiony na rysunku 4.18.

### 4.4. Filtracja zasilania

Przetwornice DC-DC na wyjściu posiadają pewne nierówności zasilania. Dlatego warto zastosować filtrację w postaci dławików i kondensatorów. Badania przeprowadzono dla przetwornicy



Rys. 4.11: Zestawienie po modyfikacji trajektorii: a) Trajektoria równoległa b) Prędkości kół



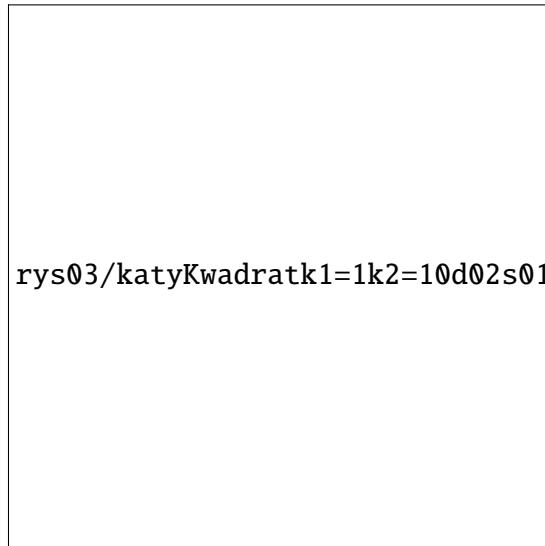
Rys. 4.12: Zestawienie po modyfikacji trajektorii: a) Trajektoria do środka b) Prędkości kół

obniżającej napięcie MP2307. Do filtracji użyto dławika **wartość** oraz kondensatora  $2200 \mu\text{F}$ . Jak widać na rysunku 4.19 filtr znacznie ograniczył częstotliwość wahań napięcia. Jednak nie udało się ich wyeliminować całkowicie i pojawiają się okresowo.

