DOI: https://doi.org/10.53052/9788366249974.33

Damian PYTKA[1], Jacek RYSIŃSKI[2]

Opiekun naukowy: Jacek RYSIŃSKI[2]

# PLATFORMA ROBOTA MOBILNEGO Z KOŁOWYM NAPĘDEM OMNIKIERUNKOWYM

**Streszczenie:** W pracy przedstawiono projekt oraz wykonanie platformy robota mobilnego z napędem omnikierunkowym. Przeprowadzono testy układu sterowania oraz układu napędowego. Zbadano magnetometr używany w urządzeniu jako kompas.

**Słowa kluczowe:** robot mobilny, koła omnikierunkowe

# OMNIDIRECTIONAL WHEELED MOBILE ROBOT PLATFORM

**Summary:** In the paper the design and construction of a mobile robot platform with omnidirectional drive is presented. Tests of the control system and the drive system were carried out. The magnetometer used in the device as a compass was investigated.

**Keywords:** mobile robot, omnidirectional wheels

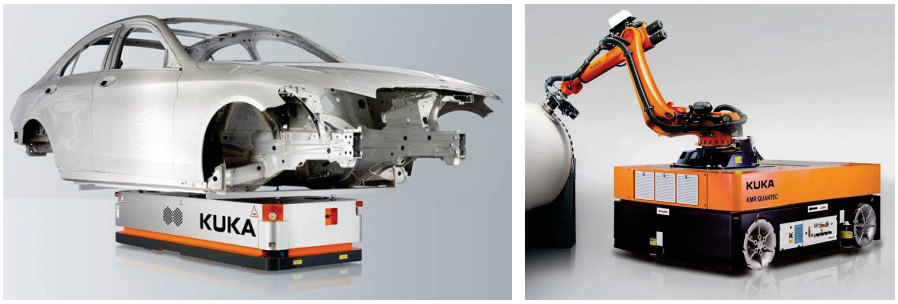
## 1. Wstęp

Napęd omnikierunkowy jest typem napędu holonomicznego – w przypadku takiego rozwiązania występuje związek pomiędzy liczbą stopni swobody dostępnych i kontrolowanych [1, 6, 8-11]. Napęd ten jest wykorzystywany w robotach mobilnych i pojazdach aby zwiększyć ich zwrotność. Znajduje on zastosowanie szczególnie w gęsto zagospodarowanych przestrzeniach, na przykład magazynach. Główna zasada działania kół omnikierunkowych jest oparta na generowaniu ruchu w kierunku równoległym do osi koła za pomocą niezależnych rolek, znajdujących się na obwodzie koła [2].

Przykładowo, firma Kuka ma w swojej ofercie szeroki wachlarz autonomicznych platform omnikierunkowych. W połączeniu z systemem nawigacyjnym, roboty te

[1] inż., Akademia Techniczno-Humanistyczna w Bielsku-Białej, Wydział Budowy Maszyn i Informatyki, email: damianpytka98@gmail.com
[2] dr inż., Akademia Techniczno-Humanistyczna w Bielsku-Białej, Wydział Budowy Maszyn i Informatyki, email: jrysinski@ath.bielsko.pl

tworzą zautomatyzowany system logistyczny. Zastosowanie kół Mecanum pozwala platformom nawigować w wąskich przestrzeniach [3, 4].




*Rysunek 1. Platforma KMP 1500 przewożąca karoserię samochodu [3] oraz KMR QUANTEC - integracja platformy omnikierunkowej z robotem przemysłowym [4]*

Zastosowanie kół omnikierunkowych jako powierzchni przenośnika pozwala na łatwą zmianę orientacji i kierunku obiektów, bez konieczności użycia dodatkowego urządzenia. Przenośniki Omnia mają formę modułów, z których można układać przenośniki, lub punkty rozdzielające [8].

*Rysunek 2. Montaż modułu przenośnika Omnia [8]*

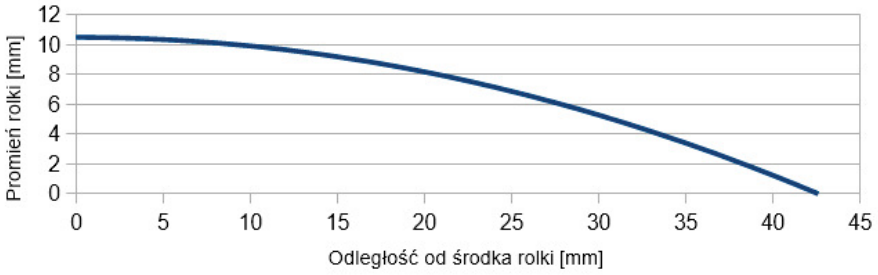
## 2. Projekt oraz wykonanie mobilnej platformy robota

### 2.1. Model 3D platformy robota

Model 3D oraz dokumentację techniczną przygotowano w Fusion 360. W procesie projektowania robota omnikierunkowego użyto głównie modelowania bryłowego. Bardziej skomplikowane kształty wymagały użycia modelowania powierzchniowego.

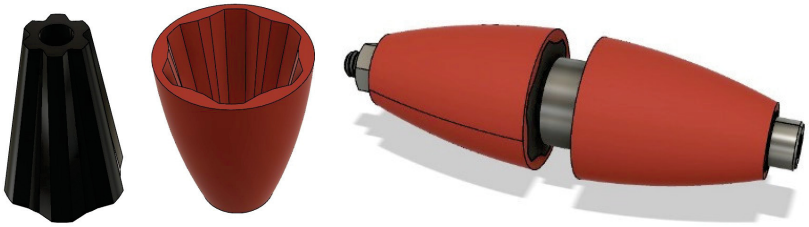
Proces modelowania rozpoczęto od wyznaczenia profilu rolki dla koła o średnicy 97 mm z rolkami o średnicy 21 mm. Do arkusza kalkulacyjnego wprowadzono

parametryczny układ równań i obliczono współrzędne punktów profilu zewnętrznego rolki [8].



