

Politechnika Warszawska

W Y D Z I A Ł E L E K T R Y C Z N Y



Instytut Sterowania i Elektroniki Przemysłowej

Praca dyplomowa inżynierska

na kierunku Automatyka i Robotyka

Opracowanie konstrukcji mechanicznej i systemu sterowania dla pięcioosiowego manipulatora z wykorzystaniem procesora NI myRIO.

Michał Bogacz

Numer albumu 279086

promotor

dr inż. Szymon Piasecki

Warszawa 2019

Opracowanie konstrukcji mechanicznej i systemu sterowania dla pięcioosiowego manipulatora z wykorzystaniem procesora NI myRIO.

Streszczenie

Celem niniejszej pracy inżynierskiej było wykonanie stanowiska edukacyjnego składającego się z platformy mobilnej i zainstalowanego na niej manipulatora o pięciu stopniach swobody wraz z dedykowanym układem sterującym. Realizacja pracy obejmowała zapoznanie się z dostępnymi na rynku technologiami oraz gotowymi rozwiązaniami manipulatorów do celów edukacyjnych, a następnie, na podstawie przeprowadzonego badania rynku, zaproponowanie własnego rozwiązania. Opracowanie konstrukcji mechanicznej i układu sterowania manipulatora wymagało zapoznania się z różnymi typami napędów oraz systemami sterowania, pozwalającymi na określenie położenia poszczególnych ogniw manipulatora. Dzięki temu możliwa była implementacja równań kinematyki odwrotnej i wyznaczenie położenia punktu TCP względem zewnętrznego układu współrzędnych. Po opracowaniu konstrukcji manipulatora, obliczone zostały jego parametry kinematyczne oraz zastosowany został sterownik mikroprocesorowy pozwalający na bezprzewodową komunikację z komputerem operatora robota. Kolejnym krokiem prac było opracowanie sposobu zewnętrznego (nadziednego) sterowania platformą i manipulatorem. Na podstawie analizy dostępnych na rynku rozwiązań zaproponowano własne rozwiązanie, umożliwiające przystosowanie kontrolera do potrzeb użytkownika. W ramach realizacji pracy skonstruowano interaktywną rękawicę wyposażoną w czujniki akcelerometryczne, która pozwala na sterowanie platformą mobilną. Ostatnią z wykonanych w trakcie trwania projektu czynności, było połączenie dwóch równoległych projektów: konstrukcji manipulatora i dedykowanej rękawicy sterującej oraz budowy platformy mobilnej, w jednego robota mobilnego. Integracja dwóch rozwiązań zawierała połączenie elementów mechanicznych (osadzenie manipulatora na platformie), elektrycznych (zasilanie i poprowadzone przewodami interfejsy komunikacyjne), programistycznych (połączenie programu wraz z interfejsem graficznym w taki sposób, aby umożliwiał sterowanie platformą i manipulatorem równolegle). Na końcu pracy przedstawione zostały instrukcje umożliwiające sterowanie robotem, realizowane za pomocą opracowanej rękawicy oraz wskazane kierunki dalszego rozwoju projektu.

Słowa kluczowe:

- robot mobilny
- manipulator
- myRio
- Arduino
- LabVIEW

Development of mechanical design and control system for a five-axis manipulator using the NI myRIO processor.

Abstract

The aim of this bachelor's project was to create an educational station consisting of a mobile platform and a manipulator with five degrees of freedom installed on it, along with a dedicated control system. The implementation of the work included familiarization with the technologies available on the market and ready solutions for manipulators for educational purposes, and then, based on the market research carried out, proposing unique solution. The development of the mechanical construction and the manipulator control system required research on various types of drivers and control systems, allowing to determine the position of individual links of the manipulator. As a result, it was possible to implement inverse kinematics equations and to determine the position of the TCP point with respect to the external coordinate system. After developing the manipulator structure, its kinematic parameters were calculated and a microprocessor controller with wireless communication with the main computer was used. In the next step an external (master) control of the platform and manipulator was developed. Based on the analysis of solutions available on the market, the own solution, enabling the controller to be adapted to the user's needs, was proposed. As part of the work, an interactive glove equipped with accelerometers has been constructed, that allows manipulation of the mobile platform. The last of the activities carried out during the project was integration of two parallel projects: the construction of a manipulator and a dedicated control glove and the construction of a mobile platform, in one mobile robot. The integration of the two solutions included a combination of mechanical elements (placing the manipulator on the platform), electrical (power supply and cable communication interfaces), programming (connection of the program along with the graphical interface in such a way as to allow parallel control of the platform and manipulator). Finally, instructions to control the mobile robot with the aid of a developed glove were presented and indicated directions for further development of the project.

Keywords:

- Mobile robotics
- manipulator
- myRio
- Arduino
- LabVIEW



miejscowość i data

imię i nazwisko studenta

numer albumu

kierunek studiów

OŚWIADCZENIE

Świadomy/-a odpowiedzialności karnej za składanie fałszywych zeznań oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie, pod opieką kierującego pracą dyplomową.

Jednocześnie oświadczam, że:

– niniejsza praca dyplomowa nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.) oraz dóbr osobistych chronionych prawem cywilnym, – niniejsza praca dyplomowa nie zawiera danych i informacji, które uzyskałem/-am w sposób niedozwolony, – niniejsza praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów lub tytułów zawodowych, – wszystkie informacje umieszczone w niniejszej pracy, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami, – znam regulacje prawne Politechniki Warszawskiej w sprawie zarządzania prawami autorskimi i prawami pokrewnymi, prawami własności przemysłowej oraz zasadami komercjalizacji.

Oświadczam, że treść pracy dyplomowej w wersji drukowanej, treść pracy dyplomowej zawartej na nośniku elektronicznym (płycie kompaktowej) oraz treść pracy dyplomowej w systemie iSOD są identyczne.

czytelny podpis

Spis treści

Streszczenie	3
Abstract	5
1. Wprowadzenie	11
1.1. Wstęp	11
1.2. Cel i zakres pracy	11
1.3. Zawartość pracy	13
2. Manipulator pięcioosiowy.....	14
2.1. Wprowadzenie	14
2.2. Dostępne na rynku rozwiązania i ich porównanie	14
2.2.1. Rozwiązania komercyjne (przemysłowe)	14
2.2.2. Rozwiązania do celów edukacyjnych typu DIY (<i>ang. Do It Yourself</i>)	17
2.2.3. Podsumowanie	18
2.3. Konstrukcja mechaniczna manipulatora	18
2.3.1. Założenia projektowe manipulatora	18
2.3.2. Analiza teoretyczna	20
2.3.3. Dobór układów napędowych oraz narzędzia dla manipulatora.....	23
2.3.4. Dobór układów zasilania dla napędów	28
2.3.5. Budowa robota.....	30
2.4. Opis matematyczny sterowania manipulatorem	35
2.4.1. Opis ramienia w notacji DH	35
2.4.2. Wykorzystanie równań kinematyki prostej do obliczania pozycji robota	38
2.5. Realizacja sterowania manipulatorem	39
2.5.1. Dobór procesora sterującego	39
2.5.2. Środowisko programowania	43
2.5.3. Dobór komunikacji komputer–procesor sterujący	44
2.5.4. Opis programu sterującego manipulatorem w środowisku LabView	45
3. Sterowanie platformą i manipulatorem.....	47
3.1. Wprowadzenie	47
3.2. Kontrolera do sterowania platformą i manipulatorem.....	51
3.2.1. Założenia projektowe kontrolera	51
3.2.2. Dobór czujników tensometrycznych	51

3.2.3.	Dobór czujnika akcelerometrycznego oraz żyroskopu	52
3.3.	Oprogramowanie kontrolera.....	54
3.3.1.	Dobór procesora sterującego dla kontrolera	54
3.3.2.	Opis interfejsu komunikacyjnego z komputerem.....	58
3.3.3.	Program sterujący rękawicą Arduino	59
3.3.4.	Program sterujący rękawicą LabVlew.....	61
4.	Sterowanie robotem (platforma mobilna i manipulator)	63
4.1.	Wprowadzenie (zintegrowanie manipulator + platforma)	63
4.2.	Komunikacja wewnętrz platformy mobilnej	63
4.2.1.	Opis komunikacji między STM32, a NI myRio.....	63
4.2.2.	Opis programu komunikacji między STM32, a NI myRio wykonanej w LabVlew	65
4.3.	Interfejs graficzny LabView.....	67
4.3.1.	Założenia projektowe dla interfejsu graficznego	67
4.3.2.	Program LabView.....	68
4.4.	Zastosowanie rękawicy jako kontrolera	69
4.4.1.	Zestawy gestów używane do sterowania	69
5.	Badania i weryfikacja zaproponowanego rozwiązania.....	72
6.	Podsumowanie i wnioski	77
7.	Bibliografia.....	79
8.	Spis rysunków	82
9.	Spis Tabel	85
10.	Dodatek	86

1. Wprowadzenie

1.1. Wstęp

Roboty mobilne o przeznaczeniu przemysłowym oraz różnego rodzaju zrobotyzowane pomoce domowe nabierają coraz większego znaczenia w wykonywanych przez człowieka codziennych aktywnościach. Robotyzacja oraz automatyzacja pozwalają na odciążenie człowieka w codziennych czynnościach i wykonywanej pracy. W przeciwieństwie do czynnika ludzkiego, po zrobotyzowanym pomocniku można spodziewać się całkowitej dokładności i stałości wykonywanych zadań. Szerokie możliwości zastosowania robotyki wynikają z wysokiej elastyczności powstających konstrukcji i algorytmów sterowania, umożliwiających realizację praktycznie dowolnych aplikacji.

Jako początek powstania robotyki można przyjąć rok 1926. W roku tym korporacja *Westinghouse* zaprezentowała pierwszego robota, który był sterowany poleceniami głosowymi wydawanymi telefonicznie [1]. Od roku 1926 minęło wiele lat i dużo zmieniło się również w robotyce, którą dziś zajmują się zarówno hobbyści jak i profesjonalisi. Platformy uruchomieniowe oraz gotowe projekty DIY (*ang. Do It Yourself, „zrób to sam”*) pozwalają na wykonanie i zaprogramowanie własnych zrobotyzowanych aplikacji i pomocników praktyczne przez każdego, nawet bez specjalistycznej wiedzy w tym zakresie. Należy jednak pamiętać, że pomimo wielu ułatwień, które pojawiły się w dziedzinie robotyki, aby opracowywać i wdrażać złożone rozwiązania i nowe technologie potrzeba dobrze wykształconego i doświadczonego w projektowaniu personelu technicznego i naukowego. Aby wykształcić taki personel konieczna jest odpowiednia baza stanowiskowa, umożliwiająca prowadzenie edukacji w robotyce na odpowiednio wysokim i atrakcyjnym dla odbiorcy poziomie.

Celem niniejszej pracy jest opracowanie nowoczesnej platformy mobilnej z zainstalowanym manipulatorem pięcioosiowym, dedykowanej do celów dydaktycznych i edukacyjnych. Wiele podobnych urządzeń jest dostępnych komercyjnie, jednakże większość oferowanych rozwiązań jest bardzo droga, ponadto ma skomplikowaną budowę i sposób programowania instrukcji ruchu, często nieodpowiedni do celów edukacyjnych. Funkcjonalność rozwiązania zaproponowanego w ramach dwóch, równolegle realizowanych prac inżynierskich, tj. opracowanie platformy mobilnej z zainstalowanym na niej 5-cio osiowym manipulatorem, jest bardzo szeroka, umożliwiając wszechstronny rozwój projektu i jego różne aplikacje. Całość pracy jest oparta o projekty wykonane w latach poprzednich przez Mateusza Plutę oraz Dominika Bednarskiego [2] [3].

1.2. Cel i zakres pracy

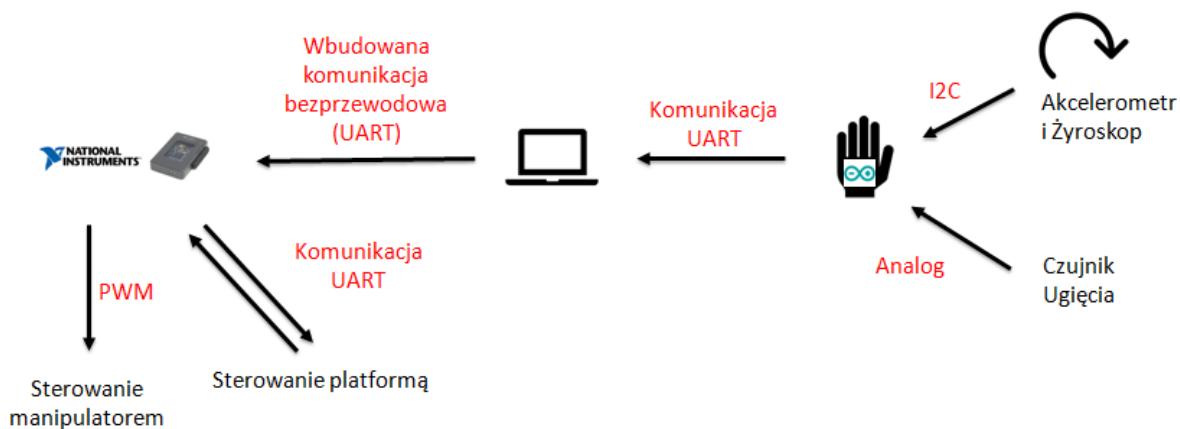
Główym celem niniejszej pracy inżynierskiej było skonstruowanie i oprogramowanie manipulatora antropomorficznego pięcioosiowego oraz budowa i oprogramowanie kontrolera sterującego dla platformy mobilnej z zainstalowanym na niej manipulatorem. Opracowana platforma mobilna wyposażona w manipulator zostanie wykorzystana w realizowanym na Wydziale Elektrycznym procesie dydaktycznym na kierunkach Automatyka i Robotyka oraz Elektromobilność. W trakcie realizacji projektu zaproponowano wiele rozwiązań, tak aby obsługa i programowanie chwytaka były możliwe dla nieprzeszkolonego użytkownika, bez wcześniejszej, szczegółowej znajomości wykorzystanego środowiska roboczego. Opracowane rozwiązanie umożliwia realizację takich zadań jak: sterowanie napędami prądu stałego, zrozumienie zasady działania i metod sterowania serwomechanizmami elektronicznymi oraz implementacja równań kinematyki odwrotnej w sterowniku mikroprocesorowym i zastosowanie ich do wyznaczania położenia narzędzia robota.

Niniejsza praca inżynierska poświęcona jest zagadnieniom dotyczącym opracowania manipulatora 5-cio osiowego (konstrukcja mechaniczna oraz sposób sterowania poszczególnymi osiami) oraz opracowaniu kontrolera wraz z oprogramowaniem (umożliwiającego poruszanie zarówno manipulatorem jak i platformą).

Realizacja części pracy dotyczącej manipulatora wymagała: opracowania jego projektu mechanicznego oraz elektrycznego, opracowania algorytmu sterowania manipulatorem, implementacji równań kinematyki ramienia (prostej oraz odwrotnej), wybrania odpowiedniego kontrolera mikroprocesorowego i interfejsów komunikacyjnych, umożliwiających bezprzewodową komunikację, zaprogramowania interfejsu użytkownika, zarówno dla manipulatora, jak i dla całej platformy. Głównym założeniem projektowym dla interfejsu oraz jednostki sterującej całym robotem była łatwość ich obsługi dla osób, które miałyby pracować z jednostką po raz pierwszy.

Realizacja części pracy dotyczącej kontrolera wymagała: opracowania projektu elektrycznego, dobrania odpowiednich przetworników, od których uzależnione będzie działanie kontrolera, wybrania odpowiedniego układu mikroprocesorowego oraz interfejsu do komunikacji z komputerem umożliwiającego sterowanie całym urządzeniem. Fizyczną realizacją kontrolera jest interaktywna rękawica.

Oprogramowanie sterujące manipulatorem oraz sterowanie pośrednie platformy zostało zrealizowane w oparciu o procesor NI myRio, programowany w języku NI LabView. Jako procesor obsługujący opracowaną rękawicę interaktywną został wybrany układ mikrokontrolera Arduino UNO. Czujniki kontrolera zostały zamocowane na rękawicy, co umożliwiło pełne korzystanie z akcelerometru oraz pięciu tensometrów. Wszystkie elementy projektu połączone są szeregowymi interfejsami komunikacyjnymi UART. Schemat komunikacji w opracowanym układzie przedstawiono na Rys. 1.1.



Rysunek 1.1. Uproszczony schemat komunikacji w zrealizowanym projekcie.

1.3. Zawartość pracy

Praca tematycznie została podzielona na trzy rozdziały przedstawiające realizację poszczególnych części wykonywanego projektu.

W pierwszej części pracy (rozdziale drugim) zawarto informacje na temat zbudowanego manipulatora pięcioosiowego, który w zamyśle ma dysponować szerokim zakresem możliwych do wykonania czynności. W rozdziale przedstawiono analizę komercyjnych rozwiązań oraz oferowanych przez nie możliwości. Na tej podstawie zostały zdefiniowane i opisane założenia projektowe oraz przedstawiono proces budowy opracowanej konstrukcji mechanicznej manipulatora. W rozdziale poświęconym manipulatorowi przedstawiono również jego opis matematyczny, umożliwiający implementację równań kinematyki manipulatora. Przedstawiono także użyte w projekcie biblioteki programowe, ułatwiające obliczenia i opis matematyczny. Na końcu tej części zawarto rozdział opisujący sposób programowania całego manipulatora oraz użytego procesora.

W kolejnej części pracy (rozdział trzeci) opisano proces projektowania i budowy kontrolera, służącego do sterowania opracowanym robotem mobilnym, składającym się z poruszającej się platformy oraz zamontowanego na niej manipulatora. W tym rozdziale przedstawiono również dostępne rozwiązania i sposoby sterowania platformą mobilną oraz dokonano porównania ich właściwości, wad i zalet. Następnie opisano konstrukcję opracowanego kontrolera (rękawicę akcelerometryczną) oraz zastosowany do jej obsługi procesor sygnałowy. W kolejnej części rozdziału pokazano szczegółowe elementy oprogramowania kontrolera oraz połączenia urządzenia z resztą systemu.

W czwartym rozdziale pracy zawarto informacje dotyczące integracji całego realizowanego projektu, w którego skład wchodzą dwie prace inżynierskie, obejmujące konstrukcję i sterowanie platformą mobilną (praca wykonana przez Huberta Skonecznego) oraz konstrukcję i sterowanie umieszczonego na platformie manipulatora oraz kontrolera, opisane w tej pracy. W kolejnych podrozdziałach opisano proces projektowania oraz programowania interfejsu graficznego służącego do obsługi całej platformy. Następnie opisane zostały interfejsy komunikacyjne użyte do połączenia komputera PC z głównym procesorem oraz z kontrolerem obsługującym sterowanie napędami platformy. Dalej przedstawiono sposób zasilania platformy oraz manipulatora, wraz z użytymi przekształtnikami energoelektronicznymi. Rozdział zakończono opisem wybranych sposobów sterowania przy pomocy opracowanego kontrolera oraz procesu doboru jak najbardziej naturalnych i wygodnych dla użytkownika sposobów sterowania robotem. Ze względu na poziom skomplikowania systemu wydawania komend i wiele możliwych do zastosowania kombinacji sterujących, w kolejnym podrozdziale skupiono się na opisaniu wszystkich możliwych sposobów sterowania i kombinacji wydawanych komend.

Rozdział piąty przedstawia opis wykonanych testów funkcjonalnych i badań na rzeczywistym układzie. Zawarto w nim wydruki ekranów oscyloskopów oraz zdjęcia gotowej konstrukcji. W rozdziale przedstawiono również potencjalne ścieżki rozwoju projektu oraz wnioski wyciągnięte podczas procesu budowy robota. Informacje te można potraktować jako podpowiedzi dla ewentualnego późniejszego rozwijania projektu.

Ostatnią częścią pracy jest wykaz literatury, zawartych tabel i obrazów oraz dodatki.

2. Manipulator pięcioosiowy

2.1. Wprowadzenie

Wraz z postępem w dziedzinie robotyki zwiększa się ilość stopni swobody (osi) manipulatorów. Manipulatory wieloosiowe powstają od bardzo dawna, są wdrażane i wykorzystywane nie tylko w przemyśle, ale również pomagają w wykonywaniu różnych zadań w gospodarstwach domowych. Manipulatory o różnych rozmiarach można spotkać w różnych dziedzinach życia takich jak:

- przemysł;
- medycyna i rehabilitacja;
- maszyny transportowe i ruchome;
- edukacja i zabawa;
- zastosowania ogólne.

Każde z wymienionych powyżej zastosowań wymusza określone właściwości konstrukcji manipulatora oraz odpowiednią strukturę oprogramowania, zarówno interfejsu użytkownika, jak i programu sterującego poszczególnymi osiami. Różne konfiguracje wymuszają dokładne przeanalizowanie i dobranie robota do potrzeb projektowych na kilku płaszczyznach, które opisano przy okazji przedstawienia założeń projektowych w rozdziale 2.3.1.

Najczęściej do celów edukacyjnych wykorzystuje się manipulatory ogólnego zastosowania. Taki typ przeznaczenia robota jest najlepszym rozwiązaniem pod względem konstrukcyjnym, naukowym oraz ekonomicznym.

Na potrzebę analizy rozwiązań komercyjnych wykonano estymowane założenia projektowe. Głównymi czynnikami konstrukcyjnymi branymi pod uwagę były: zasięg (musiał przewyjszać 800 mm, aby robot mógł dotknąć podłożą wokół całego podwozia, na którym będzie zainstalowany), udźwig (został ustalony na około 1-3 kg, aby robot mógł operować różnego rodzaju przedmiotami), liczba stopni swobody (autor projektu ustalił ją na minimum 4) oraz waga całej konstrukcji (na tyle mała, aby platforma mobilna mogła się swobodnie poruszać). Ważnym czynnikiem była również jak najniższa cena całego rozwiązania.