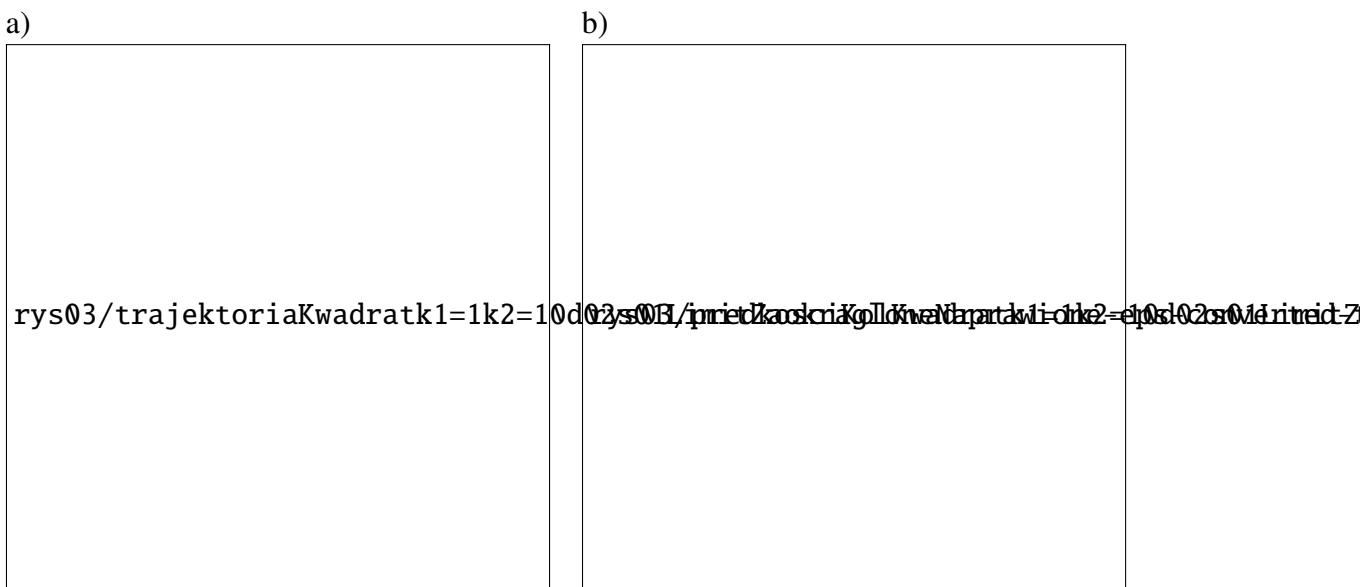
## 4.5. Zabezpieczenia tranzystorów

W przypadku wykorzystywania elementów indukcyjnych m.in silników, należy zwrócić uwagę na ich zachowanie w przypadku zmian prądów w obwodzie. Następuje wtedy zjawisko samoindukcji i w cewce jest generowana siła elektromotoryczna. Zależność indukowanej siły elektromotorycznej od zmiany prądu przedstawia wzór 4.1.

$$E = -L \frac{di}{dt} \quad (4.1)$$



Rys. 4.13: Porównanie kątów obrotu z generatora i obiektu



Rys. 4.14: Wpływ modyfikacji zakresu kątów na: a) Trajektorię b) Prędkości kół

gdzie:

$E$  – siła elektromotoryczna,

$L$  – indukcyjność cewki,

$i$  – natężenie prądu płynącego przez cewkę,

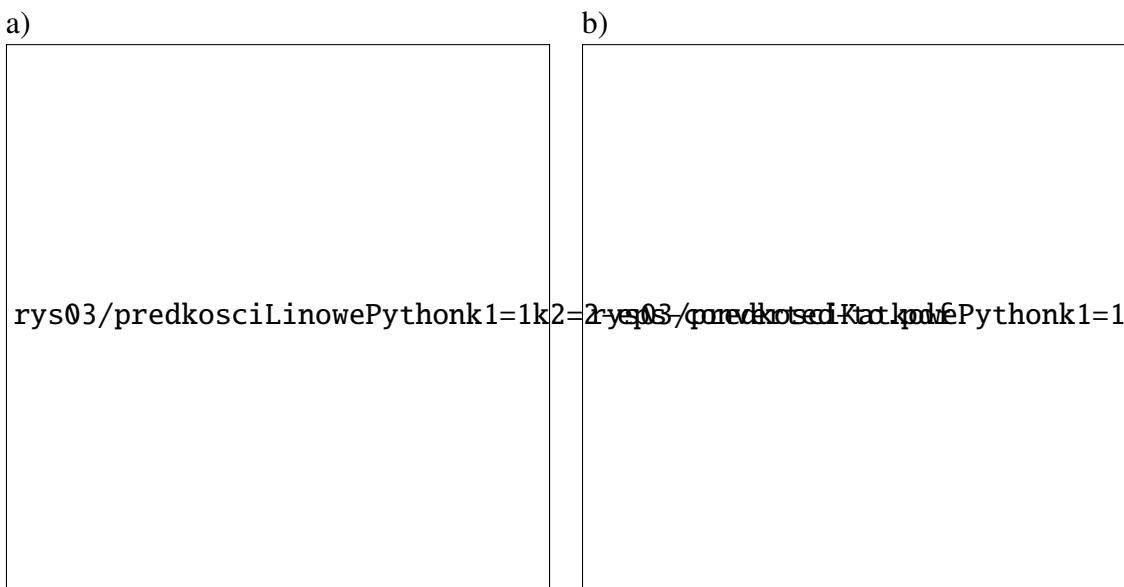
$t$  – czas.