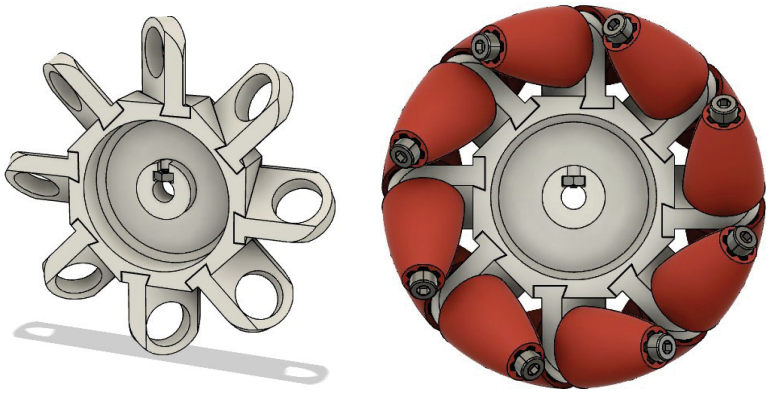

*Rysunek 3. Wykres profilu rolki*

Otrzymany zestaw współrzędnych zaimportowano do programu Fusion 360 jako krzywą i na jej podstawie wykonano model rolki. W celu uproszczenia procesu wydruku 3D, zastosowano rolki dwuczęściowe montowane na łożysku.
Połowa rolki składa się z rdzenia wykonanego z PLA oraz powłoki zewnętrznej z TPU. Dwuczęściowa konstrukcja upraszcza montaż rolki do łożyska oraz uzyskanie lepszej przyczepności rolki do podłoża.



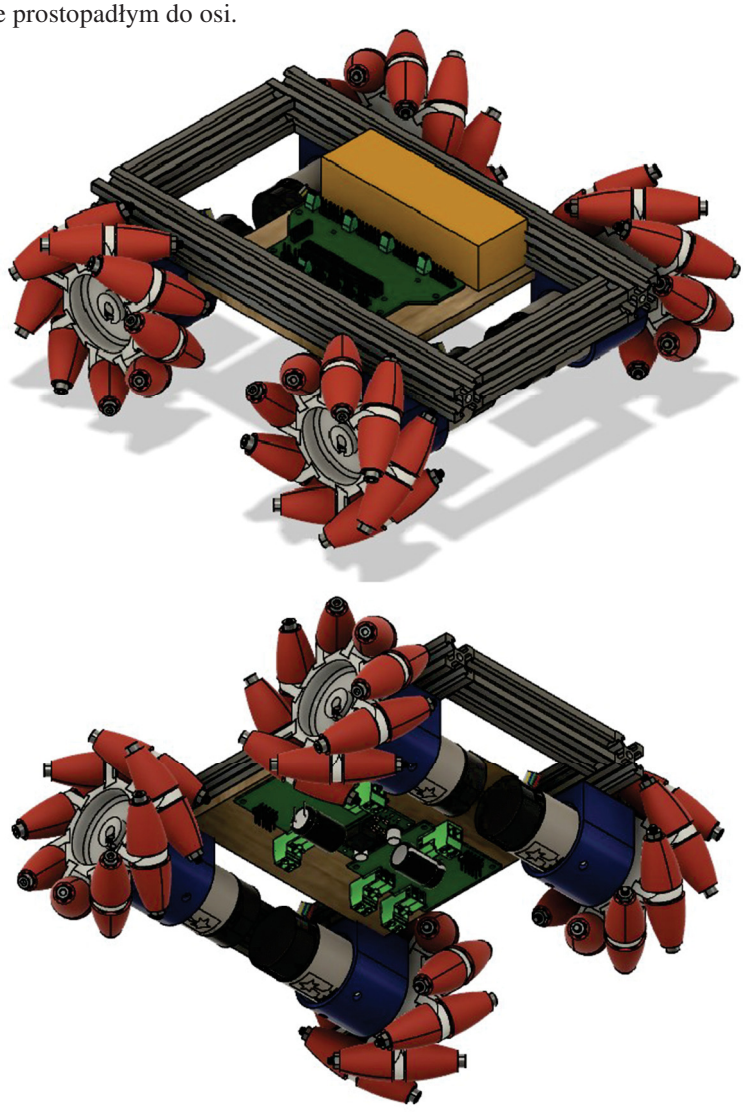
*Rysunek 4. Elementy składowe oraz złożony zespół rolki*

Śruba i nakrętka mieszczą się w profilu rolki, przez co nie mają kontaktu z podłożem w trakcie obrotu koła.
Następnie zaprojektowano piastę koła. Pozwala ona na montaż ośmiu rolek pod kątem 45° oraz montaż koła do silnika z wałem ściętym o średnicy 6 mm.



*Rysunek 5. Model piasty koła oraz zespół piasty i rolek*

Pierwszy prototyp piasty wykonano jako jednolity element, jednak otwory wydrukowane pod kątem 45° do powierzchni stolika drukarki 3D nie zachowały wymiaru oraz nie posiadały odpowiedniej wytrzymałości na rozciąganie. Próby wprasowania łożyska powodowały rozwarstwienie wydruku.
Model podzielono na części, co umożliwiło wydruk ramion piasty płasko na stoliku drukarki 3D, zwiększając dokładność wymiaru otworu oraz znacznie polepszając wytrzymałość na rozciąganie. Ramiona z wprasowanymi łożyskami są wklejane do części centralnej piasty.
Dzięki wyznaczonemu profilowi rolki, krawędzie zewnętrzne koła wyznaczają okrąg w rzucie prostopadłym do osi.

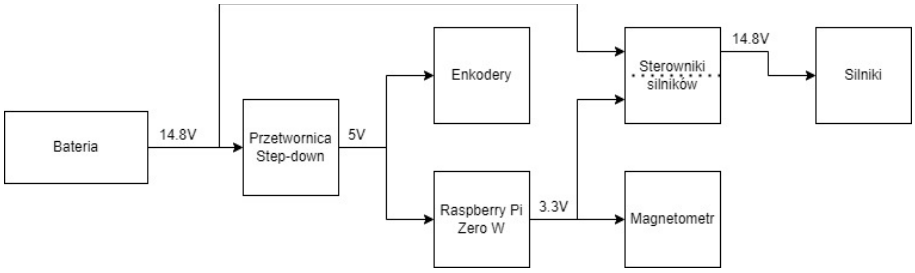


*Rysunek 6. Model 3D platformy robota*

Rama robota składa się z profili V-Slot 2020. Pozwalają one na łatwy montaż części robota bez konieczności wykonywania otworów montażowych. Profile są połączone łącznikami kątowymi do nich dedykowanymi. Wymiary zewnętrzne ramy wynoszą 250x140 mm.
Do montażu silników zaprojektowano uchwyty, które wykonano w technologii druku 3D. W procesie projektowania wykorzystano model 3D silnika udostępniony przez producenta. Pozwoliło to na bezpośrednie rzutowanie wymiarów i pozycji poszczególnych otworów.
Kształt uchwytu i konieczność dokładności wymiarów otworów montażowych silnika określiły kierunek wydruku elementu. Warstwy wydruku znajdują się w poprzek osi silnika, co znacznie osłabia element na łączeniu powierzchni montażowej silnika. Małe pole przekroju tego łączenia w pierwszym prototypie spowodowało złamanie elementu w trakcie montażu.
Szczególną uwagę zwrócono na długość śrub mocujących silnik do uchwytu. W dokumentacji technicznej producent określił maksymalną głębokość na jaką można wkręcić śruby w silnik. Wkręcenie dłuższej śruby mogłoby spowodować uszkodzenie przekładni silnika.
Do ramy robota przymocowana jest platforma ze sklejki, do której zamocowano układ elektroniczny wraz z baterią.