2.2. Dostępne na rynku rozwiązania i ich porównanie

2.2.1. Rozwiązania komercyjne (przemysłowe)

Aktualny rynek gotowych rozwiązań manipulatorów komercyjnych jest niezwykle szeroki i zawiera wiele pozycji. Światowi potentaci, tacy jak *ABB*, *KUKA*, *FANUC* czy *Kawasaki* starają się przekonać do swoich produktów tak wielu klientów, jak tylko jest to możliwe. Roboty nie są już budowane do szczególnych zastosowań, ale odpowiednie serie są sprzedawane jako gotowe moduły linii produkcyjnych. Rozwiązanie takie nie oznacza jednak całkowitej rezygnacji z konstruowania robotów do uniwersalnego i nieprzemysłowego przeznaczenia. Rozwiązania komercyjne, które można zastosować do celu opisanego w założeniach projektowych, można podzielić na dwie główne grupy: małe roboty przemysłowe oraz roboty edukacyjne, tworzone z myślą o współpracy z uczelniami wyższymi. Obie wymienione wcześniej grupy mają swoje zdecydowane zalety, ale nie brakuje im również wad, które mogą znacznie utrudnić zastosowanie takich maszyn do bardziej wymagających projektów.

Małe roboty przemysłowe są najłatwiej i najczęściej dostępne. To maszyny, które wspierają proces codziennej produkcji w wielu zakładach przemysłowych. Ich dostępność można zdecydowanie zaliczyć na ogromny plus. Robotami początkowo rozważanymi do realizacji projektu były *ABB IRB1200*, *KUKA LBR IIWA* oraz *MIRhook100* firmy Kawasaki. Poszczególne konstrukcje, wraz z ich właściwościami oraz różnicami między poszczególnymi modelami przedstawiono poniżej.



Rysunek 2.1. ABB IRB1200 [4].

Pierwszym z analizowanych rozwiązań jest robot *IRB 1200*, produkcji *ABB*, przedstawiony na Rys. 2.1. Jest zdecydowanie bardzo dobrym wyborem, jeśli chodzi o zastosowania w przemyśle do wykonywania precyzyjnych czynności, które wymagają również relatywnie sporego udźwigu. Posiadając zasięg 900 mm idealnie wpasowywałby się w potrzeby ruchomego łażika. Jego głównymi wadami jest cena, zbyt duża waga, niekorzystnie wysoka budowa oraz brak możliwości programowania poza środowiskiem *RobotStudio®*.



Rysunek 2.2. KUKA LBR IIWA [5].

Kolejnym analizowanym manipulatorem jest manipulator *KUKA LBR IIWA*, pokazany na Rys. 2.2. Jest pierwszym produkowanym seryjnie robotem, umożliwiającym człowiekowi bezpieczną pracę w ramach jednego stanowiska, nie wymagającym wygrodzonych stref bezpieczeństwa. Podobnie jak zaprezentowany wcześniej robot *ABB*, robot *KUKA LBR IIWA* ma idealny dla potrzeb projektu zasięg (około 900 mm) oraz wysoki udźwig (w zależności od wersji 7 kg lub 14 kg).

Niestety zbyt wysoka cena oraz za duża, jak na wymagania projektu masa, przekreśliły możliwość zastosowania tego robota.



Rysunek 2.3. KAWASAKI MIRHook 100 [6].

Kolejny analizowany robot, *MIRHooK* pokazany na Rys. 2.3, produkowany przez firmę Kawasaki okazał się bardzo ciekawym rozwiązaniem. Jest to gotowy robot typu jezdnego, stworzony do celów pracy autonomicznej w magazynach oraz fabrykach. Jest stosunkowo niewielki oraz posiada zintegrowany pneumatyczny chwytak o wysokiej sile udźwigu. Jego wadami są przede wszystkim zbyt wysoka, jak na potrzeby edukacyjne cena oraz niezgodność z przyjętymi założeniami konstrukcji robota mobilnego.

Innym typem analizowanych robotów komercyjnych są roboty, których przeznaczeniem jest edukacja i demonstracja możliwości technologicznych rozwiązań zastosowanych przez daną firmę. Głównymi zaletami takich rozwiązań są niewielkie rozmiary i masa. Często równocześnie z własnym, dedykowanym oprogramowaniem, producenci takich jednostek udostępniają możliwość programowania robota w dowolny sposób. Niestety dostępność takich robotów jest znikoma i bardzo trudno znaleźć egzemplarze dostępne na rynku, ponieważ ilość jednostek jest ścisłe limitowana. Równocześnie z problemem dostępności, występuje również wysoka cena, co ogólnie utrudnia zakup takiego robota.



Rysunek 2.4. KUKA YouBot 480 [7].

Przykładem takiego rozwiązania jest, pokazany na Rys. 2.4, robot to *KUKA youBot*. Jest to kolejna odsłona stworzonego do celów edukacyjnych robota Kuka. Sprzedawany jest on z gotową platformą, co znacznie ułatwiały realizację projektu pozwalając skupić się wyłącznie na metodach i sposobach programowania ruchów. Parametry konstrukcyjne robota idealnie wpisują się w założenia

projektu – przy niedużej wadze wynoszącej 20 kg, pozwala na przenoszenie nawet do 10 kg ciężaru. Wystarczający zasięg oraz wsparcie firmy Kuka dla projektów studenckich, zarysowało platformę *youBot* jako idealnego kandydata. Niestety w momencie realizacji projektu była ona już niedostępna w sprzedaży.

2.2.2. Rozwiązania do celów edukacyjnych typu DIY (*ang. Do It Yourself*)

Ze względu na wysoką cenę rozwiązań proponowanych przez dużych producentów, w ramach analizy dostępnych rozwiązań sprawdzono również rynek tanich rozwiązań hobbystycznych oraz układów typu DIY. Ceny takich robotów są znacznie niższe, a producenci tych rozwiązań pozwalają na całkowitą dowolność w wyborze jednostki sterującej. Roboty tego typu są niezwykle uniwersalne, lekkie i tanie. Niestety, właśnie z tego powodu, często nie spełniały założeń konstrukcyjnych zdefiniowanych dla projektu. Prostota konstrukcji oraz niewystarczający zasięg nie pozwalały na odpowiednie zamocowanie ramienia na platformie mobilnej. Poniżej przedstawiono kilka, najbardziej odpowiadających przyjętym założeniom projektowym robotów i na ich przykładzie pokazano ich główne wady i zalety.



Rysunek 2.5. *uArm Swift Pro* [8].

Ramię *uArm*, przedstawione na Rys. 2.5, jest najdroższą z rozważanych opcji robota (koszt około 6000 zł) do zastosowań edukacyjnych i hobbystycznych. Ramię typu desktop waży zaledwie 2,2 kg i może podnieść do 500 g. Konstrukcja oparta o serwonapędy elektryczne zapewnia wystarczającą dokładność oraz szybkość poruszania się. Jednak zasięg takiego ramienia to zaledwie 320mm, przy czterech stopniach swobody. Taki układ konstrukcyjny nie zapewnia wystarczającej funkcjonalności dla celów projektu. Ramię nie będzie miało możliwości dostania się do każdego z miejsc na podłożu dokoła platformy. Uniemożliwiłoby to zbieranie próbek wokół całego manipulatora.



Rysunek 2.6. DIY 4-Axis Servos Control Palletizing Robot Arm Model [9].

Kolejne analizowane ramię manipulatora, przedstawionym na Rys.2.6, jest zdecydowanie tańszym klonem swojego poprzednika, cena tego modelu to około 600 zł. Ma podobne wady i zalety, z wyjątkiem większego zasięgu, sięgającego 400 mm. Takie właściwości konstrukcyjne niestety nadal nie są wystarczające na potrzeby realizowanego projektu.

Dwa pokazane powyżej roboty doskonale obrazują problem, który pojawił się w trakcie analizy rozwiązań dedykowanych celom hobbystycznym. Wszystkie z tych robotów były bardzo atrakcyjne pod względem udźwigu, wag i ceny, niestety żaden nie miał wystarczającego zasięgu ramienia ani możliwości wykonania ruchu złożonego, co jest wymagane dla manipulatora znajdującego się na platformie mobilnej.

2.2.3. Podsumowanie

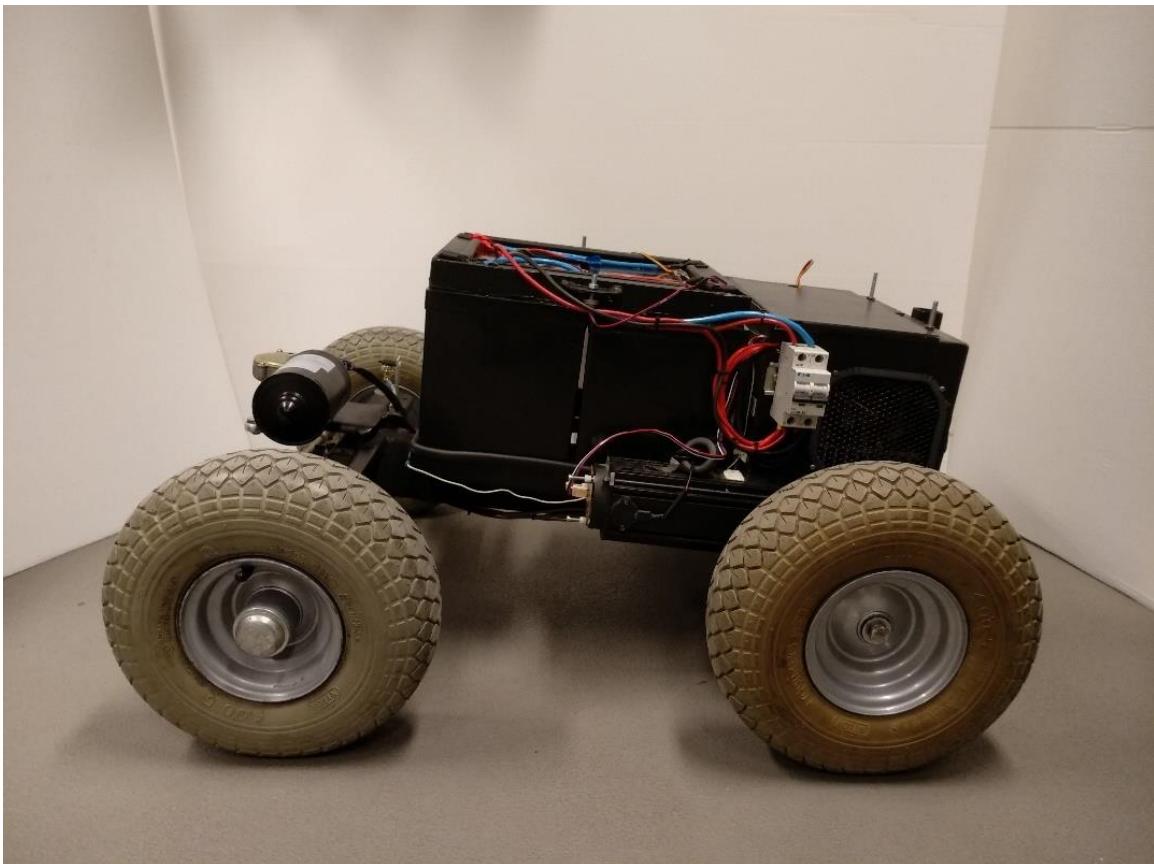
Najczęstszym problemem występującym we wszystkich produktach była ich zbyt wysoka cena oraz masa. Rozwiązania przemysłowe były po prostu zbyt drogie i ciężkie, aby znaleźć zastosowanie w realizowanym projekcie. Z kolei rozwiązania hobbystyczne nie zapewniały odpowiedniego zasięgu ramienia i możliwości wykonania złożonego ruchu.

Przeprowadzona analiza powyższych produktów pod względem założeń projektowych, wynikających z właściwości zbudowanej już platformy mobilnej wykazała, że żadna z dostępnych komercyjnie jednostek nie spełnia narzuconych wymagań. W wyniku czego zdecydowano się na stworzenie własnej konstrukcji, która będzie możliwie najlepiej dopasowana do założeń projektowych.

2.3. Konstrukcja mechaniczna manipulatora

2.3.1. Założenia projektowe manipulatora

Opracowana i wykorzystana w projekcie platforma mobilna, przedstawiona na Rys. 2.7, jest czterokołową konstrukcją, opartą o podwozie pochodzące z wózka inwalidzkiego. Ze względu na zastosowane koła oraz rodzaj i sposób montażu akumulatorów ma wysokość 460 mm, co w porównaniu z innymi tego typu platformami jest dość duże i mało praktyczne.



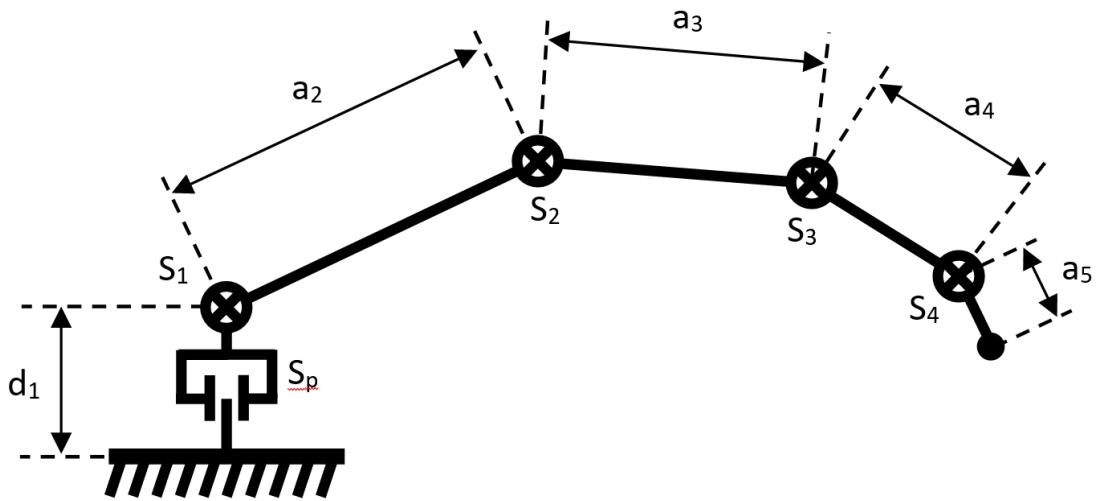
Rysunek 2.7. Platforma mobilna wykorzystana w realizowanym projekcie.

Z powodu zastosowania dużych pompowanych kół oraz wysokich żelowych akumulatorów, powstał główny problem konstrukcyjny. Ramię budowanego robota musiało być dostatecznie długie i musiało posiadać możliwość wielokrotnego zginania się, aby móc dostać się w każde miejsce wokół podstawy jezdnej. Zostało przyjęte, że układ antropomorficzny z obrotową podstawą o długości około 1m, z pięciostopniową strukturą swobody, będzie wystarczający do celów projektu. Aby zapewnić optymalny dostęp do okolic kół robota, ustalono, że pierwsze dwa przegubu muszą mieć odpowiednio 400mm oraz 300mm długości dla pierwszego i drugiego ogniw. Równocześnie, napędy ramienia muszą zostać dobrane w taki sposób, aby konstrukcja posiadała około 500g udźwigu.

Ważnym elementem, przy zastosowanych ciężkich akumulatorach, było utrzymanie możliwie niskiej wagi manipulatora, tak aby nie obciążał kół i nie utrudniał poruszania się całego robota. Jednocześnie przy zachowaniu niskiej wagi, należało upewnić się, że sztywność konstrukcji będzie odpowiednia, aby nie wywoływać drgań w trakcie poruszania się.

W celu zapewnienia prostoty programowania i wykonywania obliczeń kinematycznych, założono użycie serwomechanizmów modelarskich różnego typu, w zależności od wymaganego momentu obrotowego dla konkretnego przegubu. Przybliżona masa użytych serwomechanizmów została uwzględniona w obliczeniach kinematycznych robota.

Na Rys. 2.8 przedstawiono szkic poglądowy opracowanego manipulatora z zaznaczonymi oszacowanymi długościami ogniw a w tabelach. 2.1 oraz 2.2 zawarto opis zaplanowanych układów napędowych.



Rysunek 2.8. Poglądowy rysunek manipulatora.

Tabela 2.1. Opis ogniw manipulatora.

Nazwa	Wymaga minimalna długość
d_1	Wysokość, która pozwala na umieszczenie łożyska i napędu
a_2	Około 400 mm
a_3	Około 300 mm
a_4	Około 250 mm
a_5	Około 150 mm (odległość do końca narzędzia manipulatora)

Tabela 2.2. Opis przegubów manipulatora.

Nazwa	Opis
S_p	Układ napędowy umożliwiający obrót podstawy
S_1	Układ napędowy umożliwiający ruch pierwszego przegubu
S_2	Układ napędowy umożliwiający ruch drugiego przegubu
S_3	Układ napędowy umożliwiający ruch trzeciego przegubu
S_4	Układ napędowy umożliwiający ruch czwartego przegubu

2.3.2. Analiza teoretyczna

Przed rozpoczęciem prac konstrukcyjnych dotyczących manipulatora, wykonano analizę doboru napędów oraz zasilania napędów na podstawie przewidywanej nośności robota oraz długości jego ramion. W poprzednim rozdziale opisano przyjęte wymagania dla długości przegubów oraz ustanowione zostało maksymalne, przewidywane obciążenie robota. Te dane były podstawą do analiz dotyczących wymaganych momentów mechanicznych dla każdego z napędów.

Obliczenia wykonano kolejno dla każdego z przegubów, od najdalej położonego do tego położonego najbliżej podstawy. Na potrzeby obliczeń przyjęto najmniej korzystną sytuację ułożenia całego ramienia – manipulator skierowany jest prostopadle do powierzchni ziemi oraz wszystkie przeguby są wyprostowane. Dla uproszczenia pominięto masę konstrukcji łączącej napędy – jest ona nieporównywalnie lżejsza od samych napędów.

Z uwagi na wybrany typ manipulatora, w obliczeniach użyto poniższego wzoru do obliczenia momentu obrotowego każdego z serw [10].

$$M_n = a_{n+1} \cdot m_{n+1} \cdot g + (a_{n+2} + a_{n+1}) \cdot m_{n+2} \cdot g + \dots \quad (1)$$

gdzie:

M_n – wymagany moment obrotowy przegubu;

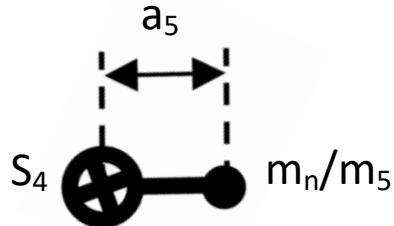
a_{n+1} – długość przegubu aktywnego napędu;

m_{n+1} – masa następnego przegubu;

g – przyśpieszenie grawitacyjne.

Poniżej przedstawiono obliczenia wykonane dla każdego z przegubów:

- Przegub S_4



Rysunek 2.9. Schemat ostatniego przegubu.

$$M_4 = a_5 \cdot (m_n + m_o) \cdot g \quad (2)$$

Gdzie:

$a_5 = 0,15 \text{ m}$ – długość piątego przegubu i narzędzia

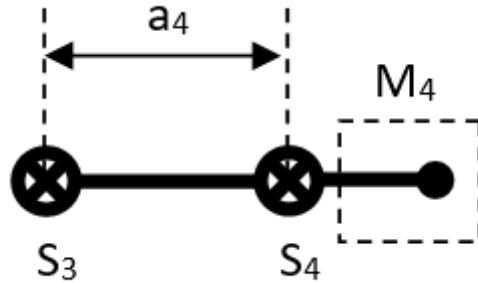
$m_n = 0,16 \text{ kg}$ – masa narzędzia

$m_o = 0,5 \text{ kg}$ – masa obiektu przenoszonego

M_4 – wymagany moment obrotowy napędu S_4

$$M_4 \approx 0,97 \text{ N} \cdot \text{m} \approx 9,9 \text{ kg} \cdot \text{cm}$$

- Przegub S_3



Rysunek 2.10. Schemat trzeciego przegubu.

$$M_3 = (a_5 + a_4) \cdot (m_n + m_o) \cdot g + a_4 \cdot m_{S4} \cdot g \quad (3)$$

Gdzie:

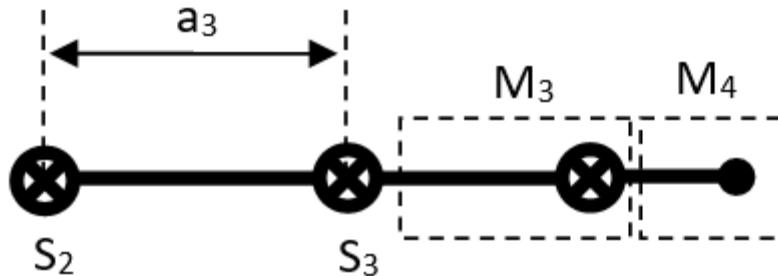
$a_4 = 0,25 \text{ m}$ – długość czwartego przegubu

$m_{S4} = 0,091 \text{ kg}$ – masa czwartego układu napędowego

M_3 – wymagany moment obrotowy napędu S_3

$$M_3 \approx 2,33 \text{ N} \cdot \text{m} \approx 23,76 \text{ kg} \cdot \text{cm}$$

- Przegub S_2



Rysunek 2.11. Schemat drugiego przegubu.

$$M_2 = (a_5 + a_4 + a_3) \cdot (m_n + m_o) \cdot g + (a_4 + a_3) \cdot m_{S4} \cdot g + a_3 \cdot m_{S3} \cdot g \quad (4)$$

Gdzie:

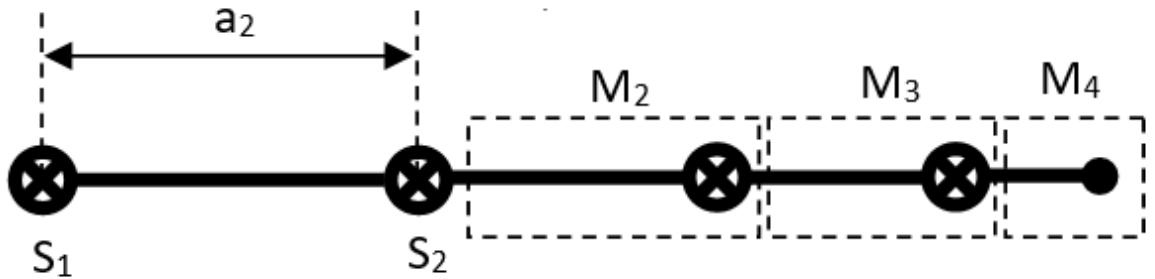
$a_3 = 0,3 \text{ m}$ – długość trzeciego przegubu

$m_{S3} = 0,091 \text{ kg}$ - masa trzeciego układu napędowego

M_2 – wymagany moment obrotowy napędu S_2

$$M_2 \approx 4,26 \text{ N} \cdot \text{m} \approx 43,4 \text{ kg} \cdot \text{cm}$$

- Przegub S_1



Rysunek 2.12. Schemat pierwszego przegubu

$$M_1 = (a_5 + a_4 + a_3 + a_2) \cdot (m_n + m_o) \cdot g + (a_4 + a_3 + a_2) \cdot m_{s4} \cdot g + (a_3 + a_2) \cdot m_{s3} \cdot g + a_2 \cdot m_{s2} \cdot g \quad (5)$$

Gdzie:

$a_2 = 0,4 \text{ m}$ – długość drugiego przegubu

$m_{s2} = 0,17 \text{ kg}$ - masa drugiego układu napędowego

M_1 – wymagany moment obrotowy napędu S_1

$$M_1 \approx 7,5 \text{ N} \cdot \text{m} \approx 76,51 \text{ kg} \cdot \text{cm}$$

Tabela 2.3. Parametry mechaniczne manipulatora.