Wynika z niego, że gwałtowne zmiany prądu występujące np. podczas sterowania sygnałem PWM powodują generowanie wysokich impulsów napięcia na cewce. Badania przeprowadzone na pompce wody potwierdzają rozważania teoretyczne. Przebieg napięcia podczas sterowania sygnałem PWM o wypełnieniu 50 % został przedstawiony na rysunku 4.20. Widać na nim gwałtowne impulsy napięcia, znacznie przewyższające wartości sygnału PWM. Takie zachowanie może zniszczyć tranzystor sterujący. Dlatego aby zniwelować ten efekt stosuje się połączoną równolegle do zacisków elementu indukcyjnego diodę Schottkyego, dodatkowo można podłączyć także kondensator. Wyniki dla takich zabezpieczeń zostały przedstawione na rysunku 4.21. Widać na nich ograniczenie przepięcia do wartości napięcia przewodzenia diody.



rys03/trajektoriaProcessingk1=1k2=1.png

Rys. 4.15: Trajektoria rysowana w środowisku Processing

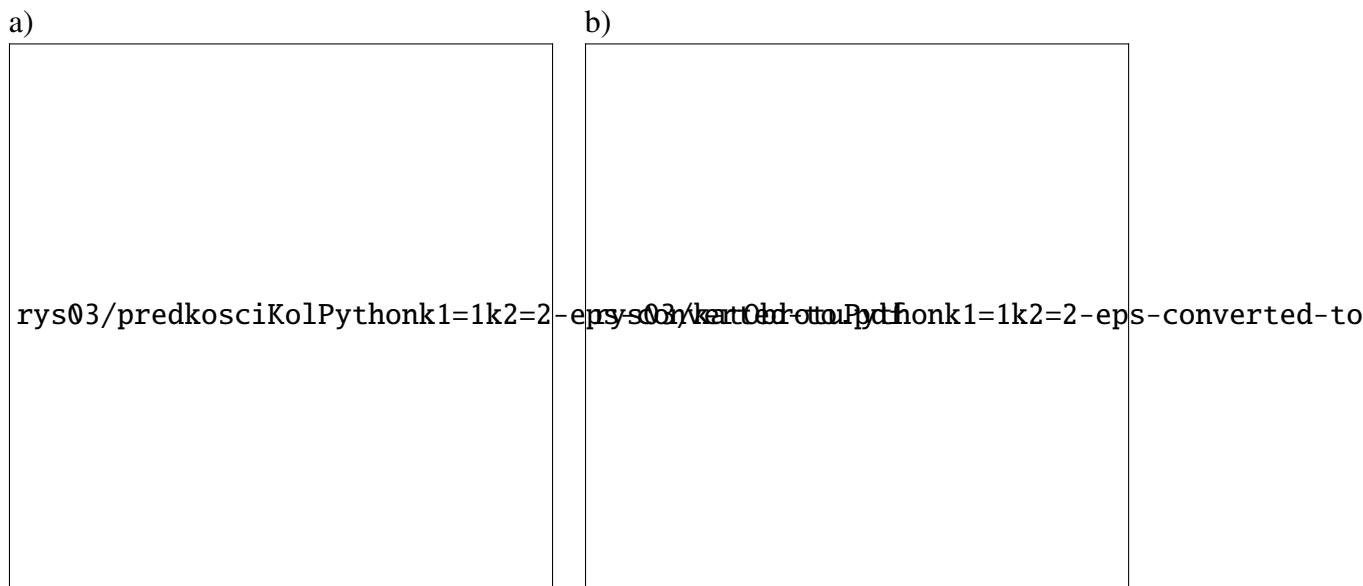


rys03/predkosciLinowePythonk1=1k2=2-eps-convex

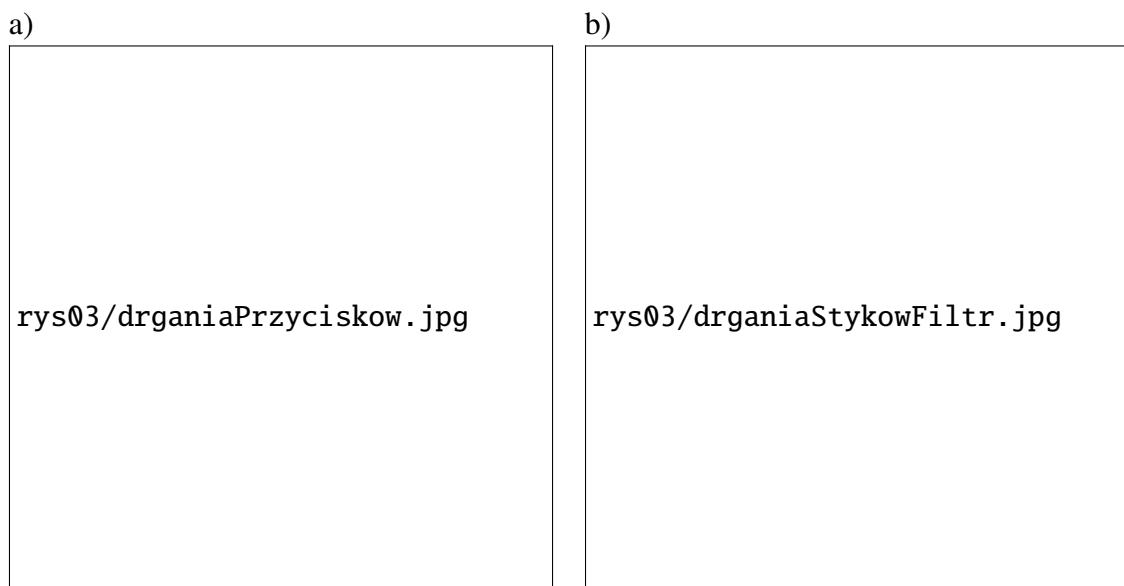
rys03/predkosciKatowePythonk1=1k2=2-eps-convex

Rys. 4.16: Prędkości robota: a) Liniowa b) Kątowa

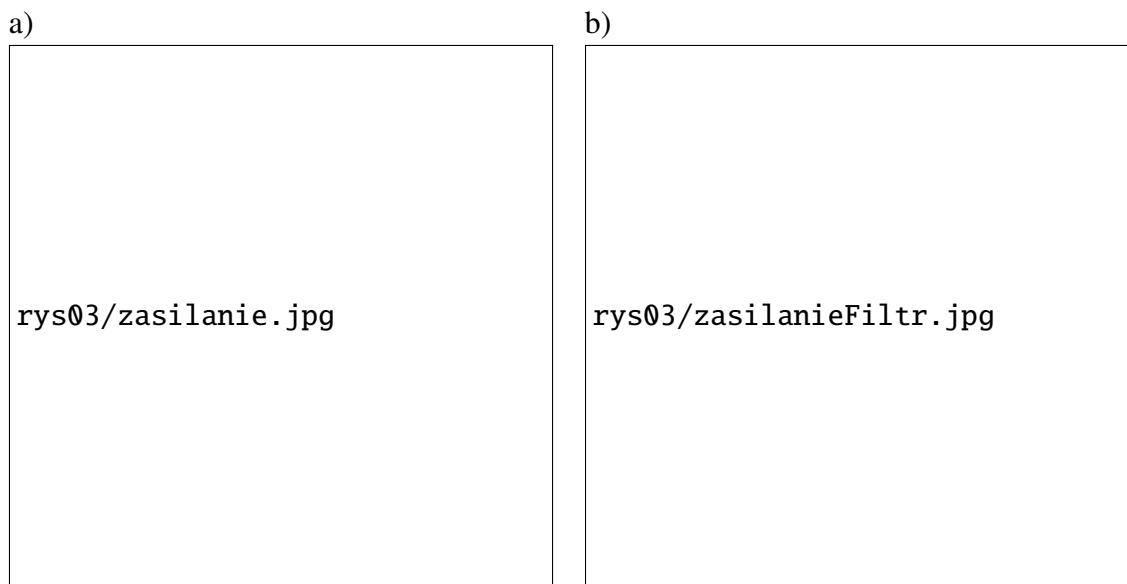
W projekcie zastosowane zostały gotowe moduły sterowania silnikami, które rozwiążają przedstawiony problem.