### 2.2. Sterowanie i zasilanie

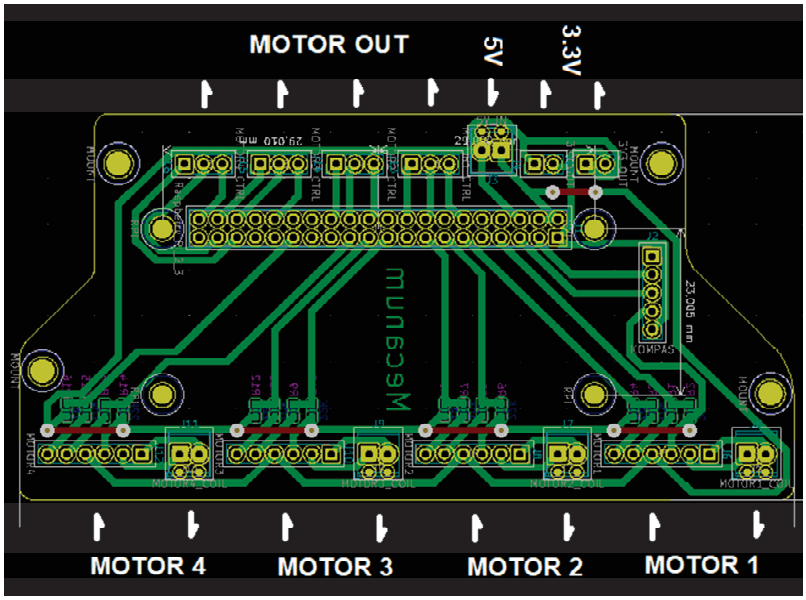
Do sterowania robotem zastosowano mikrokomputer Raspberry Pi Zero W. Wbudowany interfejs Wi-Fi i Bluetooth pozwalają na połączenie peryferii bezprzewodowych oraz łatwą komunikację z robotem. System Linux dostarcza sterowniki i obsługę protokołów, dzięki czemu połączenie na przykład kontrolera bezprzewodowego i jego obsługa nie wymaga implementacji warstwy sprzętowej. Ilość wbudowanych wejść/wyjść pozwoliła na obsługę wszystkich sygnałów.




*Rysunek 7. Schemat zasilania*

Układ zasilania robota składa się z baterii o napięciu nominalnym 14.8 V i przetwornicy step-down o napięciu wyjściowym 5 V. Silniki zasilane są przez sterowniki silników, połączone bezpośrednio do baterii. Obwód 5 V zasila mikrokomputer i enkodery znajdujące się na silnikach. Mikrokomputer posiada własny regulator 3.3 V, z którego zasilana jest część logiczna sterowników silników oraz magnetometr.
W celu ułatwienia połączenia mikrokomputera z pozostałymi komponentami robota, zaprojektowano płytkę PCB. Pozwala ona na połączenie silników za pomocą

fabrycznych konektorów. Na płytce znajdują się gniazda na mikrokomputer oraz magnetometr.




*Rysunek 8. Zaprojektowana i wykonana płytka PCB z opisem listw kołkowych i zaciskowych*

Według dokumentacji enkoderów, zakres ich napięcia zasilania wynosi od 3,5 V do 20 V. Wejścia mikrokomputera działają w zakresie 0-3.3 V. W obawie przed błędami w funkcjonowaniu enkoderów przy napięciu niższym niż zalecanym przez producenta, zasilono je napięciem 5 V. Wejścia mikrokomputera zabezpieczono przed wyższym napięciem stosując dzielnik napięcia dla każdego wyjścia enkodera.
Silniki są sterowane za pomocą mostków H. Każdy silnik zasilany jest regulowanym napięciem, z możliwością zmiany polaryzacji. Dzięki temu mikrokomputer ma kontrolę nad prędkością i kierunkiem obrotów każdego silnika. Do sterowania silnikiem wymagane są trzy sygnały.
Silniki wyposażone są w enkodery dwukanałowe CPR 64, zapewniające rozdzielczość 64 impulsów na obrót wału silnika. Obroty silnika redukowane są przez przekładnię 30:1, dzięki temu uzyskiwana jest rozdzielczość 1920 impulsów na obrót wału wyjściowego.

### 2.3. Programowanie ruchów robota

Do zaprogramowania robota wykorzystano język programowania Python. Posiada on rozbudowany pakiet bibliotek standardowych, jak i łatwo dostępnych i prostych w użyciu bibliotek użytkowników. Ponieważ jest to język interpretowany, pozwala to na łatwe testowanie i wprowadzanie poprawek do kodu, bez konieczności kompilacji.

Przykładowe, autorskie kody programów zastosowanych w robocie przedstawiono poniżej.

```
QMC5883L.py

#obsługa układu QMC5883L (magnetometr)
import smbus2

I2C_ADDRESS = 0x0D REG_ID = 0x0D
REG_CONTROL_REGISTER = 0x09 REG_STATUS_REGISTER = 0x06
REG_SR = 0x0B

MODE_STANDBY     = 0b00000000 MODE_CONTINUOUS = 0b00000001

ODR_10HZ = 0b00000000 ODR_50HZ = 0b00000100 ODR_100HZ =
0b00001000 ODR_200HZ = 0b00001100

RNG_2G = 0b00000000 RNG_8G = 0b00010000

OSR_512 = 0b00000000 OSR_256 = 0b01000000 OSR_128 =
0b10000000 OSR_64 = 0b11000000

class QMC5883L:
def init (self):
self.i2c = smbus2.SMBus(1)
if(self.i2c.read_byte_data(I2C_ADDRESS, REG_ID) != 255):
raise Exception("Device is not QMC5883L")
self.i2c.write_byte_data(I2C_ADDRESS, REG_SR, 0x01)

def setControlRegister(self, cr):
self.i2c.write_byte_data(I2C_ADDRESS, REG_CONTROL_REGISTER,
cr)

def dataReady(self):
return self.i2c.read_byte_data(I2C_ADDRESS,
REG_STATUS_REGISTER) & 0x01

def read(self):
data = self.i2c.read_i2c_block_data(I2C_ADDRESS, 0x00, 6)
return     int.from_bytes([data[1],  data[0]],
    'big',       signed='True'), int.from_bytes([data[3],
    data[2]],    'big', signed='True'),
int.from_bytes([data[5], data[4]], 'big', signed='True')
```