Nr przegubu	Masa przegubu [kg]	Długość przegubu [m]	Wymagany moment układu napędowego [N·m]
1	0,17	0,4	7,5
2	0,091	0,3	4,26
3	0,091	0,25	2,33
4	0,76	0,15	0,97

2.3.3. Dobór układów napędowych oraz narzędzia dla manipulatora

W większości aplikacji robotyki mobilnej używa się dwóch typów napędów: silników krokowych lub serwonapędów elektrycznych. Oba z te rozwiązania były rozważane jako możliwe do zastosowania w opracowywane aplikacji. Każde z rozwiązań ma swoje wady oraz zalety.

Pierwsze rozwiązanie, z użyciem silników krokowych, jest często stosowane w aplikacjach zrobotyzowanych, które wymagają precyzyjnych, takich jak maszyny CNC czy manipulatory. Niestety, w takim rozwiązaniu nie można zrealizować sterowania silnikiem bezpośrednio z kontrolera. Wymagane są dodatkowe układy sterujące, co znacznie zwiększa stopień skomplikowania i potencjalnej awaryjności układu. Drugą wadą zastosowania silników krokowych, jest brak pętli sprzężenia zwrotnego. Bez dodatkowego czujnika położenia nie można stwierdzić, czy zadane położenie zostało osiągnięte. Dodatkową trudnością jest konieczność przekształcenia zadanej liczby impulsów silnika na jego położenie kątowe. Brak czujnika położenia generującego sygnał pętli sprzężenia zwrotnego, powoduje również, że po wyłączeniu zasilania sterownika nie jest znane położenie poszczególnych osi. Ten problem można rozwiązać stosując procedurę kalibracyjną, jednak jest to mało wygodne i komplikujące działanie całego manipulatora. Po przeanalizowaniu powyższych

problemów stwierdzono, że zastosowanie w projekcie silników krokowych, wymagających równocześnie sterowników i dodatkowych czujników położenia, jest zbyt skomplikowane.

Drugą z opcji, które były rozważane, są servomechanizmy elektryczne, często używane w tańszych i prostszych aplikacjach robotyki oraz w modelarstwie. Servomechanizm tego typu zbudowany jest z silnika (najczęściej prądu stałego), przekładni mechanicznej oraz enkodera absolutnego, którego rolę pełni potencjometr. Zadawanie pozycji odbywa się poprzez dostarczenie do układu sterowania mechanizmem sygnału PWM o odpowiednim wypełnieniu. Przykładowa budowa takiego servomechanizmu pokazana jest na Rys. 2.13.

Dzięki wbudowanej pętli sprzężenia zwrotnego, opartej na enkoderze absolutnym, zapamiętywanie ułożeń przegubów oraz powrót do poprzedniej pozycji są stosunkowo proste w implementacji. Prosty sposób sterowania nie wymaga użycia dodatkowych sterowników. Wystarczy dobrać kontroler, który zapewni odpowiednią liczbę kanałów generujących sygnał PWM.



Rysunek 2.13 Przekrój budowy servomechanizmu stosowanego w modelarstwie [11].

W poniższej części tego rozdziału opisane zostaną użyte servomechanizmy lub ich konfiguracje.

- Podstawa S_p

Servomechanizm użyty w podstawie nie wymagał dużego momentu obrotowego z uwagi na zastosowanie odpowiednich łożysk oraz zastosowane smarowanie konstrukcji. Duży promień podstawy, pod którą znajduje się servomechanizm, pozwolił na zmniejszenie sił działających tnąco oraz skośnie. Wymagany moment obrotowy był tak mały, że zastosowano servomechanizm jak na Rys. 2.15 o parametrach jak w Tab. 2.4.

- Przegub S_1

Żaden z dostępnych na rynku w analizowanym przedziale cenowym servomechanizmów nie miał możliwości utrzymania, obliczonego wcześniej, obciążenia około 8 Nm. Z tego powodu, w pierwszym przegubie zastosowane zostały dwa servomechanizmy typu gigant, ułożone względem siebie lustrzanie. Pozwoliło to na podwojenie uzyskanego momentu i osiągnięcie wymaganych parametrów.

Widok zastosowanego serwomechanizmu przedstawiono na Rys. 2.14, natomiast jego parametry zawiera Tab. 2.3.



Rysunek 2.14. Serwomechanizm Power HD 1235MG [12].

Tabela 2.4. Parametry serwonapędu Power HD 1235 MG.

Nazwa modelu	<i>Power HD 1235MG</i>
Wymiary [mm]	59,5 X 29,5 X 54,3
Waga [g]	165
Kąt obrotu [°]	180
Napięcie zasilania [V]	6,0 – 7,4
Maksymalny pobór prądu [A]	9
Moment obrotowy dla 6,0 V [Nm]	3,4
Moment obrotowy dla 7,4 V [Nm]	4,0

- Przegub S₂

Drugi przegub manipulatora miał wymagany znacznie mniejszy moment obrotowy, pozwoliło to na zastosowanie tylko jednego serwomechanizmu, o dokładnie takich samych parametrach jak w przegubie poprzednim.

- Przegub S₃

Trzeci przegub był na tyle krótki, że nie wymagał zastosowania serwomechanizmu o dużym momencie. Na podstawie wyznaczonego momentu obrotowego dobrano serwomechanizm w rozmiarze „normal”. Ze względu na to, że producent nie podaje katalogowej wartości maksymalnego prądu obciążenia, przed dobraniem układów zasilania wartość ta została wyznaczona eksperymentalnie. Serwomechanizm wybrany dla osi 3 przedstawia Rys. 2.15, natomiast jego parametry zestawiono w Tab. 2.4.



Rysunek 2.15. Serwomechanizm JX PDI-6221MG [13].

Tabela 2.5. Parametry serwonapędu JX PDI-6221MG.

Nazwa modelu	JX PDI-6221MG
Wymiary [mm]	40,5X20,2X38
Waga [g]	62
Kąt obrotu [°]	180
Napięcie zasilania [V]	4,8 – 6,0
Maksymalny pobór prądu [A]	Ok. 3 (sprawdzona doświadczalnie)
Moment obrotowy dla 4,8 V [Nm]	1,7
Moment obrotowy dla 6,0 V [Nm]	2,2

- Przegub S₄

Serwomechanizm przegubu czwartego, podobnie jak w poprzednim przypadku, nie wymagało zbyt dużego momentu obrotowego. Aby ułatwić montaż narzędzia na jego końcu, wybrano mechanizm z dwustronną osią napędową oraz aluminiową ramką. Pozwoliło to uprościć znacznie konstrukcję manipulatora oraz pomogło w podniesieniu jej sztywności i stabilności. W przypadku tego produktu producent nie podał katalogowych wartości prądu maksymalnego. Zostały one wyznaczone eksperymentalnie w trakcie prac laboratoryjnych. Widok zastosowanego dla przegubu 4 serwomechanizmu przedstawia Rys. 2.16, natomiast jego parametry nominalne Tab. 2.5.



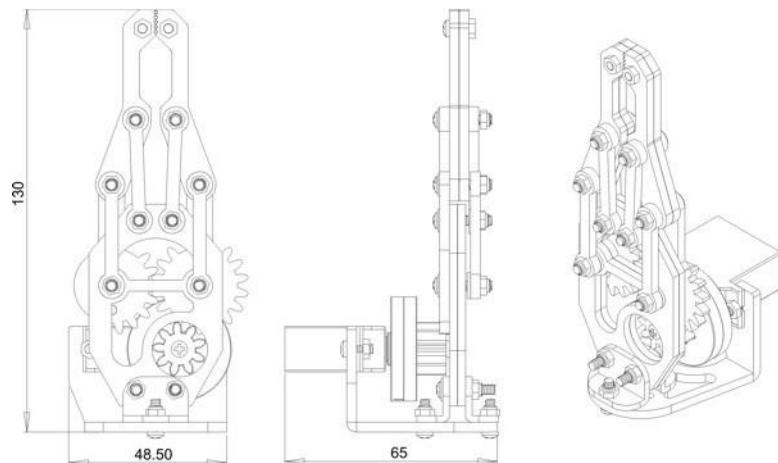
Rysunek 2.16. Serwomechanizm ServoS3315D zastosowany dla przegubu 4 [14].

Tabela 2.6. Parametry serwonapędu Servo S3315D.

Nazwa modelu	Servo S3315D
Wymiary [mm]	40X20X40,5
Waga [g]	60
Kąt obrotu [°]	180
Napięcie zasilania [V]	4,8 – 7,4
Maksymalny pobór prądu [A]	Ok. 2 (sprawdzona doświadczalnie)
Moment obrotowy dla 6,0 V [Nm]	1,47
Moment obrotowy dla 7,4 V [Nm]	1,57

- Napęd narzędzia S_n oraz chwytak

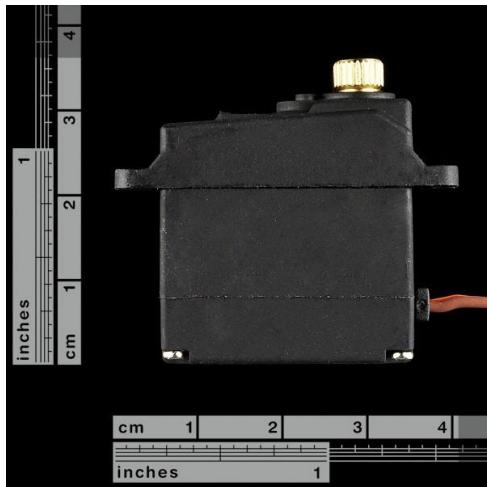
Użyty układ narzędziowy to gotowy chwytak *RoboticClaw MKII* [15]. Jest to model posiadający wbudowane sprzęgło sprężynowe, zapobiegające zablokowaniu się chwytaka, uniemożliwiające przeciążenie mechanizmu napędowego. Widok chwytaka oraz jego wymiary przedstawiono na Rys. 2.17, jego parametry zawiera Tab. 2.6. Chwytak zbudowano i zaprojektowano z myślą o wykorzystaniu w jego napędzie serwonapędu modelarskiego typu medium. W zestawie z samym chwytakiem dołączony został odpowiedni napęd *DSG S90 S05NF* [16], przedstawione na Rys. 2.18. Parametry serwomechanizmu zawiera Tab. 2.7.



Rysunek 2.17. Chwytak *RoboticClaw MKII* [15].

Tabela 2.7. Parametry chwytaka *RoboticClaw MKII*.

Nazwa modelu	<i>RoboticClaw MKII</i>
Wymiary [mm]	130X48,5X65 (z zamocowanym serwomechanizmem)
Waga [g]	130
Użyta przekładnia	2 : 1
Rozstaw szczelek po otwarciu [mm]	50



Rysunek 2.18. Serwomechanizm DSG S90 S05NF – medium [16].

Tabela 2.8. Parametry serwonapędu DSG S90 S05NF.

Nazwa modelu	<i>DSG S90 S05NF - medium</i>
Wymiary [mm]	28,8 x 13,8 x 30,2
Waga [g]	20
Kąt obrotu [°]	180
Napięcie zasilania [V]	4,8 – 6,0
Maksymalny pobór prądu [A]	-
Moment obrotowy dla 4,8 V [Nm]	0,27
Moment obrotowy dla 6,0 V [Nm]	0,31

Wszystkie serwomechanizmy w projektowanym manipulatorze zostały dobrane tak, by nigdy, nawet w najbardziej niekorzystnej pozycji, nie zostały przekroczone ich maksymalne udźwigi.

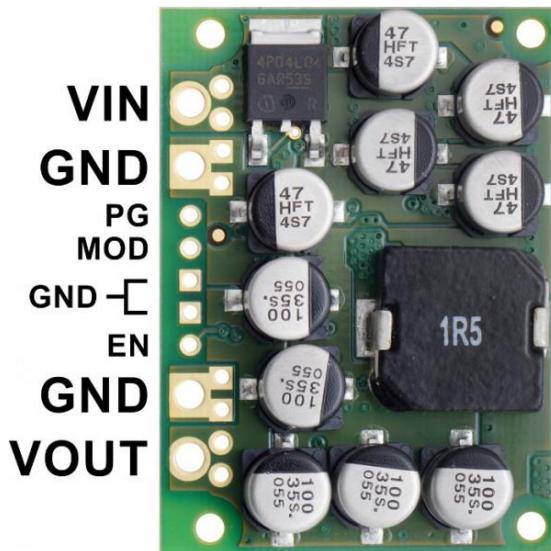
2.3.4. Dobór układów zasilania dla napędów

Głównym źródłem zasilania całego robota są dwa żelowe akumulatory połączone szeregowo, o łącznym nominalnym napięciu wyjściowym 24,0 V. Do zasilania układów napędowych użyto przetwornic DC/DC typu *step down*. Do zasilania wszystkich zainstalowanych serwomechanizmów potrzebne były trzy przetwornice zasilające o różnych wartościach napięć wyjściowych. O ile w przypadku serwomechanizmu chwytaka wydajność prądowa mogła być niewielka, to moc pozostałych dwóch zasilaczy musiała być wystarczająca do zasilenia pozostałych serwomechanizmów. Do zasilania napędu przegubu narzędziowego potrzebna była jednostka o napięciu wyjściowym 5,0 V i prądzie minimum 0,5 A. Wartości te ustalono na podstawie pomiarów laboratoryjnych. Serwomechanizmy podstawy, trzeciego i czwartego przegubu wymagają zasilenia napięciem 6,0V oraz wydajności prądowej minimum 6,0A całej przetwornicy (przy założeniu, że nigdy wszystkie trzy napędy nie wejdą w tryb maksymalnego poboru mocy równocześnie). Ostatnie trzy serwomechanizmy wymagają zasilenia napięciem 7,4V i wydajności prądowej około 25,0 A.

- Zasilanie napędów S₁ oraz S₂

Do zasilania największych układów napędowych użyto dwóch dostępnych komercyjnie przetwornic marki *PoloIu* [17], widok przetwornicy przedstawiono na Rys. 2.19, a jej parametry zawarto w Tab. 2.9. Wymagana wydajność prądowa przekracza znacznie prąd znamionowy przetwornicy, więc postanowiono podłączyć dwa współpracujące ze sobą silniki do jednej przetwornicy, a pozostały napęd, stanowiący pojedynczy przegub, do drugiej z przetwornic. Ten typ

przetwornicy został wybrany ze względu na niską cenę i dopasowane parametry prądowe. Zastosowane mechanizmy dobrze współpracują z napięciem 7,5V.



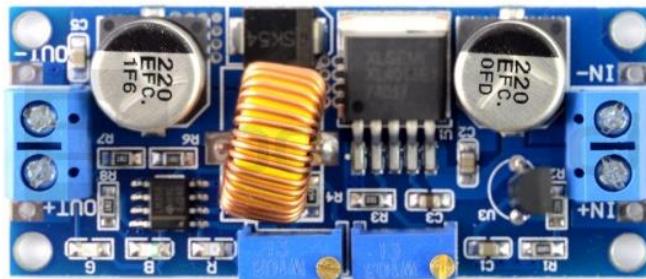
Rysunek 2.19. Pololu D24V150F7 [17].

Tabela 2.9. Parametry przetwornicy Pololu D24V150F7.

Nazwa	Pololu D24V150F7
Wymiary [mm]	43X32X11
Napięcie wejściowe [V]	9 – 40
Napięcie wyjściowe [V]	7,5
Maksymalny prąd ciągły [A]	15

- Zasilanie napędów S_p , S_3 oraz S_4

Do zasilania pozostałych trzech układów napędowych użyto jednej, gotowej przetwornicy XL4015 [18]. Widok przetwornicy przedstawiono na Rys. 2.20, a jej parametry w Tab. 2.10. Z uwagi na bardzo małe obciążenie napędu podstawy, założono, że jeden taki układ zasilający wystarczy. Dzięki zastosowaniu regulowanej przetwornicy można ustawić najbardziej optymalne napięcie pracy układu. W tym przypadku wynosiło ono 6 V, czyli było równe maksymalnemu dopuszczalnemu napięciu dla mechanizmów podstawy i trzeciego przegubu.



Rysunek 2.20. Przetwornica XL4015 [18].

Tabela 2.10. Parametry przetwornicy XL4015.

Nazwa	XL4015
Wymiary [mm]	62X26X12
Napięcie wejściowe [V]	3,3 – 36,0
Napięcie wyjściowe [V]	1,3 – V_{in}
Maksymalny prąd ciągły [A]	5

- Zasilanie napędów S_n

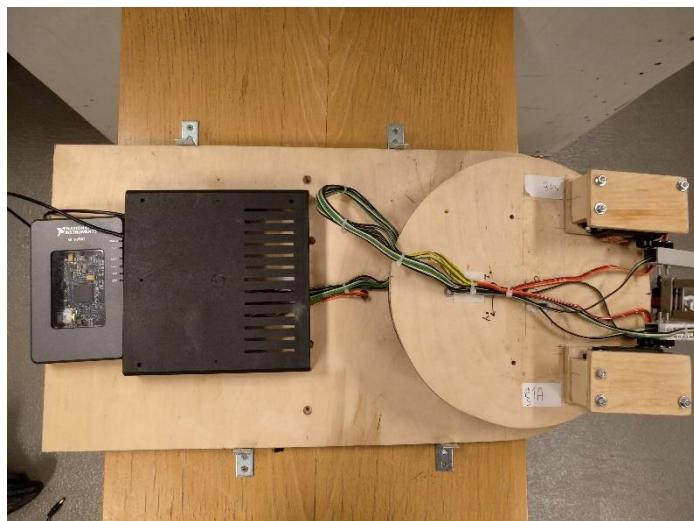
Na moduł zasilający tego napędu nie były nałożone żadne wygórowane warunki. Spełnia je przetwornica *XL4015*. Dostarcza ona wystarczający prądy i napięcie, aby zasilić serwomechanizm narzędziowy.

2.3.5. Budowa robota

W tej części pracy przedstawiono zagadnienia związane z konstrukcją manipulatora. Opisano szczegóły dotyczące konstrukcji robota oraz mocowania serwonapędów zilustrowane odpowiednim zdjęciemi. W kolejnych podrozdziałach przedstawiono modyfikacje jakim poddano serwomechanizmy, sposoby ich mocowania do konstrukcji manipulatora, zastosowane metody usztywnienia konstrukcji oraz mechanizm obracania podstawy manipulatora i sposób jego łożyskowania.

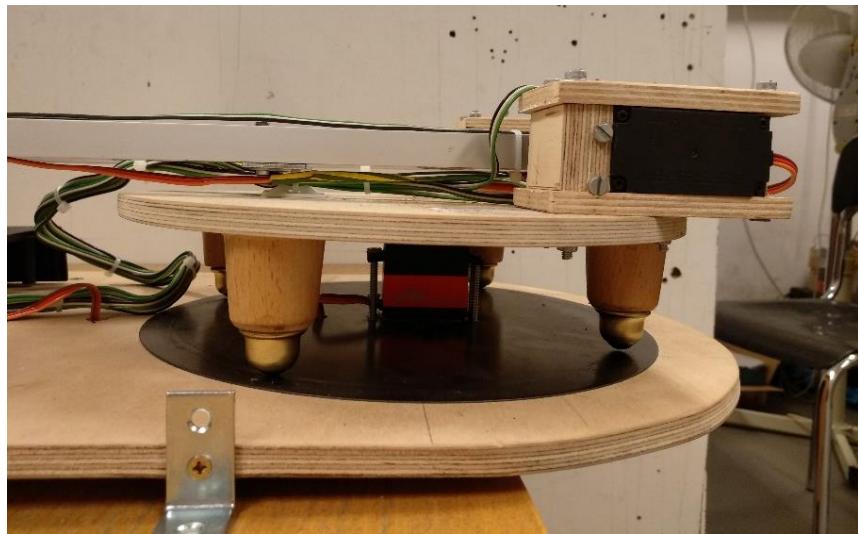
- Podstawa manipulatora i łożyskowanie

Podstawa manipulatora została wykonana ze sklejki o grubości 7mm. Dolna część podstawy jest okręgiem o średnicy 270 mm, przedłużonym w taki sposób, aby jego tylna część stanowiła odpowiednie miejsce do umieszczenia układów zasilających oraz procesora. Ostateczna konstrukcja przedstawiona jest na Rys 2.21.



Rysunek 2.21. Dolna część podstawy wraz z umieszczoną wyżej obrotową platformą.

Konstrukcja mechanizmu umożliwiającego obracanie się podstawy manipulatora jest następująca. Do dolnej części podstawy, będącej jednocześnie obszarem pracy łożysk, poprzez serwomechanizm zamontowana jest górną podstawa obrotowa. Z uwagi na dość duży rozmiar podstawy obrotowej, żadne z dostępnych łożysk maszynowych nie pozwalało na odpowiednie przeniesienie sił oraz równoczesne umożliwienie łatwego dostępu serwisowego do mechanizmu serwonapędu obracającego platformą. Z tego powodu autor zastosował łatwo dostępne w większości sklepów meblowych, łożyska meblowe. Są to wykonane z twardego metalu kulki umieszczone w specjalnej obudowie, które po odpowiednio twardej powierzchni poruszają się praktycznie bez tarcia. W celu zapewnienia odpowiednio twardego podłoża dla łożyska, użyto stalowej płyty pokrytej teflonem. Obie części połączone są przez sam mechanizm napędowy, którego jedna zębatka umieszczona jest na stałe na górnej części, a cała reszta mechanizmu napędowego przykręcona jest do dolnej części podstawy, jak na Rys 2.22.