Rys. 4.17: a) Prędkości kół b) Kąt obrotu



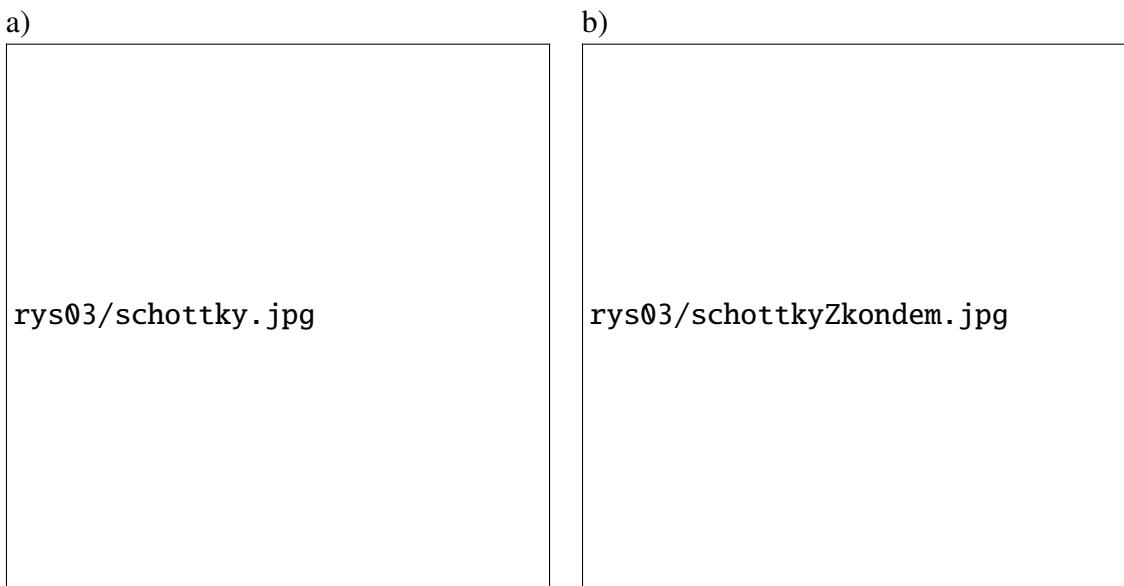
Rys. 4.18: Drania styków a) przed filtracją b) po filtracji



Rys. 4.19: Napięcie zasilania a) przed filtracją b) po filtracji



Rys. 4.20: Napięcie na pompce wody bez filtracji



Rys. 4.21: Napięcie na pompce wody: a) z diodą Schottkyego b) z diodą Schottkyego i kondensatorem

# Literatura

- [1] Ładowanie li-ion. [https://batteryuniversity.com/index.php/learn/article/how\\_to\\_prolong\\_lithium\\_based\\_batteries](https://batteryuniversity.com/index.php/learn/article/how_to_prolong_lithium_based_batteries).
- [2] Arduino framework. <https://www.arduino.cc/>.
- [3] Freertos. <https://www.freertos.org>.
- [4] Opis flag optymalizacji kompilatora gnu.
- [5] Rozładowywania li-ion 1. [https://batteryuniversity.com/learn/article/discharge\\_methods](https://batteryuniversity.com/learn/article/discharge_methods).
- [6] Rozładowywania li-ion 2. [https://batteryuniversity.com/learn/article/discharge\\_characteristics\\_li](https://batteryuniversity.com/learn/article/discharge_characteristics_li).
- [7] Strona polskiego ministerstwa cyfryzacji. <https://www.gov.pl/web/5g>.
- [8] System operacyjny raspbian. <https://www.raspberrypi.org/downloads/raspbian/>.
- [9] aa. Intelligent Systems'2014. aa, 2014.
- [10] A. Mazur. Sterowanie oparte na modelu dla nieholonomicznych manipulatorów mobilnych. a, 2009.