Program realizuje komunikacje z magnetometrem QMC5883L. Protokół I2C zaimplementowano za pomocą biblioteki smbus2. Poszczególne komendy używane w komunikacji opracowano na podstawie dokumentacji technicznej magnetometru.

```
compass.py

import QMC5883L as qmc import numpy as np import math

class Compass:
def init (self): self.mgm = qmc.QMC5883L()
self.mgm.setControlRegister(qmc.OSR_512|qmc.RNG_2G|qmc.ODR_2
00HZ|
qmc.MODE_CONTINUOUS)
```

```
self.A = np.matrix('1 0 0;0 1 0;0 0 1') self.b =
np.matrix('0;0;0')
self.corr = np.array('')

def readRawMilliGauss(self):
raw = np.array(self.mgm.read())
return raw*2000/32767

def readVector(self):
return np.asarray(np.transpose( self.A.dot(np.transpose(
[self.readRawMilliGauss()])-self.b))).reshape(-1)

def importCalibration(self,filename):
with open(filename,'r') as f: self.A =
np.matrix(f.readline()) self.b = np.matrix(f.readline())

def importCorrection(self,filename):
with open(filename,'r') as f:
self.corr = np.asarray(np.matrix(f.readline()))

def readRawAngle(self): mes = self.readVector()
heading = math.atan2(mes[1], mes[0])
if(heading < 0): heading += 2*math.pi
if(heading > 2*math.pi): heading -= 2*math.pi
return math.degrees(heading)

def correctAngle(self, ang): id = 0
for i, pt in enumerate(self.corr.transpose()[1]):
if pt > ang: id = i-1 break
a=(self.corr[id][0]-self.corr[id+1][0])/(self.corr[id][1]-
self.corr[id+1] [1])
b=(self.corr[id][0]+self.corr[id+1][0]-
a*(self.corr[id][1]+self.corr[id+1] [1]))/2
return a*ang+b

def readAngle(self):
return self. correctAngle(self.readRawAngle())
```

Program odpowiedzialny za kalibrację, odczyt i transformacje danych z magnetometru.

```
calibration_gen.py

import numpy as np
import compass
from datetime import datetime
import time

c = compass.Compass() avg_norm = 0

stamp = datetime.now().strftime('%Y_%m_%d_%H%M')

with open('cal/cal_'+stamp+'.txt', 'w') as file:
for n in range(1,20001):
mes = c.readRawMilliGauss() x,y,z = mes
```

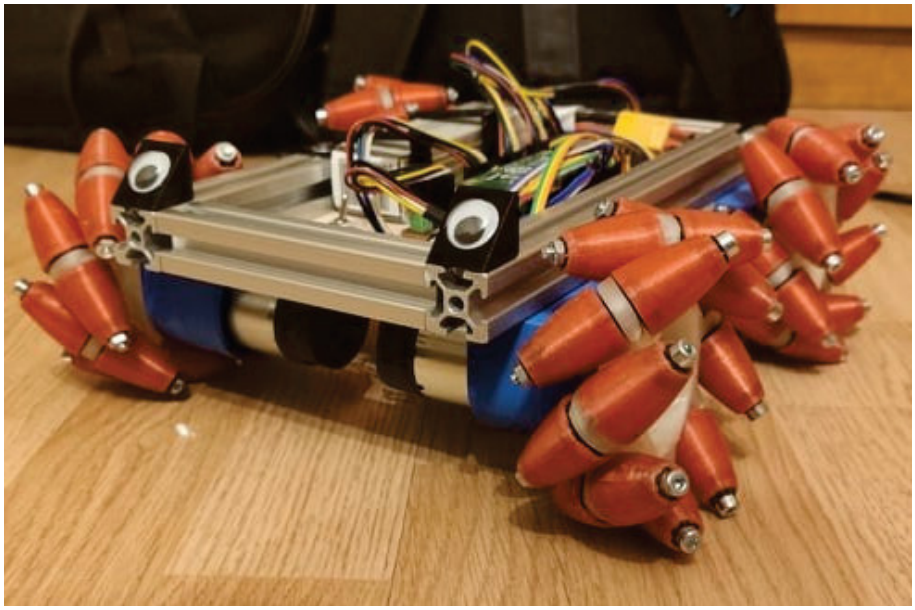
```
print(f'{x}\t{y}\t{z}', file=file)
avg_norm = np.linalg.norm(mes)/n + avg_norm*(1-1/n)
print(f'{n}: {mes}      {avg_norm}') time.sleep(.005)

with open('cal/norm_'+stamp+'.txt', 'w') as file:
print("%.8f" % avg_norm, file=file)
```

### 2.4. Montaż całego zespołu

Wydrukowano dwa komplety piast, lewe i prawe. Do ramion piast wprasowano łożyska, następnie ramiona przyklejono do części centralnej za pomocą kleju cyjanoakrylowego. Rolki zamontowano na piastach za pomocą śrub i nakrętek M4. W piastach umieszczono również nakrętki i śruby dociskowe M3 służące do zamocowania koła na wale silnika.

*Rysunek 9. Złożone koła*

Profile aluminiowe docięto do długości 250 mm i 100 mm. Ramę skręcono za pomocą dedykowanych kątowników. Wydrukowano cztery uchwyty silników.
Montaż układu zasilającego rozpoczęto od przylutowania przewodów linek do konektora baterii oraz przełącznika. Następnie wykonano połączenia, używając szybkozłączek oraz ferulek tam, gdzie przewód był umieszczany w listwie zaciskowej śrubowej. Szybkozłączki przyklejono do podstawy za pomocą kleju na gorąco.
Połączenia do uzwojeń silników, wyprowadzone na listwach zaciskowych śrubowych na płytce sterującej, połączono do sterowników silników. Połączono również zasilanie części logicznej sterowników silników.
Montaż przewodów zakończono łącząc sygnały sterujące silnikami z płytki sterujących do odpowiednich wejść sterowników silników.
Na wałkach silników osadzono koła Mecanum, umieszczając pary kół na przekątnych robota.