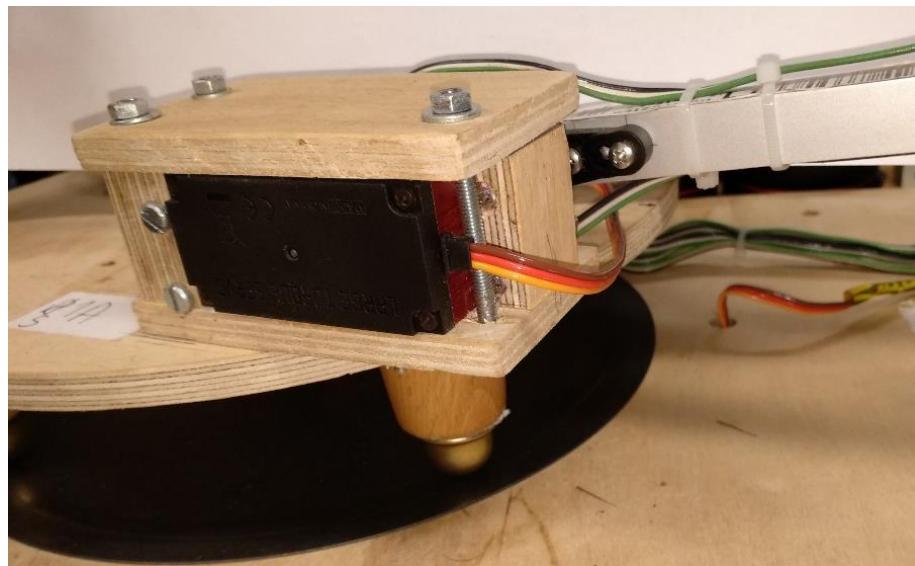


Rysunek 2.22. Połączenie dolnej i górnej części podstawy robota

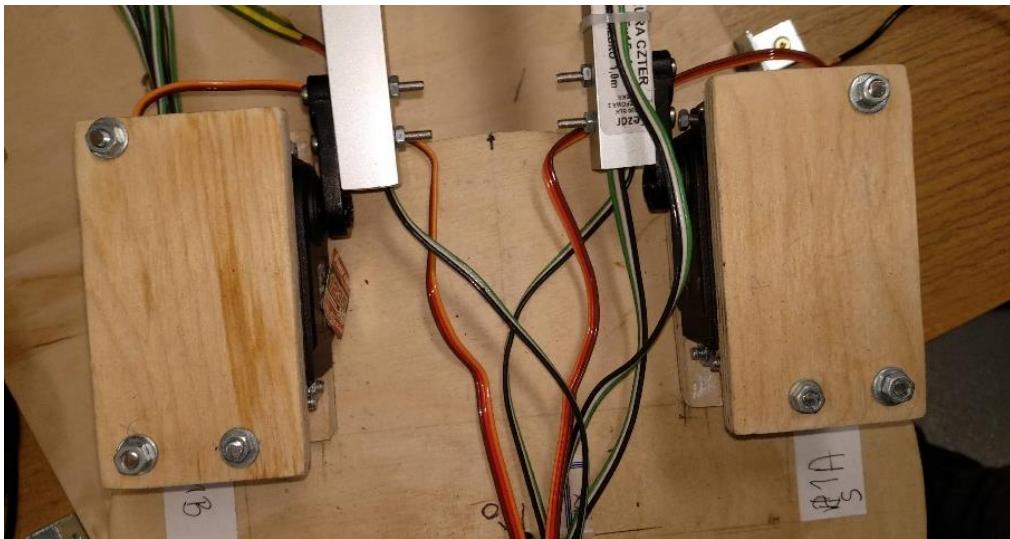
- **Pierwszy przegub manipulatora**

Pierwszy z przegubów manipulatora składa się z dwóch współpracujących ze sobą napędów. Ten zestaw serwonapędów jest przymocowany bezpośrednio do praformy obrotowej. Z uwagi na największe działające na niego siły, należało upewnić się, że będzie on odpowiedni mocno przymocowany. Bardzo ważnym było, aby układy nie były wprowadzane w drgania z powodu złego zamocowania. Takie drgania przenoszone byłyby na resztę układów napędowych i powodowałyby duże zakłócenia, których amplituda roślaby wykładniczo dla każdego kolejnego przegubu.

Serwomechanizmy zostały obudowane, dokładnie ze wszystkich stron, drewnianymi ściągaczami, które dzięki przechodzącym na wylot śrubom, umożliwiły bardzo mocne ściśnięcie napędu w wymaganym miejscu. Dodatkowo, aby jeszcze wzmacnić zakleszczenie silników w ich uchwytach, na mechanizmy zostały naklejone trzy warstwy taśmy izolacyjnej, która dzięki swoim elastycznym właściwościom, jeszcze mocniej umocowała napędy. Całą strukturę mocowania przedstawiona została na rysunkach 2.23 oraz 2.24.



Rysunek 2.23. Mocowanie napędu przegubu pierwszego.



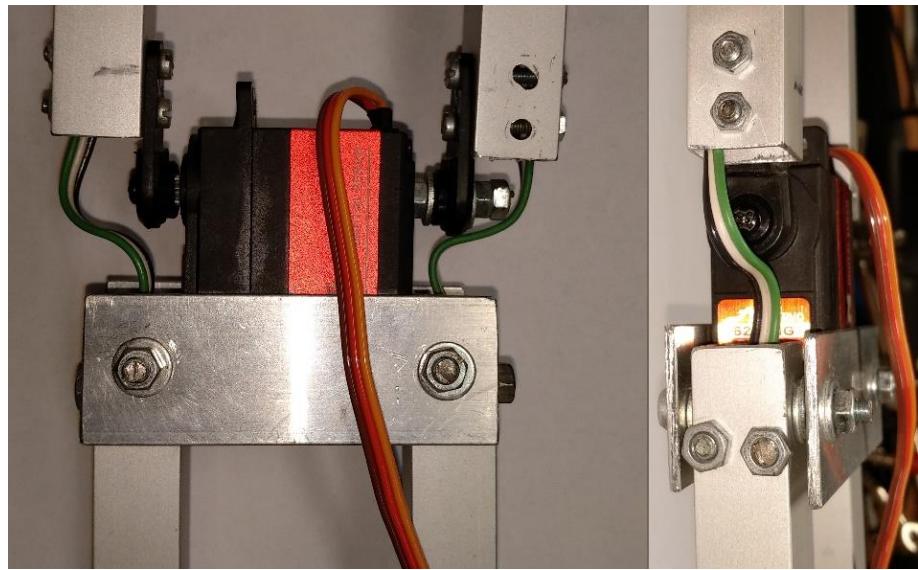
Rysunek 2.24. Układ napędowy przegubu pierwszego.

- **Drugi i trzeci przegub manipulatora**

Mocowanie dwóch kolejnych układów napędowych przeprowadzono w analogiczny sposób. Przed zamocowaniem oba servomechanizmy zostały zmodyfikowane w układy dwustronne, zgodnie z założeniami projektowymi. Uznano, że skoro układy nie mają zastosowanych żadnych zewnętrznych przekładni i są stosunkowo lekkie, to nie zachodzi potrzeba przenoszenia ich poza przegub. Zgodnie z takim założeniem sam układ napędowy został użyty jako element konstrukcyjny danej części ramienia. Nieruchoma część napędu jest na stałe przykręcana do poprzedzającego przegub ramienia oraz dodatkowo zabezpieczona z dwóch stron metalowym łącznikiem. Taka konstrukcja, przedstawiona na rysunkach 2.25 i 2.26, zapewnia stabilniejszy uchwyt i zmniejszenie wychyleń pod wpływem działających sił. Kolejne ramię opiera się bezpośrednio na osi napędowej oraz dodanej osi wspierającej.



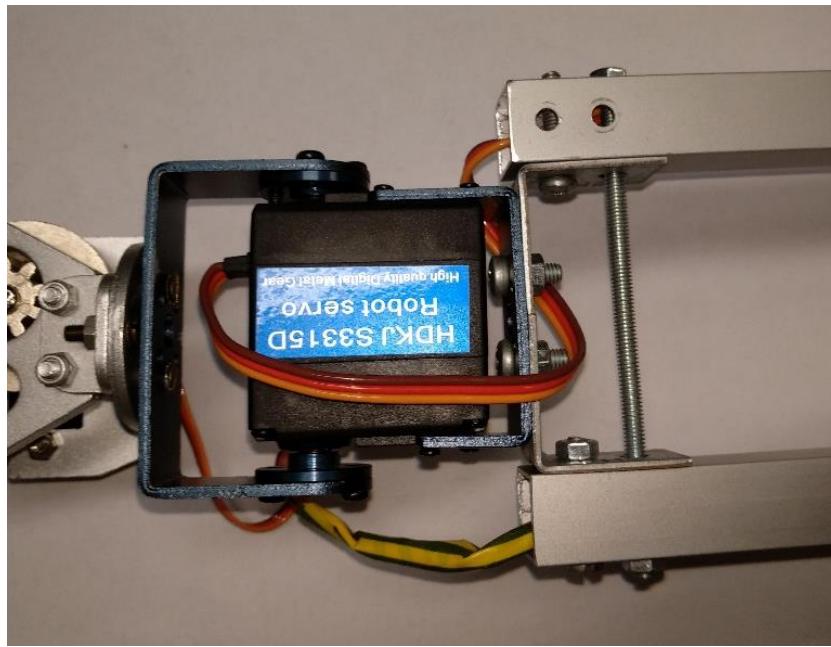
Rysunek 2.25. Układ napędowy przegubu drugiego.



Rysunek 2.26. Układ napędowy przegubu trzeciego.

- **Przegub czwarty manipulatora**

Ostatni z zastosowanych przegubów był najprostszym do zamocowania. Serwonapęd, który został użyty, był fabrycznie wyprodukowany jako dwuosiowy, a w zestawie wraz z układem napędowym producent zapewnił również odpowiednie aluminiowe ramki, które pozwalają na wykonanie mocowania za pomocą kilku śrub i nakrętek.



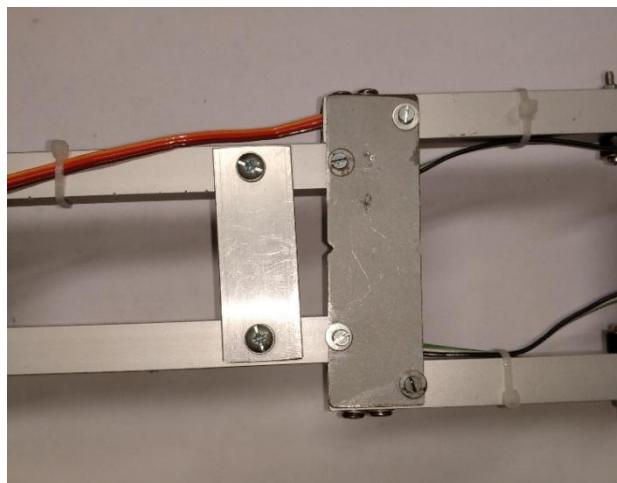
Rysunek 2.27. Układ napędowy czwartego przegubu.

- **Ogniwa łączące poszczególne przeguby**

Z uwagi na konieczność uzyskania sztywnej konstrukcji manipulatora, przy jednoczesnym zachowaniu jej lekkości oraz łatwości łączenia elementów, do budowy ogniw łączących przeguby użyto aluminiowych profili 10 x 10 mm, jak na Rys. 2.28. Równocześnie, na dwóch najdłuższych ogniwach użyto aluminiowych łączników w celu usztywnienia konstrukcji. Bez nich, przy niektórych ustawieniach, działające siły skręcające powodowałyby wyginanie się manipulatora, pokazane na Rys 2.29.



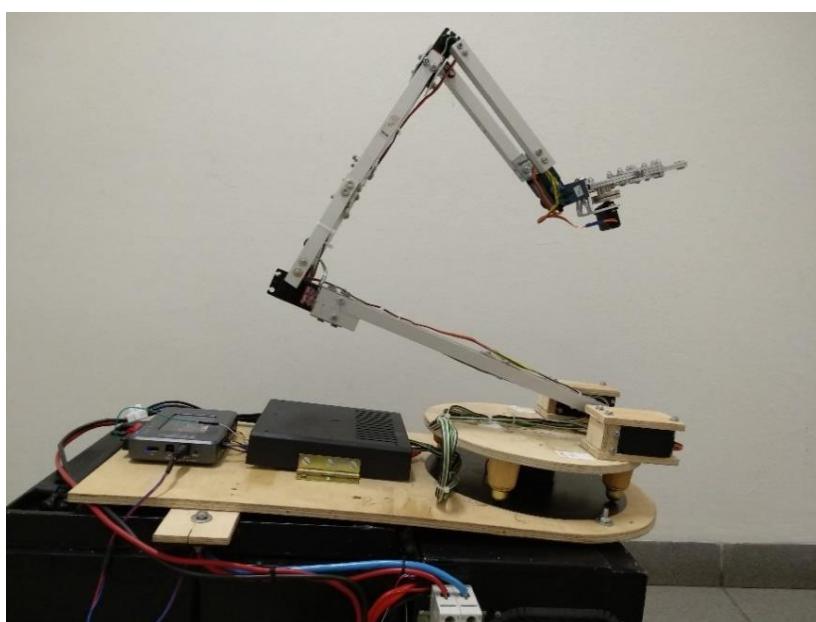
Rysunek 2.28. Aluminiowy profil [19].



Rysunek 2.29. Przykład połączenia usztywniającego ogniw.

- Ostateczna konstrukcja manipulatora

Ostatecznie uzyskano konstrukcję manipulatora przedstawioną na Rys. 2.30. Charakteryzuje się pięcioma stopniami swobody oraz typem budowy antropomorficznej. Przedstawiona konstrukcja spełnia wszystkie postawione wcześniej założenia konstrukcyjne i dynamiczne.

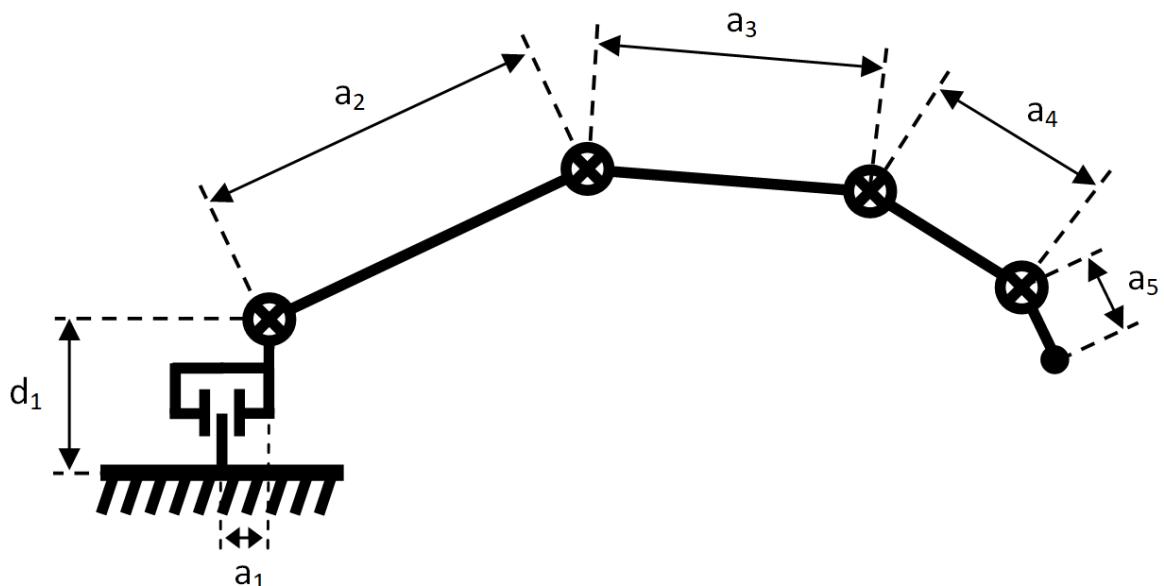


Rysunek 2.30. Gotowa konstrukcja razem z układami zasilania i sterowania.

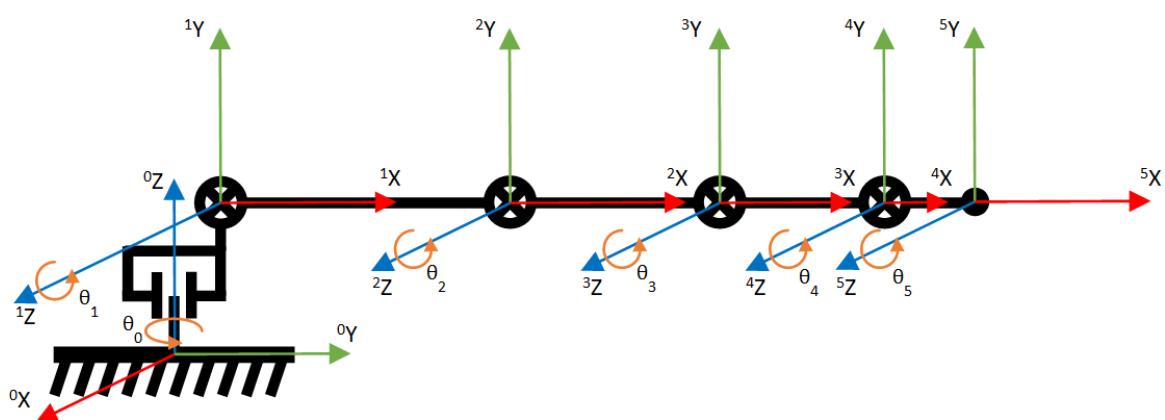
2.4. Opis matematyczny sterowania manipulatorem

2.4.1. Opis ramienia w notacji DH

Skonstruowany manipulator jest rozbudowanym manipulatorem antropomorficznym klasy 5R. Posiada on pięć stopni swobody. W skład robota wchodzi pionowa kolumna obrotowa o wysokości d_1 , o stałym przesunięciu względem osi obrotu o wartości a_1 . Reszta konstrukcji to czteroczęściowe ramię składające się z czterech przegubów i czterech ogniw łączących. Długości ogniw wynoszą odpowiednio a_2, a_3, a_4 oraz a_5 , gdzie w długości ostatniego ognia zawiera się długość układu narzędziowego. Schemat wraz z dopasowanymi układami przedstawiono na Rys. 2.34.



Rysunek 2.31. Poglądowy rysunek manipulatora z przesunięciami.



Rysunek 2.32. Układy kinematyczne w manipulatorze.

Opis matematyczny układów wykonano przy zastosowaniu notacji Denawita-Hartenberga [20] [21]. Tabela 2.10 przedstawia parametry i zmienne wiążące układy poszczególnych ogniw.

Tabela 2.11. Parametry notacji DH opracowanego manipulatora

Numer ogniwa	a_i	α_i	d_i	Θ_i
1	a_1	90°	d_1	Θ_1
2	a_2	0	0	Θ_2
3	a_3	0	0	Θ_3
4	a_4	0	0	Θ_4
5	a_5	0	0	Θ_5

Przed wyznaczeniem macierzy transformacji między układami wyznaczono rodzaje transformacji względem osi układów dla każdego przegubu.

$$\begin{aligned}
 A_0^1(q_1) &= \mathbf{Rot}(Z, \theta_1) \mathbf{Trans}(Z, d_1) \mathbf{Rot}\left(X, \frac{\pi}{2}\right) \mathbf{Trans}(X, a_1) \\
 A_1^2(q_2) &= \mathbf{Rot}(Z, \theta_2) \mathbf{Trans}(X, a_2) \\
 A_2^3(q_3) &= \mathbf{Rot}(Z, \theta_3) \mathbf{Trans}(X, a_3) \\
 A_3^4(q_4) &= \mathbf{Rot}(Z, \theta_4) \mathbf{Trans}(X, a_4) \\
 A_4^5(q_5) &= \mathbf{Rot}(Z, \theta_5) \mathbf{Trans}(X, a_5)
 \end{aligned} \tag{6}$$

Kolejnym krokiem było zapisanie macierzy transformacji między kolejnymi układami, gdzie $c = \cos$, a $s = \sin$:

- Przejście między układami 0 i 1:

$$A_0^1(q_1) = \begin{bmatrix} c\theta_1 & 0 & s\theta_1 & a_1 c\theta_1 \\ s\theta_1 & 0 & -c\theta_1 & a_1 s\theta_1 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

- Przejście między układami 1 i 2:

$$A_1^2(q_2) = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & a_2 c\theta_2 \\ s\theta_2 & c\theta_2 & 0 & a_2 s\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8}$$

- Przejście między układami 2 i 3:

$$\mathbf{A}_2^3(q_3) = \begin{bmatrix} c\theta_3 & -s\theta_3 & 0 & a_3c\theta_3 \\ s\theta_3 & c\theta_3 & 0 & a_3s\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (9)$$

- Przejście między układami 3 i 4:

$$\mathbf{A}_3^4(q_4) = \begin{bmatrix} c\theta_4 & -s\theta_4 & 0 & a_4c\theta_4 \\ s\theta_4 & c\theta_4 & 0 & a_4s\theta_4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (10)$$

- Przejście między układami 4 i 5:

$$\mathbf{A}_4^5(q_5) = \begin{bmatrix} c\theta_5 & -s\theta_5 & 0 & a_5c\theta_5 \\ s\theta_5 & c\theta_5 & 0 & a_5s\theta_5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (11)$$

Po wyznaczeniu transformacji między pojedynczymi inkrementacjami układów obliczono przejście z układu $\mathbf{A}_0^5(q)$, co zapisano jako $\mathbf{K}(q)$:

$$\mathbf{K}(q) = \mathbf{A}_0^1 \mathbf{A}_1^2 \mathbf{A}_2^3 \mathbf{A}_3^4 \mathbf{A}_4^5(q) : X_0 Y_0 Z_0 \rightarrow X_5 Y_5 Z_5, \quad (12)$$

więc końcowa postać macierzy została zapisana jako macierz 4×4 . Ostatnia kolumna reprezentuje przesunięcie w osiach x, y, z . Przesunięcia te oznaczono literami p z indeksem. Rotacje natomiast reprezentują wszystkie elementy oznaczone literami r .

$$\mathbf{K}(q) = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (13)$$

Dla uproszczenia zapisu i czytelności poniżej przedstawiono w postaci układu równań wszystkie elementy macierzy.

$$\left\{ \begin{array}{l} r_{11} = \cos(\theta_2 + \theta_3 + \theta_4 + \theta_5) \cdot \cos \theta_1 \\ r_{12} = -\sin(\theta_2 + \theta_3 + \theta_4 + \theta_5) \cdot \cos \theta_1 \\ r_{13} = \sin \theta_1 \\ r_{21} = \cos(\theta_2 + \theta_3 + \theta_4 + \theta_5) \cdot \sin \theta_1 \\ r_{22} = \sin(\theta_2 + \theta_3 + \theta_4 + \theta_5) \cdot \sin \theta_1 \\ r_{23} = -\cos \theta_1 \\ r_{31} = \sin(\theta_2 + \theta_3 + \theta_4 + \theta_5) \\ r_{32} = \cos(\theta_2 + \theta_3 + \theta_4 + \theta_5) \\ r_{33} = 0 \\ \\ p_x = \cos \theta_1 \cdot (a_1 + a_5 \cdot \cos(\theta_2 + \theta_3 + \theta_4 + \theta_5) + a_4 \cdot \cos(\theta_2 + \theta_3 + \theta_4) \\ \quad + a_3 \cdot \cos(\theta_2 + \theta_3) + a_2 \cdot \cos \theta_2) \\ \\ p_y = \sin \theta_1 \cdot (a_1 + a_5 \cdot \cos(\theta_2 + \theta_3 + \theta_4 + \theta_5) + a_4 \cdot \cos(\theta_2 + \theta_3 + \theta_4) \\ \quad + a_3 \cdot \cos(\theta_2 + \theta_3) + a_2 \cdot \cos \theta_2) \\ \\ p_z = d_1 + a_5 \cdot \sin(\theta_2 + \theta_3 + \theta_4 + \theta_5) + a_4 \cdot \sin(\theta_2 + \theta_3 + \theta_4) \\ \quad + a_3 \cdot \sin(\theta_2 + \theta_3) + a_2 \cdot \sin \theta_2 \end{array} \right. \quad (14)$$

2.4.2. Wykorzystanie równań kinematyki prostej do obliczania pozycji robota

Wyznaczenie obliczonych w poprzednim rozdziale wartości wyrażeń $\mathbf{p}_x, \mathbf{p}_y, \mathbf{p}_z$, dla odpowiednich wartości długości ogniw oraz odczytanych zgięć przegubów pozwoliło na wyznaczenie we współrzędnych kartezjańskich bazowych umiejscowienia końca manipulatora. Operacja ta realizowana jest w czasie rzeczywistym przez procesor znajdujący się na robocie. Dzięki wsparciu środowiska *Matlab* przez producenta wybranego procesora (*NationalInstruments*) możliwa jest implementacja kodu wyznaczającego współrzędne x, y, z punktu TCP manipulatora bezpośrednio z programu *Matlab*.