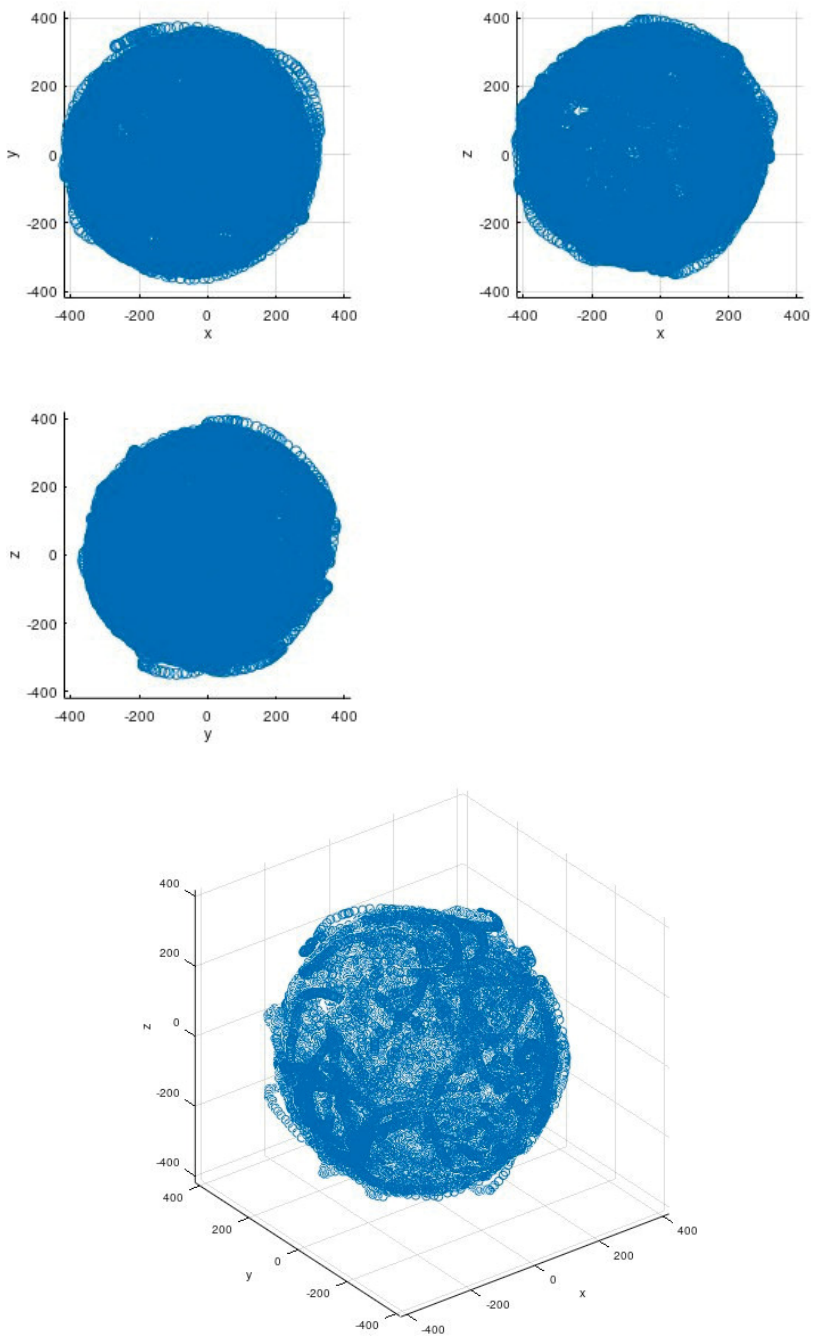
*Rysunek 10. Złożony robot omnikierunkowy*

## 3. Badania i testy

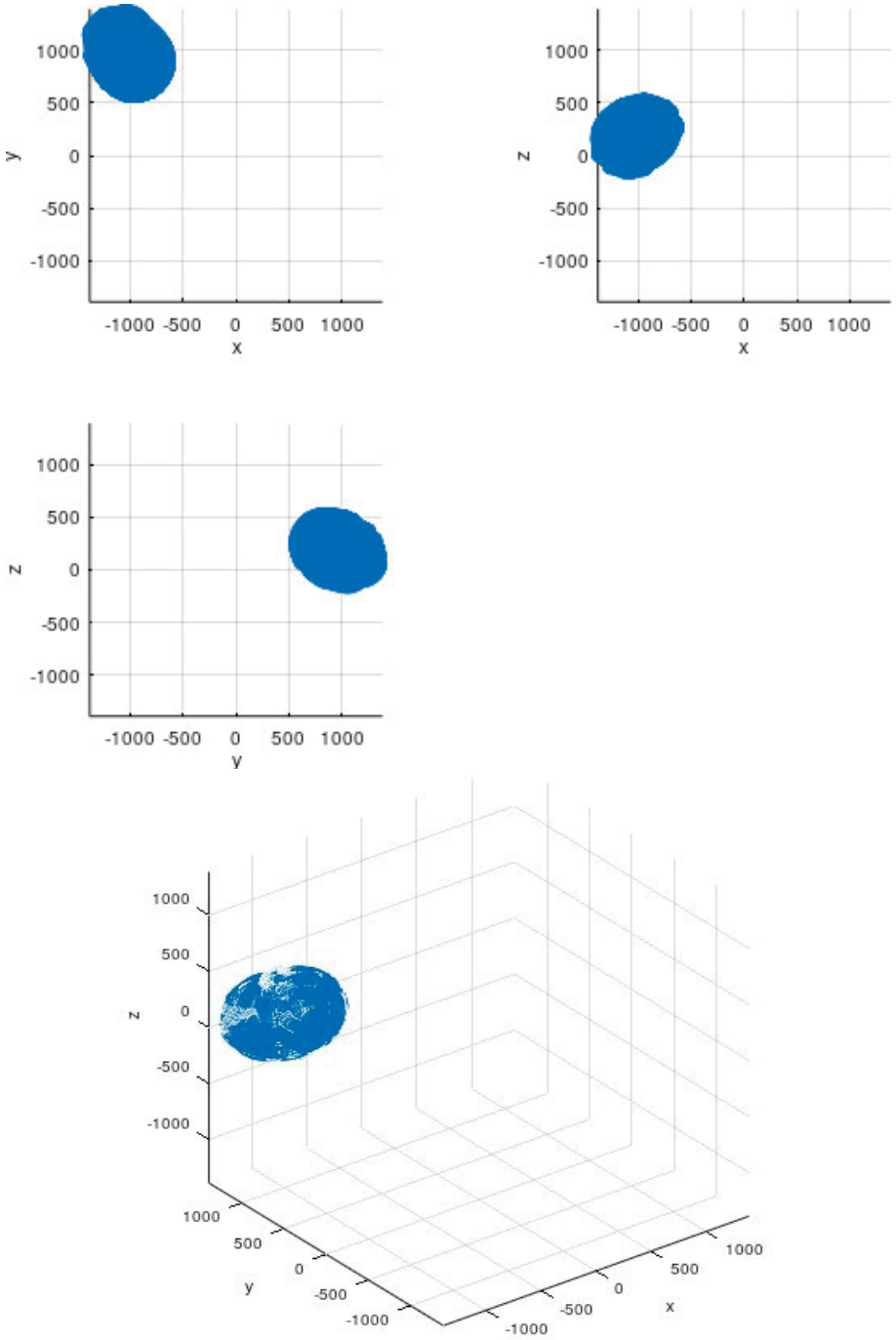
### 3.1. Test magnetometru

Magnetometr na pokładzie robota ma służyć jako kompas. Pozwoli to na odczyt pozycji kątowej robota w przestrzeni, oraz zdefiniowanie globalnego układu współrzędnych. Układ QMC5883L zawiera trzy ortogonalne czujniki pola magnetycznego. W celu kalibracji czujnika napisano program zbierający 10000 odczytów do pliku tekstowego. W trakcie działania programu magnetometr był obracany w przestrzeni.

Podstawowym założeniem przy kalibracji magnetometru jest stałość ziemskiego pola magnetycznego w danym położeniu na Ziemi. Oznacza to, że wykres wektorów odczytanych z magnetometru powinien wykreślać sferę z środkiem w początku układu współrzędnych. Nieskalibrowany czujnik może posiadać przesunięcia w swoich osiach, oraz osie mogą być przeskalowane względem siebie. Może to być spowodowane niedoskonałością czujnika, lub obecnością materiałów ferromagnetycznych w pobliżu czujnika.




*Rysunek 11. Zestawienie danych kalibracyjnych*




*Rysunek 12. Zestawienie danych kalibracyjnych, czujnik zamontowany stalową śrubą do podstawy*

Po zamontowaniu czujnika do podstawy stalową śrubą, surowe odczyty z czujnika wykreślają elipsoidę przesuniętą we wszystkich osiach od środka układu współrzędnych (rysunek 11).
Aby dokonać kalibracji czujnika należy na podstawie zebranych wektorów wyznaczyć wzór wykreślonej elipsoidy. W tym celu wykorzystano darmowy program do kalibracji magnetometrów i akcelerometrów Magneto [12]. Oprogramowanie używa algorytmu dopasowującego elipsoidę do danych pomiarowych. Po wykonaniu obliczeń program zwraca macierz i wektor, których używa się do korekty surowych odczytów z magnetometru według wzoru:

$$h_{cal} = A^{-1} \cdot (h - b) \tag{1}$$

gdzie:
$A^{-1}$ – macierz korekcyjna,
$h$ - wektor odczytany z magnetometru,
$b$ - wektor korekcyjny.

Oprogramowanie poza danymi pomiarowymi wymaga od użytkownika podania modułu wektora pola magnetycznego w jednostkach pomiarowych czujnika. Odczytanie tej wartości bezpośrednio z magnetometru nie jest możliwe przed kalibracją. Korzystając z danych kalibracyjnych obliczono średnią wartość modułu wszystkich zmierzonych wektorów.
Obliczono przybliżony środek elipsoidy:

$$m_x = \frac{\max(h_{ix}) - \min(h_{ix})}{2} + \min(h_{ix}) \tag{2}$$

$$m_y = \frac{\max(h_{iy}) - \min(h_{iy})}{2} + \min(h_{iy}) \tag{3}$$

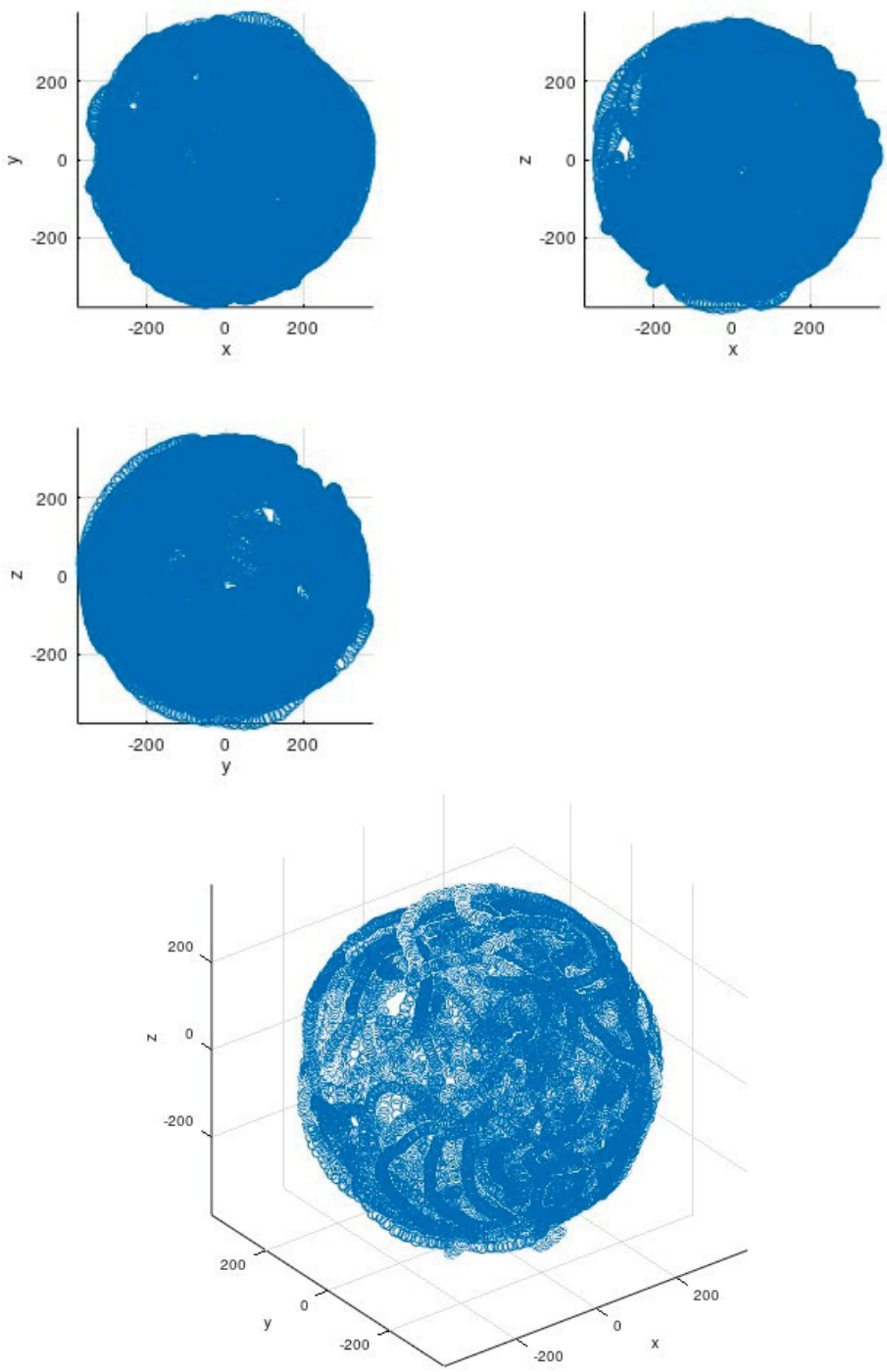
$$m_z = \frac{\max(h_{iz}) - \min(h_{iz})}{2} + \min(h_{iz}) \tag{4}$$

$$m = \begin{bmatrix} m_x & m_y & m_z \end{bmatrix} \tag{5}$$

Wyznaczono średni moduł $h_n$:

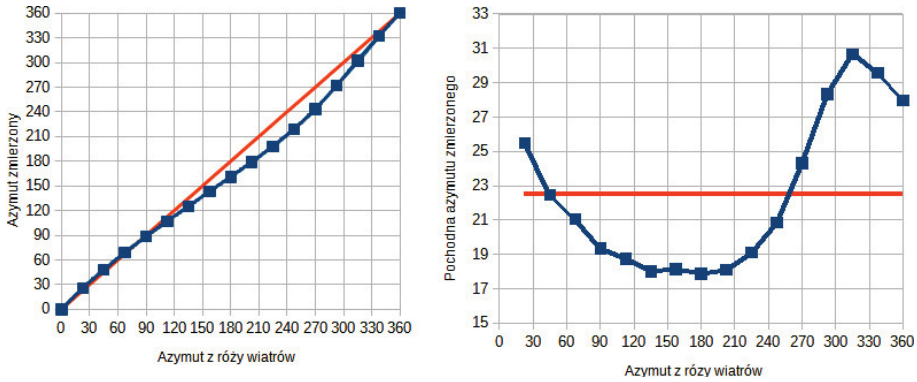
$$h_n = \frac{\sum_{i=1}^{10000} \left\| h_i - m \right\|}{10000} \tag{6}$$

Azymut magnetyczny względem dodatniego kierunku osi X magnetometru wyznaczono za pomocą dwuargumentowej funkcji arcus tangens. Argumentami tej funkcji są współrzędne Y i X wektora, a zwraca ona kąt pomiędzy dodatnim kierunkiem osi X a wektorem tworzonym przez argumenty.
W celu weryfikacji działania kompasu, na kartce papieru wydrukowano różę wiatrów dzielącą okrąg na szesnaście części.



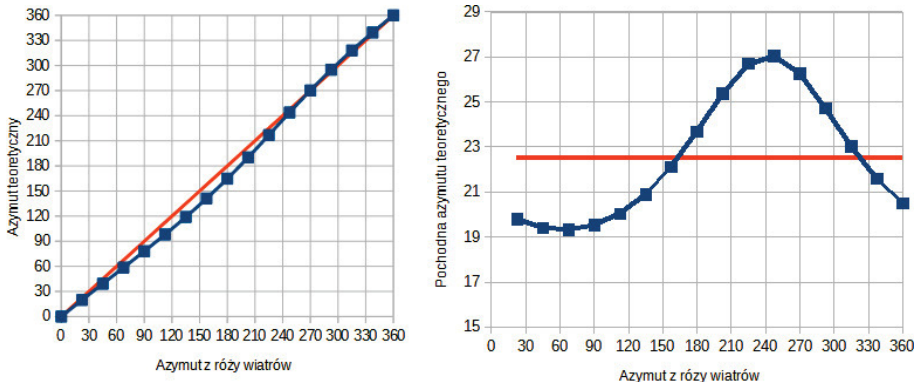

*Rysunek 13. Zestawienie danych po kalibracji, czujnik zamontowany stalową śrubą do podstawy*

Na podstawie odczytów z magnetometru ustalono azymut 0°, różę wiatrów zorientowano z podstawą testową i przyklejono do podłoża za pomocą taśmy klejącej. Następnie obracano podstawą zgodnie z kierunkiem wskazówek zegara, orientując ją zgodnie z kolejnymi liniami na róży wiatrów, odczytując wskazania kompasu.




*Rysunek 14. Wykres azymutów odczytanych w trakcie testu*

Relacja azymutu zmierzonego i wzorcowego okazała się nieliniowa. Możliwym źródłem błędu było odchylenie magnetometru od pionu. Za pomocą oprogramowania Octave wyznaczono teoretyczny odczyt azymutu przy odchyleniu magnetometru o 3° w osiach X i Y.




*Rysunek 15. Wykres azymutów teoretycznych*

Po kilkukrotnym powtórzeniu testu w różnych miejscach, błąd był większy niż wynikałoby z samego wychylenia magnetometru. Jeden z pomiarów wskazał azymut 0° w innej orientacji niż pozostałe próby. Źródłem błędu były najprawdopodobniej zbrojenia znajdujące się w betonowej podłodze.

### 3.2. Testy sterowania

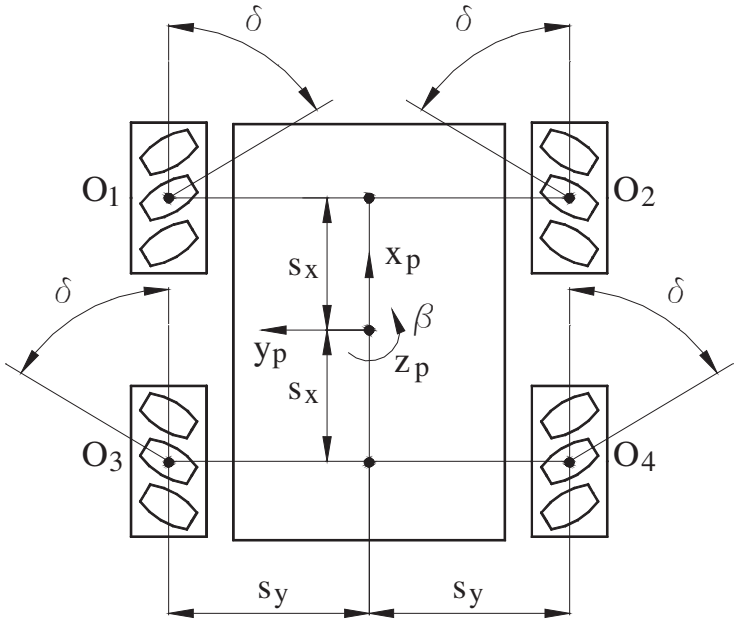
Do wysterowania sygnałów na Raspberry Pi użyto oprogramowania pi-blaster [7]. Umożliwia ono programowe generowanie sygnałów PWM bez dużego obciążenia CPU, poprzez wykorzystanie Direct Memory Access (DMA). Dzięki temu każde

z wyjść cyfrowych mikrokomputera można wykorzystać jako wyjście analogowe PWM. Umożliwiło to uproszczenie obwodów na płytce sterującej i mogła być ona wykonana z jednostronnego laminatu do płytek drukowanych. Pi-blaster tworzy bufor FIFO w folderze `/dev/pi-blaster`. Wysyłanie komend do oprogramowania jest wykonywane poprzez zapis do wspomnianego buforu, analogicznie do zapisu danych w pliku tekstowym. Dzięki temu pi-blaster jest kompatybilny z każdym językiem programowania, a komendy można wysyłać nawet z wiersza poleceń [7]. Program skonfigurowano tak, żeby obsługiwał wszystkie wyjścia obsługujące sterowniki silników.