Podczas realizacji zagadnienia w projekcie pierwszym krokiem prac było odczytanie wartości zgięć przegubów manipulatora i przekształcenie ich ze stopni na radiany, tak aby wartości mogły być odczytane przez zastosowany skrypt obliczeniowy. Kolejnym krokiem było wpisanie odczytywanych wartości zgięć we wcześniej przygotowane miejsca w macierzach. Następnie obliczono całkowity wektor przesunięcia i pobrano jego ostatnią kolumnę w celu uzyskania wartości przesunięcia względem punku bazowego $P = [0,0,0]$ znajdującego się na osi dolnej podstawy. Cała procedura matematyczna pokazana jest na Rys. 2.33.

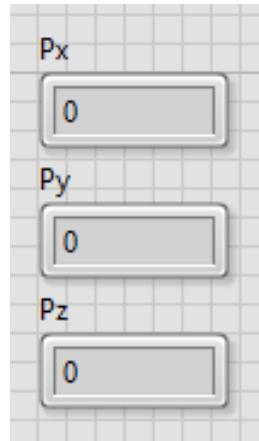
```

1 % zmiana stopnie na radiany
2 Sp = deg2rad(Spn);
3 S1 = deg2rad(S1n);
4 S2 = deg2rad(S2n);
5 S3 = deg2rad(S3n);
6 S4 = deg2rad(S4n);
7
8 % macierze DH
9 A1 = [cos(Sp) 0 sin(Sp) x1*cos(Sp); sin(Sp) 0 -cos(Sp) x1*sin(Sp); 0 1 0 1; 0 0 0 1];
10 A2 = [cos(S1) -sin(S1) 0 x2*cos(S1); sin(S1) cos(S1) 0 x2*sin(S1); 0 0 1 0; 0 0 0 1];
11 A3 = [cos(S2) -sin(S2) 0 x3*cos(S2); sin(S2) cos(S2) 0 x3*sin(S2); 0 0 1 0; 0 0 0 1];
12 A4 = [cos(S3) -sin(S3) 0 x4*cos(S3); sin(S3) cos(S3) 0 x4*sin(S3); 0 0 1 0; 0 0 0 1];
13 A5 = [cos(S4) -sin(S4) 0 x5*cos(S4); sin(S4) cos(S4) 0 x5*sin(S4); 0 0 1 0; 0 0 0 1];
14
15 % macierz wymnożona
16 T4 = A1*A2*A3*A4*A5;
17
18 % wektory przesunięcia
19 Px = 5*cos(Sp)*(41*cos(S1 + S2 + S3) + 33*cos(S1 + S2 + S3 + S4) + 62*cos(S1 + S2) + 80*cos(S1) + 12);
20 Py = 5*sin(Sp)*(41*cos(S1 + S2 + S3) + 33*cos(S1 + S2 + S3 + S4) + 62*cos(S1 + S2) + 80*cos(S1) + 12);
21 Pz = 205*sin(S1 + S2 + S3) + 165*sin(S1 + S2 + S3 + S4) + 310*sin(S1 + S2) + 400*sin(S1) + 88;

```

Rysunek 2.33. Skrypt służący do analitycznego wyznaczenia pozycji manipulatora.

Następnie wyniki obliczeń wysyłane są do interfejsu, gdzie zostają wyświetlane w postać przedstawionej na Rys 2.34.



Rysunek 2.34. Położenie wyświetlane w interfejsie.

2.5. Realizacja sterowania manipulatorem

2.5.1. Dobór procesora sterującego

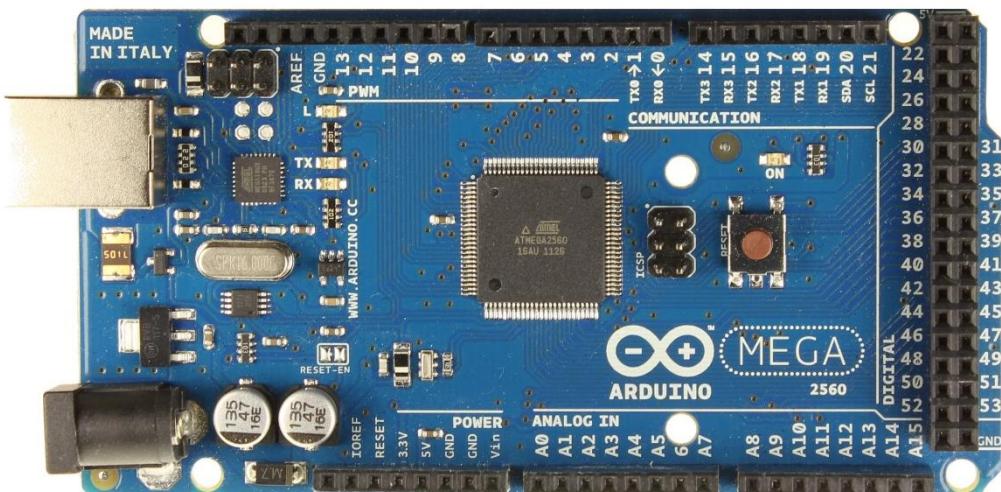
Dobór procesora sygnałowego do sterowania manipulatorem podyktywany był kilkoma podstawowymi kryteriami: prostotą interfejsu użytkownika, łatwością programowania, odpowiednią ilością wejść wyjść cyfrowych oraz możliwymi do zaimplementowania interfejsami komunikacyjnymi. Parametry te zestawiono w Tab. 2.11. Głównym priorytetem była przystępnośc uzywanego języka programowania tak, aby nawet niedoświadczeni w danym środowisku użytkownicy mogli obsługiwać i programować główną platformę.

Tabela 2.12. Wymagania wejść i wyjść procesora głównego.

Zastosowanie	Rodzaj sygnału	Ilość kanałów	Opis
Sterowanie impulsowe serwonapędami	Cyfrowy sygnał wyjściowy	7	Wykorzystywane kanały PWM
Komunikacja z interfejsem komputerowym	Interfejs komunikacyjny	1	Dowolny dostępny interfejs (UART, I2C lub inny interfejs komunikacyjny dostarczony przez producenta)
Komunikacja z STM32 obsługującym platformę	Interfejs komunikacyjny	1	UART

Biorąc pod uwagę powyższe wymagania wybrano trzy kontrolery spełniające podstawowe kryteria, z których następnie wybrano ten najbardziej odpowiadający potrzebom. Poniżej, na rysunkach 2.35, 2.36 oraz 2.37 pokazano wszystkie trzy analizowane kontrolery wraz z ich podstawową specyfikacją techniczną.

- **Arduino Mega**



Rysunek 2.35.Arduino Mega [22].

Tabela 2.13. Parametry kontrolera Arduino Mega.

Nazwa	Ardunio Mega
Zasilanie [V]	5 (standard USB)
Standardowe napięcie pracy [V]	5
Liczba wejść i wyjść cyfrowych	54
Liczba wejść i wyjść analogowych	16
Liczba wyjść PWM	15
Dostępne protokoły komunikacyjne	4 x UART

Platforma *Arduino*, przedstawiona na Rys. 2.35 była pierwszą platformą, którą wzięto pod uwagę. Duża łatwość programowania w uproszczonym języku *ANSI C* pozwala na bardzo szybkie zapoznanie się z platformą i łatwe rozpoczęcie tworzenia własnych aplikacji. Kolejnym atutem tego kontrolera jest duża ilość różnych konfiguracji wejść i wyjść. Niestety, *Arduino* nie posiada żadnego wbudowanego protokołu komunikacji bezprzewodowej, więc zachodziłaby konieczność dokupienia i konfiguracji komunikacji Bluetooth lub Wi-Fi. Kolejnym dużym problemem jest potrzeba zbudowania własnego interfejsu graficznego. Zadanie to nie jest trudne, ale aby skonfigurować ekran sterujący do swoich potrzeb wymaga bardziej zaawansowanych umiejętności programistycznych.

- ***STM32 Discovery***



Rysunek 2.36. *STM32F411 Discovery* [23].

Tabela 2.14. Parametry kontrolera *STM32F411 Discovery*.

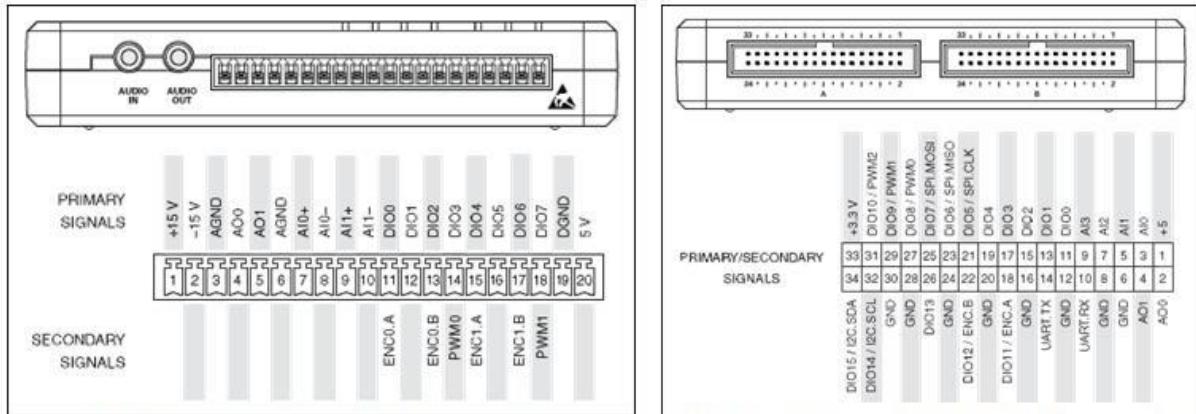
Nazwa	<i>STM32 F411 Discovery</i>
Zasilanie [V]	5 lub 3 (standard USB)
Standardowe napięcie pracy [V]	3.3
Liczba wejść i wyjść cyfrowych	100
Liczba wejść i wyjść analogowych	49 (dzielone z wyjściami cyfrowymi)
Liczba wyjść PWM	Każde wyjście cyfrowe może generować PWM
Dostępne protokoły komunikacyjne	3x USART, 5x SPI, 3x I2C

Kolejnym analizowanym kontrolerem jest *STM32 Discovery*. Płytką uruchomieniową *STM32 Discovery*, przedstawiona na Rys. 2.36, pod wieloma aspektami jest bardzo podobna do *Arduino*. Do programowania tego układu używa się języka *ANSI C* z biblioteką *CubeMX* [24], która bardzo ułatwia proces programowania. Nie jest ona aż tak intuicyjna jak środowisko *Arduino*, jednak nadal bardzo prosta w obsłudze. Wymaga wcześniejszej znajomości biblioteki, działania rejestrów oraz zegarów (*ang. timer*) procesora. Podobnie, jak zestaw *Arduino*, płytka *Discovery* również nie posiada wbudowanego systemu komunikacji bezprzewodowej, co wymusza użycie dodatkowego układu przesyłającego dane typu Bluetooth lub Wi-Fi. W przypadku tego układu zachodzi potrzeba zaprojektowania i zaprogramowania odpowiedniego interfejsu graficznego. Implikuje to takie same problemy jak w przypadku *Arduino*.

- **National Instrument myRIO**



Rysunek 2.37. NI myRIO [25].



Rysunek 2.38. Opis wejść i wyjść NI myRIO [25].

Tabela 2.15. Parametry kontrolera NI myRIO.

Nazwa	NI myRIO
Zasilanie [V]	12
Standardowe napięcie pracy [V]	3.3
Liczba wejść i wyjść cyfrowych	40
Liczba wejść i wyjść analogowych	Wejście – 10 Wyjście – 6
Liczba wyjść PWM	8
Dostępne protokoły komunikacyjne	2 x UART, 2x SPI, 2x I2C

Ostatnim z rozważanych rozwiązań jest płytka uruchomieniowa *NI myRIO*, stworzona w celach badawczo-edukacyjnych przez firmę *National Instruments* [25]. Cały układ został przedstawiony na Rys. 2.38 oraz 2.39. Programowanie układu odbywa się w środowisku *LabView*, przy pomocy graficznego języka programowania stworzonego przez *National Instruments*. Dzięki łatwości i przystępności graficznych interfejsów można zaprojektować ćwiczenia nawet dla osób, które nigdy wcześniej nie używały środowiska ani języka *LabView*. Kolejnymi zaletami procesora jest szeroka gama wejść i wyjść cyfrowych oraz analogowych, która zapewnia wystarczającą ilość kanałów komunikacyjnych i sygnałów PWM. Kontroler posiada bardzo łatwy w skonfigurowaniu i używaniu bezprzewodowy interfejs komunikacyjny, który może zapewnić odpowiednią komunikację między komputerem i platformą. Ostatnim ogromnym atutem działania kontrolera jest łatwość tworzenia

interfejsów graficznych w środowisku *LabView*. Funkcję tę opisano dokładniej w następnym rozdziale.

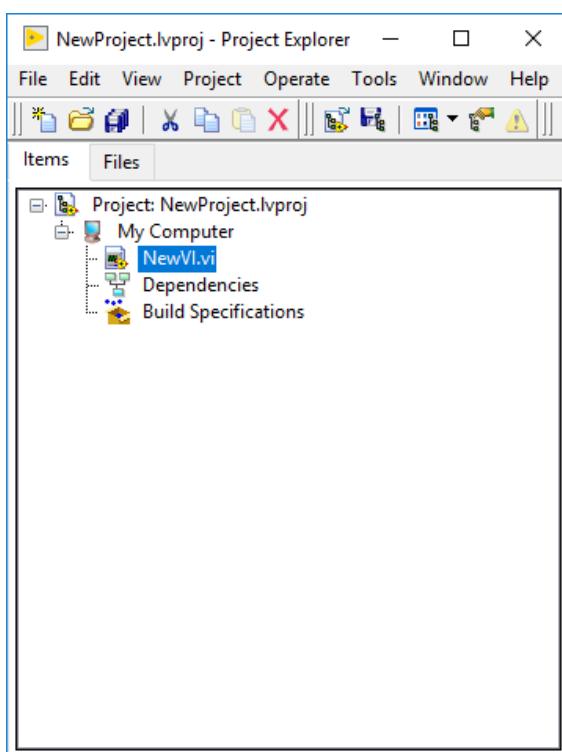
W porównaniu nie wspomniano o takich parametrach jak czasy taktowania procesorów czy rozmiary samych urządzeń. Charakter aplikacji, do której zostaną zastosowane, nie wymusza dużej mocy obliczeniowej i wszystkie zaproponowane urządzenia bez problemu poradzą sobie z powierzonymi zadaniami. Cały manipulator wraz z procesorem umieszczone zostaną na platformie, co powoduje całkowity brak ograniczeń w rozmiarach jednostki obliczeniowej.

Podsumowując, po porównaniu wszystkich wad i zalet, do sterowania manipulatorem wybrano platformę *myRIO* firmy *National Instruments* z powodu wszechstronności i możliwości programowania przy użyciu środowiska *LabView*.

2.5.2. Środowisko programowania

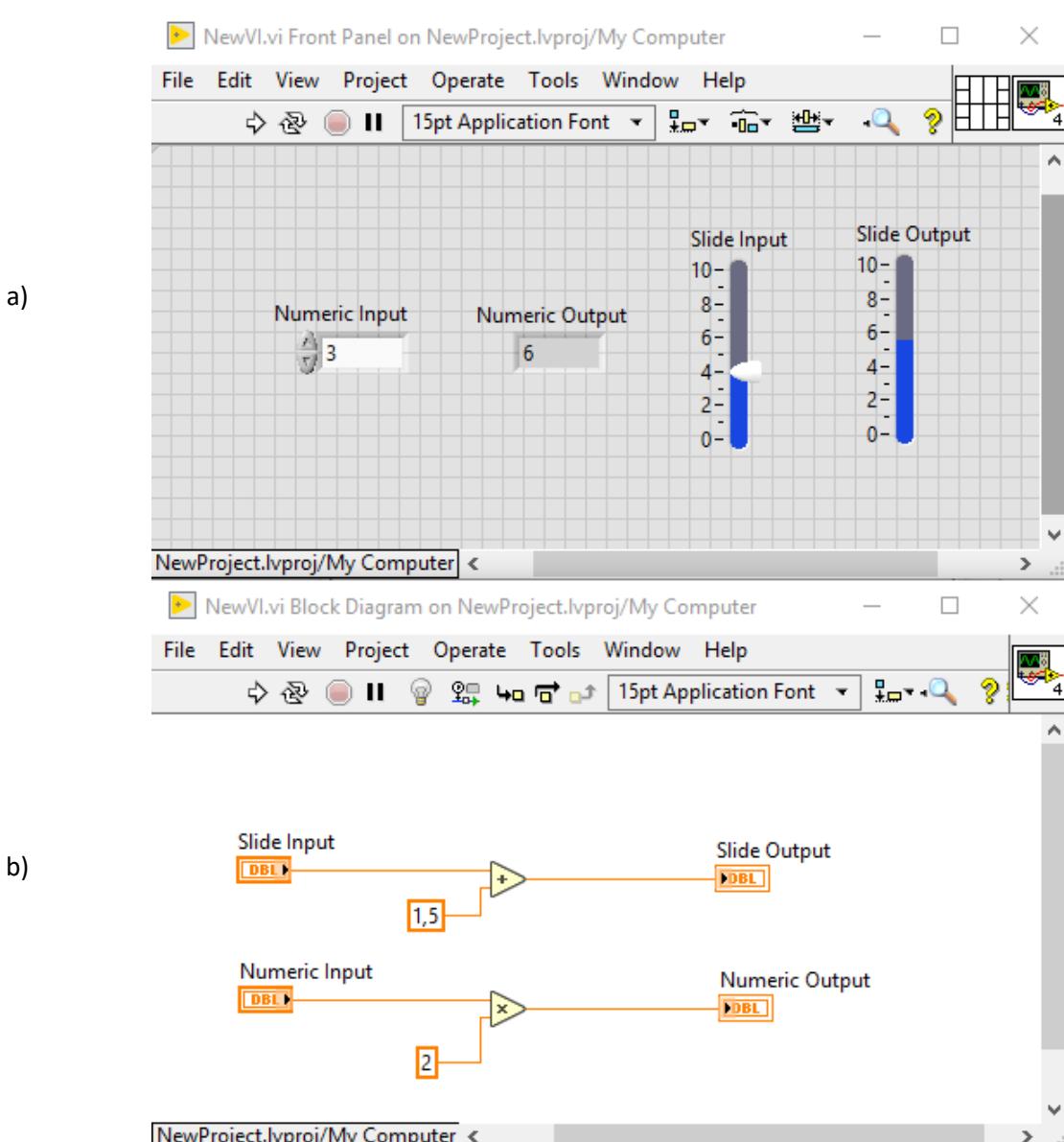
Wybrany kontroler manipulatora programowany jest przez graficzne środowisko *LabView*. W tym rozdziale przedstawiono zalety programowania w graficznym środowisku *LabView*. Opisano podstawowe elementy używane do programowania tak, aby ułatwić zrozumienie dalszych rozdziałów opisujących działanie programu sterującego platformą.

Po utworzeniu nowego projektu można, w ramach jednego drzewa struktur, łączyć nie tylko programy działające na jednym urządzeniu, ale również działające na zewnętrznych kontrolerach, tak jak w przypadku realizowanego projektu. Taka struktura pokazana została na Rys. 2.39.



Rysunek 2.39. Nowy projekt.

Tworzenie projektu odbywa się poprzez równoczesną pracę w oknie interfejsu oraz oknie programu logicznego. Okno interfejsu pozwala na projektowanie zróżnicowanych systemów zadawania i wyświetlania zmiennych, analizowanych przez program, którego struktura projektowana jest w oknie diagramów logicznych.



Rysunek 2.40. a) Okno interfejsu. b) Okno diagramu logicznego.