Następnie w języku programowania Python napisano następujące programy:

- piBlaster.py – obsługa komend pi-blaster,
- Motor.py – obsługa sterowników silników,
- motors.py – zawierający konfigurację GPIO połączeń ze sterownikami silników.

Do kontroli robota wykorzystano kontroler PS4, ze względu na możliwość połączenia za pomocą Bluetooth. Do obsługi kontrolera w Pythonie posłużyła biblioteka Gamepad, która posiada gotowe mapowania poszczególnych osi i przycisków [12].




*Rysunek 16. Geometria robota omnikierunkowego w ruchomym układzie współrzędnych [6]*

Kinematykę ruchu robota omnikierunkowego w ruchomym układzie współrzędnych opisują następujące zależności [6]:

$$v_x - v_y - \beta(s_x + s_y) = \dot{\varphi}_1(R + r) \tag{7}$$

$$v_x + v_y + \beta(s_x + s_y) = \dot{\varphi}_2(R + r) \tag{8}$$

$$v_x + v_y - \beta(s_x + s_y) = \dot{\varphi}_3(R + r) \tag{9}$$

$$v_x - v_y + \beta(s_x + s_y) = \dot{\varphi}_4(R + r) \tag{10}$$

Ponieważ sterowanie nie opiera się na rzeczywistych jednostkach, we wzorach pominięto wyrazy odpowiadające za wymiary kół i robota. Rozdzielono również ruch od obrotu wokół własnej osi.

Program sterujący odczytuje z kontrolera stan przycisków kierunkowych oraz przycisków L1 i R1 odpowiedzialnych za obrót robota. Odczytane wartości przekształcane są w wektor kierunku lub wartość obrotu i na ich podstawie wysterowane są silniki.
Dla ruchu w osi X zastosowano eksperymentalnie dobrany współczynnik korekcji dla kół znajdujących się na jednej z przekątnych robota. Było to konieczne, ponieważ sterowniki silników sterują prądem dostarczonym do silnika, a co za tym idzie momentem silnika. Podczas ruchów w osi X silniki na jednej przekątnej są bardziej obciążone od drugiej przekątnej, w której koła toczą się na rolkach.
W celu zniwelowania problemu sterowania momentem podjęto próbę implementacji enkoderów do kodu programu. Niestety nawet z wykorzystaniem przerwań programowych, częstotliwość impulsów była za duża dla mikrokomputera.

## 4. Podsumowanie i wnioski

Celem pracy było zaprojektowanie robota z napędem omnikierunkowym. Przedstawiono wybór użytych materiałów i technologii, opisano implementację oprogramowania i zaprezentowano konstrukcję.

Zewnętrzna część rolek wykonana z TPU okazała się mieć małą przyczepność do podłoża. Wpływa to negatywnie na zdolność poruszania się robota.
Koła Mecanum lepiej sprawują się przy niskich prędkościach obrotowych silników. Dobrane silniki mają za małe przełożenie, przez co przy niskich prędkościach blokują się. Robot porusza się za szybko, co w połączeniu z małą przyczepnością powoduje, że łatwo wpada w poślizg. Przy hamowaniu siła bezwładności jest na tyle duża, że robot obraca się.

Przy sterowaniu napędem omnikierunkowym ważnymi parametrami silników są moment obrotowy przy niskich prędkościach obrotowych i dokładna kontrola nad prędkością obrotową.

Magnetometr używany jako kompas jest bardzo podatny na zakłócenia. Zmiana środowiska w jakim znajduje się magnetometr wymusza wykonanie kalibracji. Zbrojenia w betonowej podłodze powodowały na tyle duże zakłócenia, że uniemożliwiły zastosowanie kompasu do określenia pozycji kątowej robota.

## LITERATURA


1. Serwis internetowy: *http://www.robotplatform.com/knowledge/Classification_of_Robots/Holonomic_and_Non- Holonomic_drive.html,* 05.01.2022.
2. KYUNG-SEOK B., JAE-BOK S.: Design and Construction of Continuous Alternate Wheels for an Omnidirectional Mobile Robot, Department of Mechanical Engineering Korea University, 2003.
3. Serwis internetowy: *https://www.kuka.com/en-in/products/mobility/mobile-platforms/kmp-1500,* 13.01.2022.
4. Serwis internetowy: *https://www.kuka.com/en-in/products/mobility/mobile-robots/kmr-quantec,* 13.01.2022.
5. Serwis internetowy: *https://www.chiefdelphi.com/uploads/default/original/3X/1/e/1e642108d1c88fdf8c220e3e9c67ffac1 04c99ae.pdf,* 26.03.2021.
6. HENDZEL Z., RYKAŁA Ł.: Opis kinematyki mobilnego robota kołowego z kołami typu Mecanum, Modelowanie Inżynierskie 26(2015)57, 5-12.
7. Serwis internetowy: *https://github.com/sarfata/pi-blaster,* 25.01.2022.
8. Serwis internetowy: *https://www.omniawheel.com/conveyor-systems,* 28.01.2022.
9. GIERGIEL J. M., HENDZEL Z., ZYLSKI W.: Modelowanie i sterowanie mobilnych robotów kołowych, PWN, Warszawa 2002.
10. HENDZEL Z., GIERLAK P.: Sterowanie robotów kołowych i manipulacyjnych. Oficyna Wydawnicza Politechniki Rzeszowskiej, Rzeszów 2011.
11. HENDZEL Z.: Sterowanie ruchem nadążnym mobilnych robotów kołowych. Rzeszów: Ofic. Wyd. Pol. Rzesz., 1996.
12. Serwis internetowy: *https://sailboatinstruments.blogspot.com/2011/09/improved-magnetometer-calibration-part.html,* 25.01.2022.