Na przedstawionych rysunkach można zaobserwować, że z poziomu interfejsu możliwe jest nie tylko odczytywanie wartości, ale również modyfikowanie zmiennych przechowywanych w pamięci mikroprocesora. Informacje, wprowadzone w odpowiednich miejscach w oknie górnym, mogą być później analizowane przez jednostkę logiczną i odpowiednio modyfikowane.

2.5.3. Dobór komunikacji komputer–procesor sterujący

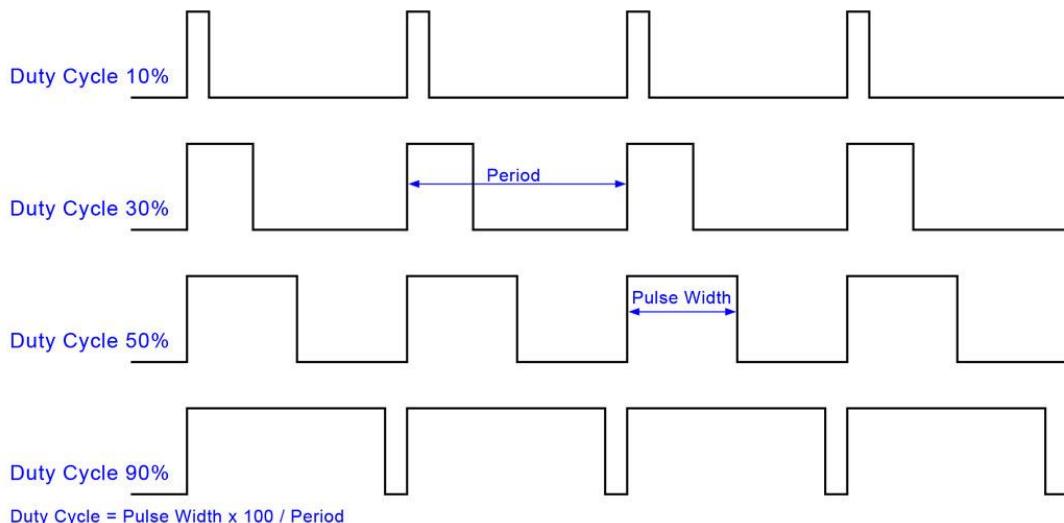
W trakcie wyboru odpowiedniego sposobu komunikacji bezprzewodowej analizie poddano drogę przenoszenia sygnału oraz typ magistrali danych. Pierwszym z napotkanych problemów był wybór sposobu przenoszenia informacji, tu analizowano trzy główne możliwości: drogą radiową (dedykowany nadajnik i odbiornik), Bluetooth oraz Wi-Fi. Następnie zdecydowano jakim rodzajem interfejsu komunikacyjnego będą przesyłane dane. Z uwagi na typ informacji oraz strukturę połączenia ze sobą elementów robota analizie poddano UART i I2C.

Po wybraniu odpowiedniego procesora, czyli *NI myRio* wybór był oczywisty, z uwagi na wbudowaną w procesor i środowisko programistyczne możliwość połączenia się bezprzewodowo procesora z komputerem. Komunikacja szeregowa UART zapewniona przez producenta następuje poprzez sieć bezprzewodową Wi-Fi. Sieć tworzona jest bezpośrednio przez odpowiedni podzespoł podłączony do mikroprocesora. Istnieje możliwość zabezpieczenia połączenia przy użyciu odpowiednich standardów bezpieczeństwa takich jak WPA2.

2.5.4. Opis programu sterującego manipulatorem w środowisku LabView

Program sterujący manipulatorem składa się z dwóch głównych części. Pierwsza służy do sterowania napędami manipulatora oraz do obsługi wszystkich blokad interfejsowych dla manipulatora. Druga część to, opisane wcześniej, wyznaczanie pozycji manipulatora na podstawie matematycznych obliczeń oraz odczytów zgięć poszczególnych przegubów. Z uwagi na wcześniejsze dokładne opisanie działania drugiej części w tym rozdziale opisano tylko aspekty sterowania manipulatorem.

Główna częścią strefy odpowiadającej za sterowanie manipulatorem jest obsługa odpowiednich wejść i wyjść, które wysyłają sygnał sterujący w postaci impulsów PWM. Generowanie sygnału PWM odbywa się przez zadanie kontrolerowi dwóch wartości. Pierwszą z nich jest częstotliwość sygnału, który ma być generowany, a drugą wypełnienie. Wypełnienie jest stosunkiem czasu, przez który sygnał ma nadaną logiczną wartość jeden, do czasu, przez który sygnał ma nadaną logiczną wartość zero. Przykład modulacji PWM przedstawiony jest na Rys. 2.41.



Rysunek 2.41. Sygnał PWM o różnym wypełnieniu [26].

Częstotliwość sygnału określana jest przy konfiguracji parametrów generatora PWM, natomiast, jego wypełnienie lub czas trwania impulsu należy podać w postaci wartości liczbowej na wejście bloku funkcyjnego zaimplementowanego w środowisku obsługującym procesor. Biorąc pod uwagę zadawanie wartości PWM w postaci stopni o jakie ma zgiąć się konkretny przegub, wymaganym było wykonanie odpowiedniego przeliczenia. Blok funkcyjny wymaga podania wartości wypełnienia w postaci ułamka dziesiętnego, o wartości z przedziału <0 ; 1>. Każdy z układów napędowych musi poruszać się między zgięciem zero stopni, a zgięciem sto osiemdziesiąt stopni. Z noty katalogowej odczytano, że serwonapęd współpracuje z częstotliwością 330 Hz, a zakres sterownia odbywa się między 0,8 do 2,2 ms czasu trwania pulsu. Serwomechanizm znajduje się w

pozycji neutralnej, kiedy czas trwania pulsu wynosi 1,5ms. W celu przekształcenia zakresu liczbowego <-90;90> na zakres <0,8;2,2> wykonane zostały kolejne przekształcenia:

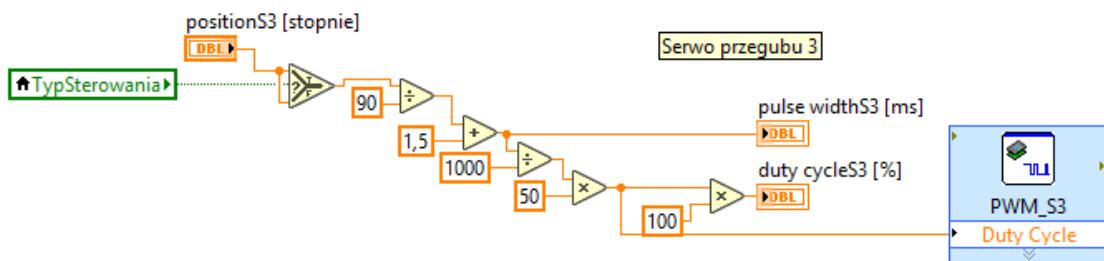
$$Duty\ cycle = \frac{\left(\frac{\alpha}{90^\circ} + 1,5\right)}{1000} \cdot 50, \quad (15)$$

gdzie:

Duty cycle – wypełnienie w postaci ułamka dziesiętnego;

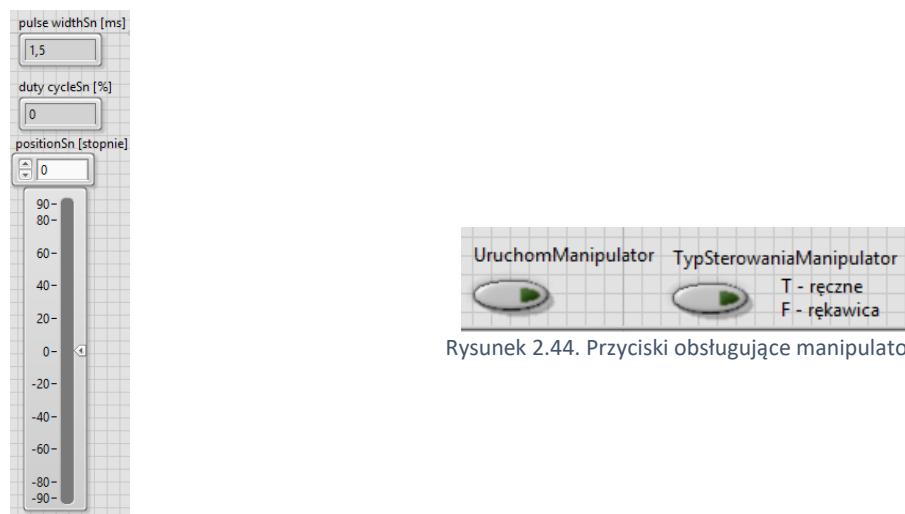
α – zadawany kąt w zakresie < -90; 90 >.

To przekształcenie zostało zaimplementowane w postaci blokowej, przedstawione na Rys. 2.42. Blok funkcyjny znajdujący się najbliżej po lewej na Rys. 2.42 jest blokiem funkcyjnym, który na podstawie zadanych wartości spowoduje odpowiednie ustalenie wyjść IO procesora.



Rysunek 2.42. Zaimplementowane przekształcenia zadawanego kąta na wypełnienie sygnału PWM.

Zabezpieczenia i odpowiednie kontrolki w środowisku *LabView* zaimplementowane są jako duże pętle warunkowej, które obejmują całość programu. Jedna z pętei odpowiada za włączanie manipulatora, czyli odblokowuje napędy z bezpiecznej pozycji oraz uruchamia przycisk wybór systemu kontroli. Przełącznik kontroli pozwala użytkownikowi sterowanie robotem przy pomocy kontrolera lub suwaków dostępnych w interfejsie graficznym, pokazanych na Rys. 2.43 i 2.44.



Rysunek 2.43. Suwaki pozwalające na sterowanie manipulatorem.

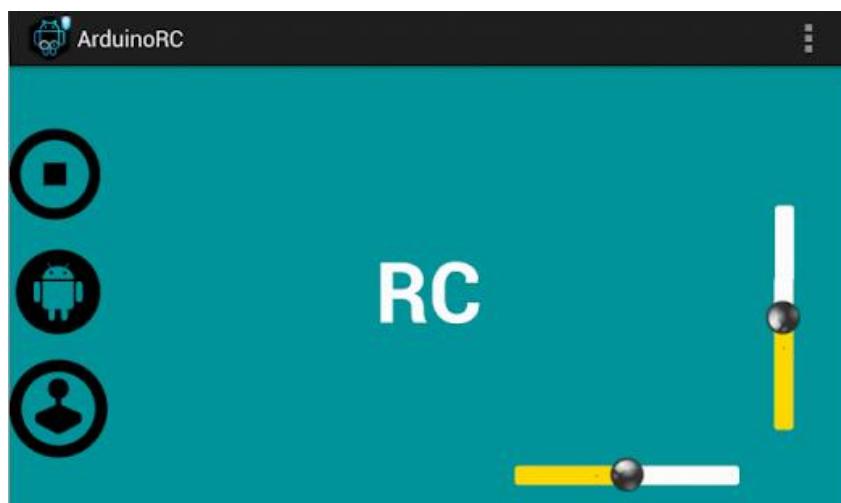
Rysunek 2.44. Przyciski obsługujące manipulator.

3. Sterowanie platformą i manipulatorem

3.1. Wprowadzenie

Od czasów pierwszych robotów sposoby sterowania i zadawania parametrów ruchu ewoluowały w sposób znaczący, od najprostszych przyciskowych pilotów do użycia wirtualnej rzeczywistości, która pozwala operatorowi na dosłowne „patrzenie oczyma robota”. Oczywiście, w zależności od potrzeb system sterowania może być odpowiednio uproszczony lub zaawansowany, pozwalając na wprowadzanie bardzo precyzyjnych zmiany położenia w czasie rzeczywistym.

Pośród konstrukcji prostych robotów edukacyjnych i hobbystycznych dominuje aktualnie najprostszy i najtańszy sposób sterowania, czyli emulowanie kontrolerów. Kontrolery symulowane, takie jak interfejsy Bluetooth, łączące się z komputerem bądź telefonem, pozwalają na bardzo szeroki zakres dopasowania potrzeb i przycisków czy joysticków interfejsu do założeń projektowych. Interfejs tego typu przedstawiony jest na Rys. 3.1. Niestety, emulowane przyciski i kontrolki nie dają użytkownikowi możliwości fizycznego wyczucia tego co dzieje się z robotem. Często pojawiającym się problemem w takich sposobach kontrolowania są opóźnienia w komunikacji wynikające z możliwości obliczeniowych urządzeń realizujących sterowanie i ograniczeń zastosowanych protokołów komunikacyjnych. Niekontrolowane i zbyt duże opóźnienia (*ang. lag*) w trakcie pracy mogą prowadzić do uszkodzenia robota, jeżeli operator w niekorzystnym momencie utraci połączenie, a system nie jest odpowiednio zabezpieczony. Z uwagi na założone przeznaczenie robota, wymagane jest całkowite bezpieczeństwo, dlatego wykluczono realizację sterowania przez dodatkowy interfejs.



Rysunek 3.1. Emulowany pad Android [27].

Kolejnym często używanym typem kontrolera, także początkowo rozważanym przez autora pracy, były kontrolery *RC* oraz kontrolery używane do obsługi konsol multimedialnych takich jak *Xbox* czy *PS3*. Są to produkty, których głównymi założeniami projektowymi są: bardzo krótki czas odpowiedzi na polecenia oraz duża wytrzymałość. Gotowe programy komputerowe lub odpowiednie interfejsy komunikacyjne, zapewniane przez producenta, pozwalają na łatwą implementację w projekcie oraz dużą dowolność w zaprogramowaniu funkcji poszczególnych przycisków. Niestety, po wykonaniu badania rynku oraz dostępnych rozwiązań, okazało się, że żaden z dostępnych kontrolerów konsolowych nie może współpracować bezpośrednio ze środowiskiem *LabView*. Realizacja projektu została skomplikowana o konieczność zakupienia lub stworzenia odpowiedniego programu dekodującego odczyty z joysticka. Z kolei komercyjne kontrolery *RC* nie

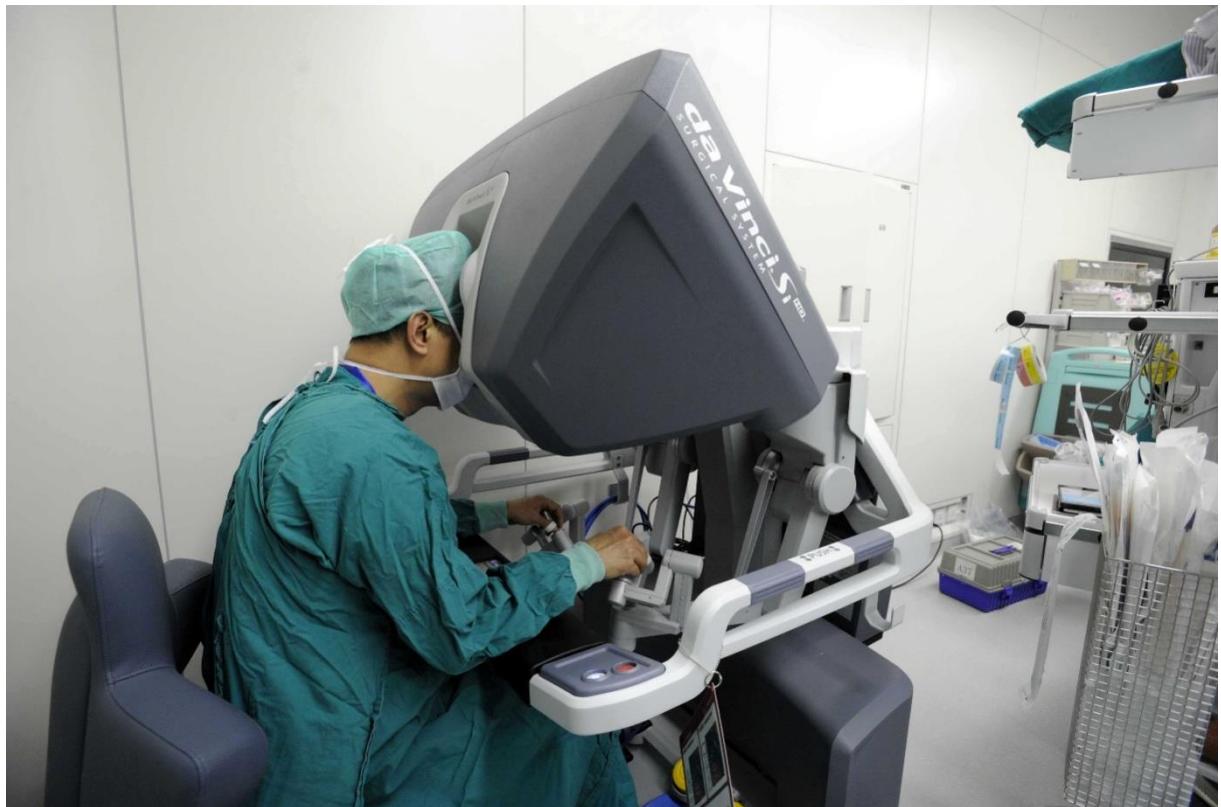
posiadają najczęściej możliwości programowania. Kupuje się je w gotowym zestawie zawierającym również sterownik, który może obsługiwać dowolne rodzaje silników czy serwomechanizmów. Z tego powodu proste kontrolery *RC* nie nadają się do zastosowania w realizowanym projekcie. Sterownik manipulatora musi również wykonywać równocześnie wiele innych czynności oprócz sterowania samymi napędami. Pomimo obiecujących parametrów i doskonałych teoretycznych możliwości, pomysł musiał zostać odrzucony na rzecz innego rozwiązania. Przykładowe kontrolery, rozważane do zastosowania w realizowanym projekcie, przedstawiono na Rys. 3.2.



Rysunek 3.2. Kolejno: Kontroler *RC*, Kontroler *PS3* oraz Kontroler *Xbox360* [28] [29] [30].

Dzięki bardzo szybkiemu rozwojowi technik cyfrowych, takich jak kontrolery wirtualnej rzeczywistości i różnego rodzaju kontrolery ruchowe, bardzo atrakcyjnym sposobem sterowania jest sterowanie poprzez bezpośrednie ruchy operatora. Używając gogli rozszerzonej rzeczywistości można wprowadzić operatora dokładnie w miejsce pracy robota oraz dzięki odpowiedniemu przeskalowaniu wykonywanych przez operatora czynności ułatwić mu precyzyjne prace. Idealnym przykładem takiego sposobu sterowania jest medyczna rewolucja w postaci robota *daVinci*, który pozwala na wykonywanie bardzo skomplikowanych operacji, całkowicie przy tym eliminując braki w dokładności człowieka. Lekarz operujący znajduje się niedaleko pacjenta, jednak to nie on wykonuje wszystkie czynności. Dzięki powiększającemu wizjerowi i zestawowi kontrolerów lekarz może wykonywać duże ruchy, które następnie są przekształcane przez robota w ruchy o dokładności do ułamków milimetra. Cały układ sterujący robota *daVinci* pokazany jest na Rys. 3.3 [31]. Niestety, jak łatwo można się domyślić, takie rozwiązanie jest niezwykle drogie i chociażby odtworzenie podobnego narzędzia byłoby bardzo trudne. Jednak może ono posłużyć jako inspiracja i nadać kierunek dla projektu własnego kontrolera. Innym przykładem takiego rozwiązania może być środowisko *ABB RobotStudio*.

współpracujące z goglami rozszerzonej rzeczywistości, pozwalające na poruszanie wirtualnym robotem poprzez rzeczywiste ruchy operatora [32].



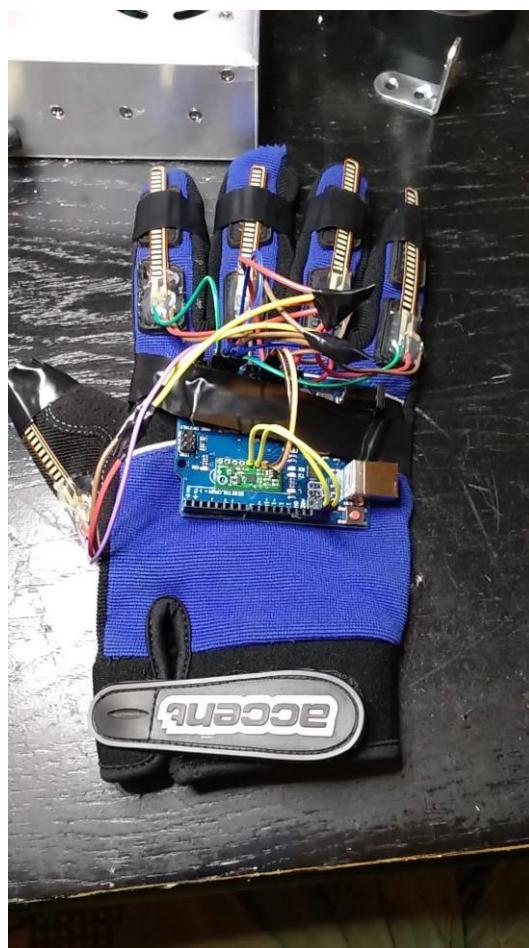
Rysunek 3.3. Panel sterujący robota *DaVinci* [33].

Przedstawione powyżej rozwiązania, które mają swoje wady i zalety, mogą zostać odpowiednio połączone w jedno funkcjonalne i minimalizujące problemy rozwiązanie końcowe. Inspiracją do stworzenia własnego rozwiązania były dwa znane wcześniej autorowi pomysły, pierwszym był robot *daVinci*, a drugim interaktywna rękawica wyprodukowana przez firmę Nintendo. *Power Glove* (Rys. 3.4) od Nintendo [34] miała służyć do alternatywnego sposobu sterowania grami komputerowymi poprzez odpowiednie poruszanie dlonią. Rękawica była bardzo zaawansowana, jak na czasy swojego powstania, ponieważ używała czujników akcelerometrycznych i żyroskopów do wykrywania wychyleń oraz przemieszczeń dłoni operatora w przestrzeni.

Konstrukcja i zasada działania *Power Glove* oraz chęć większej interakcji podczas pracy z robotem były inspiracją do opracowania autorskiego modelu rękawicy interaktywnej, skonstruowanej bezpośrednio na potrzeby projektu. Dokładną konstrukcję i założenia projektowe opisano w następnym rozdziale. Skonstruowana rękawica została przedstawiona na Rys. 3.5.



Rysunek 3.4. *Nintendo Power Glove* [34].



Rysunek 3.5. Rękawica interaktywna - sterownik platformy.

3.2. Kontrolera do sterowania platformą i manipulatorem

3.2.1. Założenia projektowe kontrolera

Dostosowanie kontrolera do wydajnego i intuicyjnego sterowania platformą było zadaniem złożonym. Najważniejszym z zagadnień było dostosowanie gestów, tak aby były intuicyjne i łatwe do zapamiętania dla operatora, będąc równocześnie łatwo rozpoznawalnymi dla systemu.

Oczywistym było, że rękawica będzie wydawać polecenia poprzez odpowiednie wychylenie ręki odczytywane przez akcelerometr zgodne z osiami X i Y w układzie kartezjańskim, którego środek będzie znajdował się na wierzchu dłoni operatora, czyli tam, gdzie umieszczony jest akcelerometr.

Po ustaleniu w jaki sposób ma działać przesuwanie zarówno kół platformy jak i manipulatora należało rozgraniczyć względem siebie te dwie operacje. Manipulator nie powinien poruszać się równocześnie z przemieszczaniem się platformy. Dlatego właśnie powstała potrzeba wybierania za pomocą odpowiednich gestów elementów, które będą przemieszczane. Pierwszym nasuwającym się rozwiązaniem było użycie zgięć palców, tak aby odpowiednio zaznaczać polecenia. Dzięki umieszczeniu czujnika zgięcia na każdym z pięciu palców dłoni uzyskano możliwość zakodowania pięciu cyfr binarnych. W ten sposób powstał kod np. zgięcie kciuka powodowało wysłanie do procesora liczby „10000” co dawało sygnał, że należy wykonywać teraz ruchy pierwszym przegubem. Wszystkie możliwe kombinacje opisano w oddzielnym rozdziale.

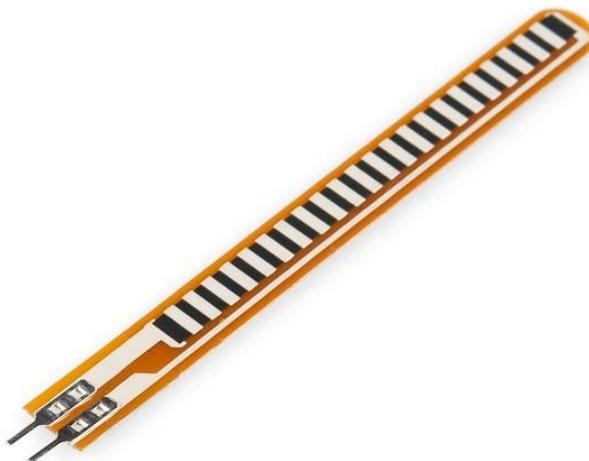
Założenia konstrukcyjne całej rękawicy były relatywnie proste. Na wierzchu dłoni miał znajdować się kontroler, do którego przymocowana została płytka z akcelerometrem. Do każdego palca zamontowane zostały małe czujniki ruchu, którymi umożliwiono swobodne przesuwanie się, gdy zginany jest palec operatora, tak aby nie rozerwać czujnika ani nie wyrwać przewodów sygnalowych.

3.2.2. Dobór czujników tensometrycznych

Tensometrami nazywamy rodzinę czujników, która pozwala zmierzyć odkształcenia materiału. Tensometr mierzy kształt odkształcenia, czyli stopień wygięcia, a następnie na podstawie prawa Hooka można obliczyć wartość siły naprężenia [35]. Czujników tych można używać do pomiarów wielu rodzajów wielkości takich jak: ciśnienie, siłę, masę lub stopień odkształcenia, czyli innymi słowy ugięcie.

W aplikacji rękawicy interaktywnej działanie czujnika można w sposób uproszczony porównać do rezystora o zmiennej rezystancji, która zależy od ugięcia. Większość tensometrów dostępnych na rynku sprzedawanych jest w postaci zalaminowanej folii z metalowymi wyprowadzeniami służącymi do podłączenia czujnika. Użyty w aplikacji procesor poprzez wejście analogowe mierzy wartość rezystancji tensometru, poprzez dzielik napięcia z podłączonym napięciem 5 V.

Wybrany przez autora tensometr, pokazany na Rys. 3.6, jest dokładnie takim samym modelem jakiego używa firma Nintendo do wyprodukowania swojej rękawicy *Power Glove* [34]. Poniżej, w Tab. 3.1 znajdują się dane techniczne użytego modelu tensometru.



Rysunek 3.6. Czujnik ugięcia *SparkFun* [36].

Tabela 3.1. Parametry Czujnika ugięcia *SparkFun*.

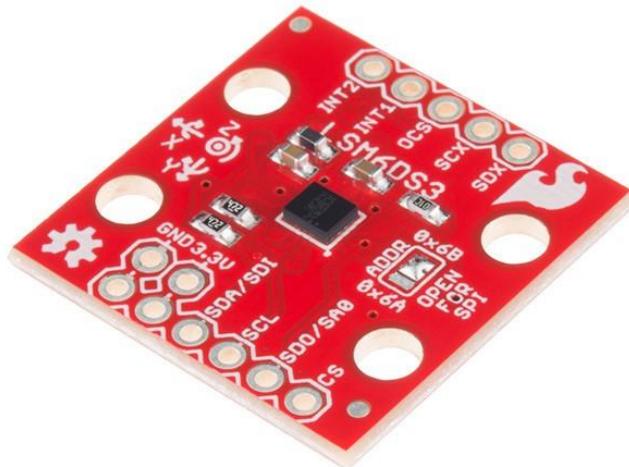
Nazwa	Czujnik ugięcia <i>SparkFun</i>
Rozmiary [mm]	73 X 6,3
Rezystancja bez ugięcia (0°) [$k\Omega$]	22
Rezystancja maksymalne ugięcie (180°) [$k\Omega$]	42
Zalecany rezistor w dzielniku [$k\Omega$]	22

3.2.3. Dobór czujnika akcelerometrycznego oraz żyroskopu

Dobór układu akcelerometrycznego został wykonany w dwóch krokach. Pierwszym był wybór między układem analogowym, a cyfrowym. Drugim krokiem był wybór konkretnego modelu układu scalonego bądź gotowej płytki zawierającej kilka układów obsługiwanych przez jeden interfejs.

Wybór układu ze względu na sposób przekazywania danych do kontrolera zależał w przypadku aktualnie opisywanego projektu wyłącznie od ilości dostępnych wejść przetwornika analogowego dostępnego w procesorze pomiarowym. Jak opisano w poprzednim rozdziale, w projekcie rękawicy sterującej manipulatorem użyto pięciu wejść analogowych. Większość układów zawierających równocześnie akcelerometr oraz żyroskop wymagają dodatkowych sześciu wejść analogowych. Dlatego, pomimo dużo łatwiejszej obsługi wejść i wyjść analogowych w porównaniu z dowolną komunikacją typu szeregowego, ograniczenia projektowe większości dostępnych procesorów spowodowały, że użyto interfejsu szeregowego. Większość gotowych rozwiązań tego typu ma wbudowane równocześnie dwa interfejsy komunikacyjne I2C oraz SPI. Z uwagi na niewielkie rozmiary projektu oraz małą ilość podłączonych do procesora układów peryferyjnych, wybrany został prostszy sposób komunikacji, czyli interfejs I2C.

Wybór układów akcelerometrycznych odbywał się między dwoma modelami. Pierwszym z układów był model *LSM6DS3* od firmy *SparkFun* [37]. Jest to gotowy układ zawierający trzyosiowy akcelerometr oraz żyroskop, który może komunikować się z procesorem zarówno po SPI jak i I2C. Układ został przedstawiony na Rys. 3.7, a jego parametry w Tab. 3.2.

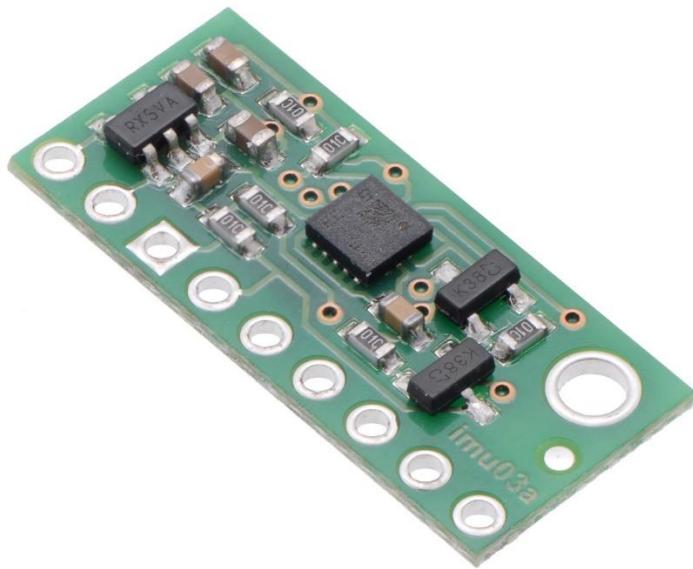


Rysunek 3.7. *SparkFun LSM6DS3*[34].

Tabela 3.2 Parametry akcelerometru *LSM6DS3*.

Nazwa	<i>SparkFun LSM6DS3</i>
Dostępne układy pomiarowe	Akcelerometr, żyroskop, termometr
Napięcie zasilania [V]	1,8 – 3,6
Pobór prądu [mA]	Ok. 1
Osie	X, Y, Z
Dostępne interfejsy komunikacyjne	I2C, SPI
Napięcie sygnału komunikacji [V]	3,3
Rozdzielcość pomiarowa	16 bit

Układ, o parametrach bardzo dobrze dopasowanych do celów projektu, został odrzucony z powodu napięcia stanu wysokiego na linii komunikacyjnej, które wynosi, jak podano powyżej w Tab. 3.2 3,3 V, podczas gdy wybrany do sterowania rękawicą mikrokontroler (przedstawiony w kolejnym rozdziale) posiada wejście na linii sygnałowej 5 V. Uznano, że stosowanie dodatkowego konwertera napięć zwiększy komplikację i potencjalną awaryjność opracowanej rękawicy. Dlatego zdecydowano się na użycie, podobnego do powyższego, układu od firmy Pololu *LSM6DS33* [38]. Wszystkie funkcjonalności układu potrzebne do projektu zostały zachowane. W modelu tym zostało zmienione jedynie napięcie stanu wysokiego na magistrali komunikacyjnej, będące napięciem zasilania układu. Wyżej opisany układ przedstawiony jest na Rys. 3.8, a jego parametry opisane są w Tab. 3.3.



Rysunek 3.8. Pololu LSM6DS33 [38].

Tabela 3.3. Parametry akcelerometru LSM6DS33.

Nazwa	<i>Pololu LSM6DS33</i>
Dostępne układy pomiarowe	Akcelerometr, żydroskop
Napięcie zasilania [V]	2,5 – 5,5
Pobór prądu [mA]	Ok. 2
Osie	X, Y, Z
Dostępne interfejsy komunikacyjne	I2C, SPI
Napięcie sygnału komunikacji [V]	Napięcie zasilania
Rozdzielcość pomiarowa	16 bit

Wybrany układ ma parametry idealnie dopasowane do zastosowania w realizowanym projekcie rękawicy interaktywnej. Posiada dużą rozdzielcość pomiarową, dzięki czemu można dokładnie odczytywać zmienne oraz sterować układami robota. Producent dostarcza również odpowiednie biblioteki kompatybilne z kontrolerami *Discovery STM* oraz *Arduino*. Oprogramowanie takie znacznie ułatwiło wykonanie końcowego programu sterującego.

3.3. Oprogramowanie kontrolera

3.3.1. Dobór procesora sterującego dla kontrolera

W przypadku doboru układu sterującego rękawicą, w przeciwieństwie do układu sterującego platformą mobilną i umieszczonym na niej manipulatorem, należało uwzględnić nie tylko odpowiednią ilość wejść i wyjść, ale również rozmiar i masę układu. Układ sterujący wraz z czujnikiem akcelerometrycznym powinien zmieścić się na wierzchu przeciętnej dłoni. Dla dobrego dopasowania zostały zmierzone dlonie trzech osób i przyjęto uśrednione wymiary płytki z procesorem wynoszące ok. 60 x 45 mm.

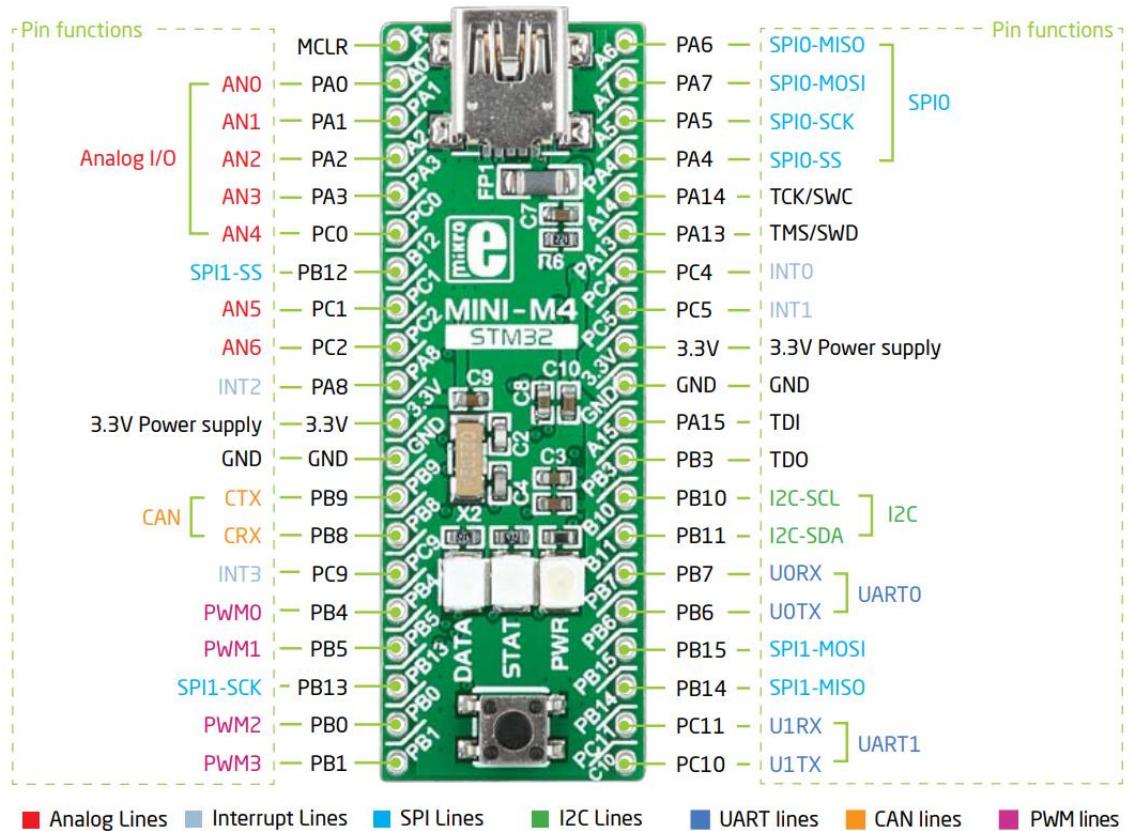
Jako pozostałe kryteria wyboru zostały w pierwszej kolejności uwzględnione potrzebne niestandardowe zestawy wejść i wyjść. W przeciwieństwie do doboru procesora głównego, ten sterujący rękawicą nie musi charakteryzować się prostym językiem programowania. Po zaprogramowaniu jednostki, kod wewnętrz rękaawicy nie będzie już zmieniany przez użytkowników.

Tabela 3.4. Wymagania wejść i wyjść procesora rękawicy.

Zastosowanie	Rodzaj sygnału	Ilość kanałów	Opis
Odbieranie sygnałów z czujników zgięć	Wejścia analogowe	5	Wykorzystywane wejścia do przetwornika analogowo-cyfrowego
Komunikacja z akcelerometrem	Interfejs komunikacyjny	1	I2C, SPI
Komunikacja z interfejsem komputerowym	Interfejs komunikacyjny	1	Dowolny dostępny interfejs (UART, I2C lub gotowa komunikacja dostarczona przez producenta)

Powyższe wymagania spełnia wiele dostępnych na rynku mikrokontrolerów. Poniżej przedstawiono wybrane trzy jako najbardziej spełniające wymogi projektu. Każdy z nich opisano i wymieniono jego wady i zalety w kontekście aplikacji w projekcie rękawicy.

Pierwszym z opisywanych kontrolerów jest płytka uruchomieniowa używająca mikrokontrolera z rodziny STM32 pokazana na Rys. 3.9, wszystkie jej najważniejsze parametry zostały przedstawione w Tab. 3.5. Biorąc pod uwagę jej rozmiary, jest ona idealnym kandydatem do zastosowania w projekcie. Ma wystarczającą ilość interfejsów komunikacyjnych oraz wejść analogowych. Niestety w porównaniu z pozostałymi procesorami, przedstawionymi dalej w tym rozdziale, biblioteka *CubeMX* [24] nie dostarcza prostych i łatwych w obsłudze narzędzi programowych do zarządzania urządzeniami pomiarowymi współpracującymi z magistralą I2C lub SPI. Z powodu trudniejszej konfiguracji interfejsów komunikacyjnych, zarówno przy komunikacji z komputerem jak i akcelerometrem, powyższy mikrokontroler nie został wybrany do realizacji projektu.

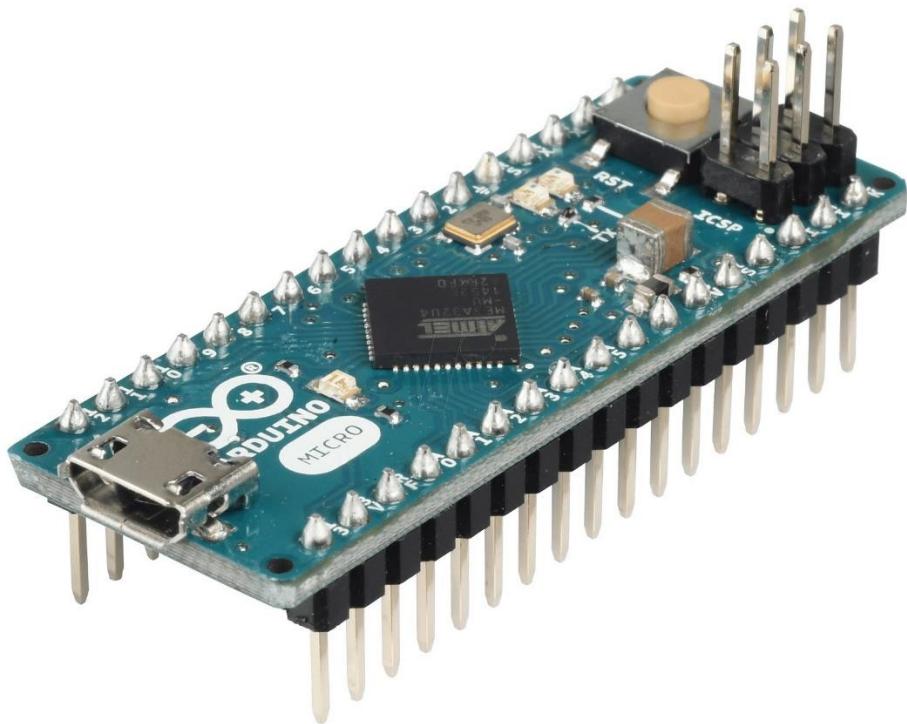


Rysunek 3.9. MINI-M4 FOR STM32 [39].

Tabela 3.5. Parametry kontrolera MINI-M4 FOR STM32.

Nazwa	Mini-M4 for STM32
Zasilanie [V]	3,3 lub 5
Rozmiary [mm]	50,8 X 17.78
Standardowe napięcie pracy [V]	3.3
Liczba wejść i wyjść cyfrowych	28
Liczba wejść i wyjść analogowych	6
Liczba wyjść PWM	4
Dostępne protokoły komunikacyjne	2 x UART, 2x SPI, 1x I2C, USB szeregowy

Drugim z opisywanych kontrolerów jest płytka uruchomieniowa *Arduino Micro*, pokazana na Rys. 3.11, oparta o układ obliczeniowy *Atmega* oraz interfejs programistyczny *Arduino*. Tab. 3.6 zawiera parametry procesora, na jej podstawie można stwierdzić, że wszystkie wymagania systemowe są spełnione i powyższy układ spełnia wszystkie założenia projektowe. W porównaniu z poprzednim układem znacznie ułatwiona jest realizacja komunikacji z komputerem, bez konieczności tworzenia własnych, złożonych bibliotek. Niestety, dedykowana biblioteka, która umożliwia prostą realizację komunikacji z wybranym akcelerometrem, dostępna jest tylko dla większej jednostki z tej rodziny, modelu *Arduino Uno*.



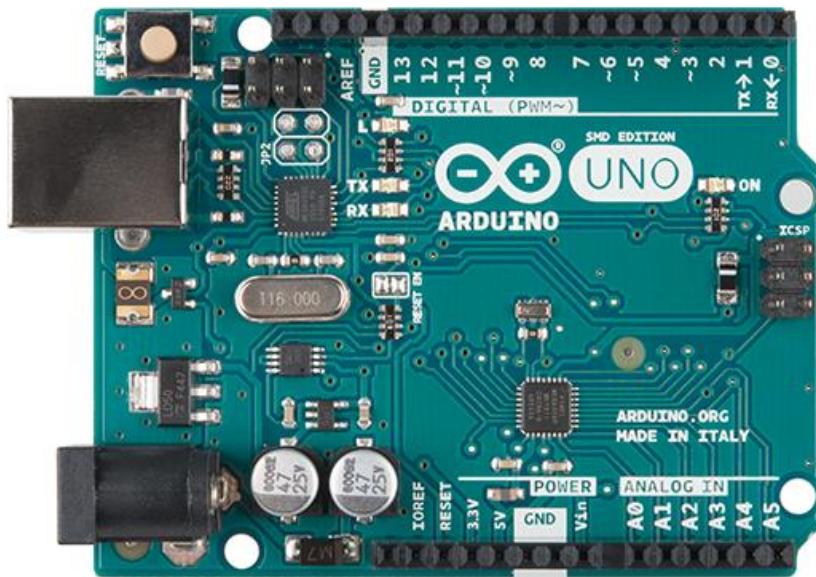
Rysunek 3.10. *Arduino Micro* [40].

Tabela 3.6. Parametry kontrolera *Arduino Micro*.

Nazwa	<i>Arduino Micro</i>
Zasilanie [V]	7 - 12
Rozmiary [mm]	48 X 18
Standardowe napięcie pracy [V]	5
Liczba wejść i wyjść cyfrowych	12
Liczba wejść i wyjść analogowych	12
Liczba wyjść PWM	7
Dostępne protokoły komunikacyjne	UART, SPI, I2C, USB szeregowy

Ostatnim z analizowanych procesorów jest układ również oparty o procesor Atmega, pochodzący z tej samej rodziny, mikrokontroler *Arduino Uno*. Posiada on wystarczająco małe rozmiary, pomimo, że przekraczają one przyjęte założenia. Wstępne testy wykazały, że mimo większych rozmiarów nie ogranicza on ruchów ręki operatora w rękawicy. Posiada także wystarczającą ilość wejść i wyjść oraz interfejsów komunikacyjnych. Procesor pozwala również na użycie biblioteki obsługującej magistralę I2C. Dzięki wbudowanej bibliotece, dostarczonej przez producenta akcelerometru, procedura uruchamiania i utrzymania połączenia z czujnikiem jest ułatwiona.

Po wykonanej analizie, zdecydowano się na zastosowanie kontrolera *Arduino Uno*. Głównym czynnikiem, który wpłynął na wybór tego układu, była możliwość użycia gotowej biblioteki służącej do obsługi akcelerometru, co w przypadku pozostałych jednostek obliczeniowych było niemożliwe lub bardzo utrudnione. Równocześnie użycie kontrolera *Arduino*, który ma wystarczającą moc obliczeniową, umożliwia łatwiejsze dokonywanie zmian w przypadku rozwoju projektu. Opisany układ przedstawiony jest na Rys. 3.11, a parametry w Tab. 3.7.



Rysunek 3.11. Arduino UNO ref 3 [41].

Tabela 3.7. Parametry kontrolera Arduino UNO ref 3.

Nazwa	Arduino UNO ref 3
Zasilanie [V]	7 - 12
Rozmiary [mm]	69 X 53,5
Standardowe napięcie pracy [V]	5
Liczba wejść i wyjść cyfrowych	14
Liczba wejść i wyjść analogowych	6
Liczba wyjść PWM	6
Dostępne protokoły komunikacyjne	UART, SPI, I2C, USB szeregowy

3.3.2. Opis interfejsu komunikacyjnego z komputerem

Dylemat dotyczący typu używanego interfejsu komunikacyjnego został rozwiązany przez użycie gotowych funkcji wbudowanych w systemy kontrolne. Zastosowano komunikację szeregową USB. Arduino posiada bardzo łatwy do oprogramowania interfejs wystawiania dowolnych informacji na port USB, do którego aktualnie jest podłączone. Program *LabView* umożliwia równocześnie odczytanie całych ramek danych i analizę ich poszczególnych składowych. Rozbite dane z ramek są następnie wysyłane do odpowiednich części programu w celu sterowania ruchami lub blokadami platformy.

Główny z problemów, który stanął przed autorem, jest równocześnie połączony z problemem optymalnego sterowania platformą przy użyciu jak najmniejszej ilości danych wejściowych. Znaczy to, że należy uniknąć osobnych ciągów danych dla platformy i manipulatora przychodzących z rękawicy. Zoptymalizowana ramka danych została przedstawiona na Rys. 3.12.



Rysunek 3.12 Graficzna reprezentacja wysyłanej ramki danych

Jak można zaobserwować na powyższym rysunku, jako ramkę danych wysyłany jest zestaw dwóch wartości odczytanych z akcelerometru oraz pięciu wartości zero-jedynkowych reprezentujących zgięcie poszczególnych palców, gdzie zero to wyprostowany palec, a jeden to palec zgięty. Na podstawie tych danych można poprzez zginanie i prostowanie palców w rękawicy wydawać polecenia odpowiednim elementom urządzenia. Skręt dłoni w płaszczyźnie X lub Y działa na potrzeby opracowanej aplikacji jak zwykły drążek analogowy.

3.3.3. Program sterujący rękawicą Arduino

Program napisany na platformę *Arduino*, która zbiera informacje z czujników umieszczonych na rękawicy, zawiera trzy główne części. Pierwszą z części jest odczyt z czujników ugięcia i analiza otrzymanych wyników w taki sposób, aby do interfejsu komunikacyjnego wysyłać już wartości logiczne zero lub jeden. Druga część polega na odczytaniu danych z układu *LSM6* i przygotowaniu ich, w jak najbardziej zoptymalizowany sposób, do wysłania na szeregowy port USB z resztą danych w postaci ramki. Trzecia i ostatnia część programu polega na cyklicznym wywołaniu odpowiednich partii programu, oraz na konstrukcji ramki danych, która zawiera wszystkie odczytane wcześniej wartości.

Pierwsza część programu składa się z zainicjowania odpowiedniego wejścia kontrolera jako właściwego wejścia analogowego. Następnie funkcja odczytująca odpowiednie wartości z wejścia przekształca je przy każdorazowym odczycie w taki sposób, aby komputer mógł odczytać je już bez żadnych późniejszych zmian. Kolejnym krokiem jest cykliczne wysyłanie danych do bufora konstruującego ramkę. Poniżej znajduje się przykładowy proces odczytu wartości z czujnika ugięcia pokazany na przykładzie pojedynczego palca, w tym konkretnym wypadku małego (*ang. Pinki*). Kod przedstawiony jest na Rys. 3.13.

```
// zdefiniowanie wejść analogowych
static int flexSensorPin_1 = A0;
...
void readFlexSensors(){
    // inicjalizacja wejść analogowych jako czujnika ugięcia
    int flexSensorReadingsPinki = analogRead(flexSensorPin_1);
    ...
    // uruchmienie zmiennych lokalnych
    // które reprezentują stan logiczny czujników ugięcia
    bool flexSensorReadingsPinkiLogic;
    ...
    // przetworzenie stanu analogowego czujnika na stan logiczny 1/0
    if(flexSensorReadingsPinki < 390){
        flexSensorReadingsPinkiLogic = 1;
    } else{
        flexSensorReadingsPinkiLogic = 0;
    }
    ...

    // podanie odczytanych stanów na odpowiednie miejsca
    // w ramce danych przesyłanych do komputera
    Serial.print(flexSensorReadingsPinkiLogic);
    Serial.print(",");
}
// zakończenie funkcji odczytu wartości
```

Rysunek 3.13. Kod programu obsługującego czujniki ugięcia.

Kolejna część programu, która ma odpowiadać za odczytywanie wartości z urządzenia *LSM6*, czyli akcelerometru i żyroskopu. Gdyby nie kompatybilna z *Arduino UNO* biblioteka ***LSM6.h*** [42], dostarczona przez producenta, wykonanie takiego programu mogłoby przysporzyć sporo problemów oraz zająć dużo czasu. Biblioteka wykonuje wszystkie czynności uruchomienia i zakończenia komunikacji z akcelerometrem automatycznie bez ingerencji użytkownika. Zanim zimportowana zostanie biblioteka ***LSM6.h***, należy wcześniej dołączyć procedury odpowiedzialne za komunikację po magistrali I2C. Wszystkie te procedury znajdują się w pakiecie ***Wire.h*** [43]. Po zimportowaniu wszystkich danych należy nadać urządzeniu, z którym nawiązana zostanie komunikacja, nazwę oraz zainicjować dwie zmienne wewnętrzne, które będą przechowywać wysłane po magistrali zmienne zawierające potrzebne informacje. Poniżej pokazano uproszczone działanie programu na bazie oprogramowania jednej z osi, na potrzeby przykładu użyta zostanie oś X. Kod przedstawiony jest na Rys. 3.14.

```
// biblioteka I2C
#include<Wire.h>
//biblioteka żyroskop i akcelerometr
#include<LSM6.h>
...
// zdefiniowanie akcelerometru
LSM6 imu;
char reportAcc[80];
char reportGyro[80];
...
void setup(){
    ...
    // otwarcie komunikacji I2C
    Wire.begin();
    // jeśli nie znajdziemy urządzenia LSM6 to wysyłamy komunikat
    if (!imu.init()){
        Serial.println("Failed to detect and initialize IMU!");
        while (1);
    }
    imu.enableDefault();
}
...
// odczyt danych z akcelerometru przez magistralę I2C
void readAccel(){
    // uruchomienie odczytu z urządzenia LSM6
    imu.read();
    // odczytanie odpowiednich zmiennych z zadanego urządzenia
    int pozycjaX = map(imu.a.x, -16900, 16900, -99, 99);
    ...
    // podanie zmiennych do ramki danych
    Serial.print(pozycjaX);
    // zakończenie funkcji odczytu
}
```

Rysunek 3.14. Kod programu obsługującego akcelerometr I2C.

Ostatnia, trzecia część, to cykliczne wywołanie obu przedstawionych funkcji odczytu danych z czujników. W tym zestawie komend zawiera się również uruchomienie i konfiguracja komunikacji szeregowej USB oraz wywołanie funkcji odczytujących, które już w sobie mają zawarte odpowiednie instrukcje tworzące ramki danych. Kod przedstawiony jest na Rys. 3.15.

```

void setup(){
    // otwarcie komunikacji szeregowej USB
    Serial.begin(9600);
    ...
}

void loop(){
    // uruchomienie funkcji odczytu z czujników
    // w powtarzającej się pętli
    readFlexSensors();
    readAcce();
    // wysłanie danych do komputera
    Serial.println();
    // oczekanie przed kolejnym wysłaniem
    delay(100);
}

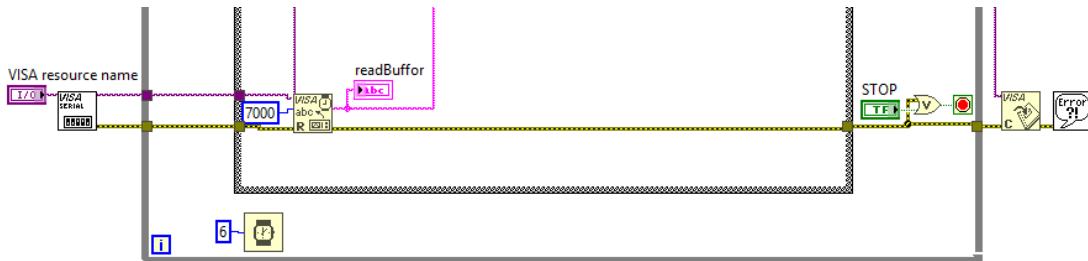
```

Rysunek 3.15. Kod programu przesyłającego wartości do portu USB.

3.3.4. Program sterujący rękawicą LabView

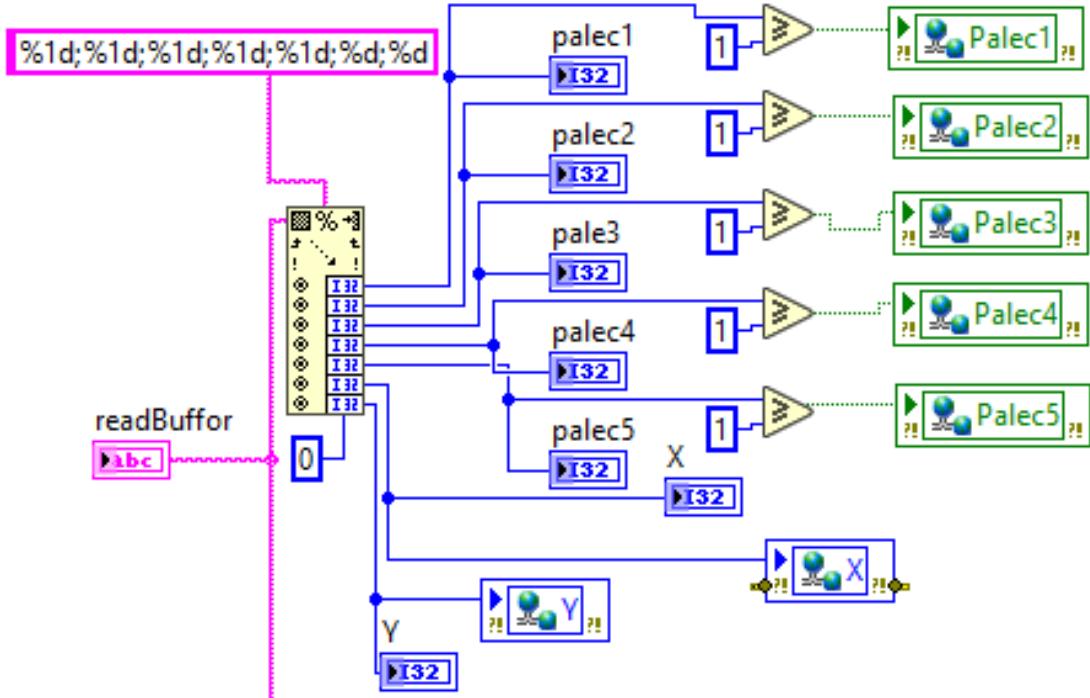
Program działający na komputerze jest zbudowany z dwóch głównych części. Pierwsza z nich ma za zadanie odczytywanie ramek danych otrzymywanych z odpowiedniego portu USB. Druga dekoduje całą otrzymaną ramkę i rozbija ją na odpowiednie zmienne tak, aby program sterujący pojazdem mógł je łatwo analizować.

Pierwsza część programu używa wbudowanej funkcjonalności środowiska *LabView*, które umożliwia odczytywanie dowolnych danych wysłanych na port USB i przedstawienie ich w postaci ciągu znaków lub macierzy. Na potrzeby projektu wygodniejszym i szybszym było przedstawienie danych w postaci ciągu znaków zapisywanych do zmiennej „*readBuffer*”. Cały system pobierania danych zamknięty jest w pętli powtarzającej odczyt oraz w strukturze warunkowej, która umożliwia wyłączenie i włączenie przesyłu w dowolnym momencie.



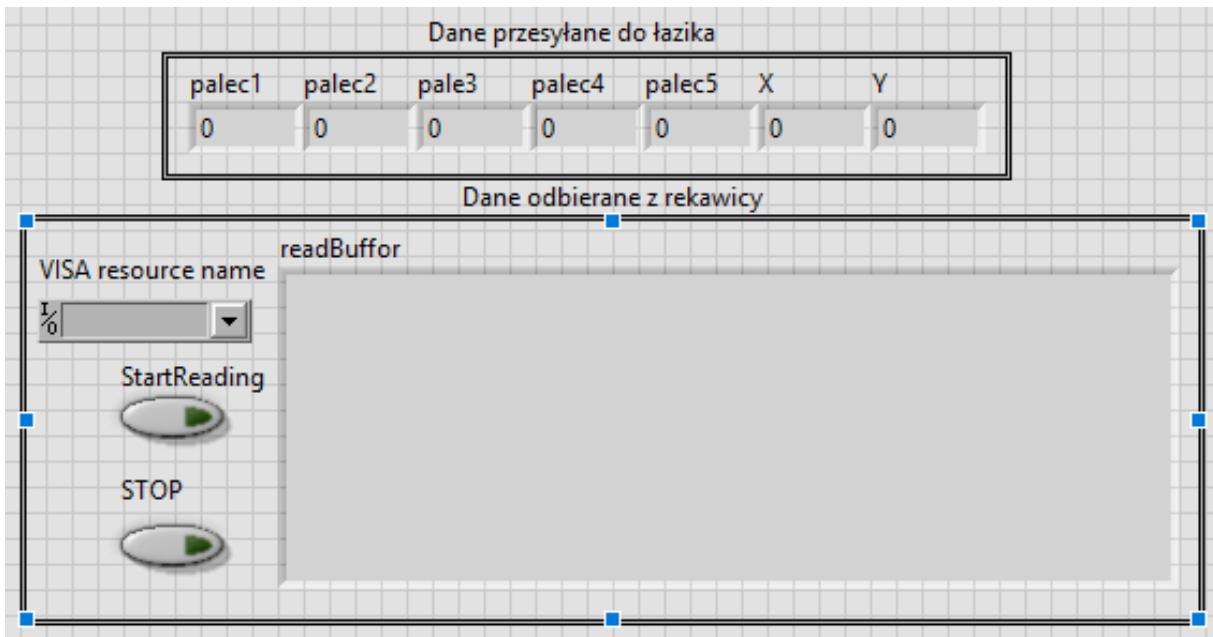
Rysunek 3.16. Struktura odbioru danych z rękawicy.

Następnie taki ciąg znaków jest rozbijany na pojedyncze wartości za pomocą odpowiedniego bloku funkcyjnego. Te wartości wysyłane są na stworzony na procesorze *myRio* serwer zmiennych, który umożliwia odczytywanie danych w ramach projektu, pomimo że zmienne znajdują się na dwóch różnych komputerach. To właśnie funkcja *SharedVariables* pozwala na sprawne przekazywanie odczytanych danych na platformę, dzięki czemu cały system sterowania jest responsywny i umożliwia łatwe sterowanie całym urządzeniem.



Rysunek 3.17. Struktura analizy danych i zapisu do serwera zmiennych.

Cały program posiada swój oddzielny interfejs nie powiązany w żaden sposób z interfejsem podwozia. Na taki sposób zdecydowano się z powodu założenia uruchomienia rękawicy jako osobnego medium. Takie rozwiązanie pozostawia elastyczna furtkę dla ewentualnego zastosowania rękawicy jako kontrolera dla innego systemu.



Rysunek 3.18. Interfejs obsługujący interaktywną rękawicę.

4. Sterowanie robotem (platforma mobilna i manipulator)

4.1. Wprowadzenie (zintegrowanie manipulator + platforma)

Integracja manipulatora oraz platformy w jednego robota mobilnego odbywała się w kilku etapach, przedstawionych w kolejnych rozdziałach. Integracji dokonano na trzech przenikających się płaszczyznach: mechanicznej, elektrycznej i programowej. Mechaniczne połączenie manipulatora i podstawy jezdnej wymagało zespawania odpowiedniej ramy, która obejmuje akumulatory zasilające cały robot oraz zawiera uchwyty, które stabilizują manipulator w trakcie ruchu. Zasilanie manipulatora i platformy (24 V) pochodzi z dwóch akumulatorów 12 V połączonych ze sobą szeregowo. Ostatnim zadaniem było połączenie programów sterujących manipulatorem i platformą. W tym rozdziale opisano interfejs komunikacyjny między głównym procesorem robota (*NI myRio*), a procesorem obsługującym napędy samej platformy (*STM32*). Równocześnie przedstawiono interfejs graficzny, za pomocą którego można sterować całym urządzeniem.

4.2. Komunikacja wewnętrz platformy mobilnej

4.2.1. Opis komunikacji między STM32, a NI myRio

Informacje pobrane z rękawicy lub interfejsu po przeanalizowaniu i wykonaniu odpowiedniej obróbki wysyłane są, przy pomocy odpowiedniego interfejsu bezprzewodowego, do procesora *NI myRio*, który znajduje się na platformie i jest głównym procesorem sterującym w układzie. Dane te mogą zostać bezpośrednio użyte do sterowania manipulatorem, jeżeli akurat taki tryb sterowania wybrał operator. Natomiast jeżeli operator ma zamiar przemieszczać platformę, to należy przesyłać informacje z procesora głównego do procesora *STM32*, który wysyła odpowiednie wartości sterujące do sterowników silników. W tym celu autor stworzył dodatkowy interfejs komunikacyjny działający przewodowo na bardzo krótką odległość. Interfejs taki spełnia bardzo specyficzne wymagania dotyczące rodzaju komunikacji oraz jej parametrów.

Interfejs umożliwia sterowanie platformą w czasie rzeczywistym, dlatego wymagany jest krótki czas odpowiedzi oraz duża szybkość transmisji danych, przy równoczesnym zachowaniu dużego stopnia bezpieczeństwa. Eliminacja błędów komunikacji jest niezwykle ważna dla zachowania bezpieczeństwa poruszania się platformy. Zastosowany interfejs UART pozwala na spełnienie tych rygorystycznych wymagań, pozwalając równocześnie na dwustronną komunikację, ponieważ do prawidłowego działania platformy potrzebne są nie tylko informacje zadające prędkość i położenie napędów platformy, ale również informacje diagnostyczne określające stan działania platformy.

Utworzona ramka wysyłana z kontrolera *NI myRio* serię danych, które umożliwiają poprawną pracę platformy. Przesyłane są pokazane w Tab. 4.1 oraz objaśnione poniżej:

- Prędkość jazdy

Wartość prędkości przedstawiona jest jako liczba z przedziału <00 ; 99>. Dla zmniejszenia ilości przesyłanych danych kierunek jazdy jest bezpośrednio przypisany do wartości prędkości. Przedział <00 ; 44> odpowiada jeździe do tyłu, gdzie liczba im bliższa zeru tym szybciej do tyłu porusza się platforma. Przedział <44 ; 54> odpowiada pozycji neutralnej, czyli platforma nie porusza się w żadnym kierunku. Przedział <54 ; 99> odpowiada jeździe do przodu. Im wyższa liczba, tym szybciej porusza się platforma.

- Skrót kół przednich

Przednia oś może obrócić się o maksymalny kąt wychylenia około 60° , w oba kierunki z punktu neutralnego. Wartość w ramce, którą wysyłamy do STM32, przyjmuje tutaj identyczną strukturę co ramka poprzednia, przesyłane są wartości z zakresu <00 ; 99>, gdzie pozycja neutralna oznaczona jest jako liczba 50, wszystkie mniejsze wartości to skrót w lewo, wszystkie większe to skrót w prawo.

- Blokada napędów, światła, klakson i reset zabezpieczenia PWM

Wszystkie funkcjonalności opisane w tym punkcie zapisane są w ramce w postaci wartości cyfrowych 1 dla włączenia działania oraz 0 dla wyłączenia działania. Blokada napędów jest to programowy hamulec. Ustawienie jej wartości na 1 powoduje natychmiastowe zatrzymanie i unieruchomienie pojazdu. Reset zabezpieczenia PWM uruchamia się automatycznie w razie problemów z komunikacją. Ustawienie tego pola na 1 powoduje zdobycie takiej blokady. Światła oraz klakson są to proste systemy sygnalizacji platformy, które pomagają w komunikacji ewentualnych problemów, które mogą wystąpić.

Tabela 4.1. Ideowa budowa ramki wysyłanych danych.

0 – 9	0 – 9	0 – 9	0 – 9	0/1	0/1	0/1	0/1
Prędkość napędów	Wychylenie osi skrętnej	Blokada napędów	Światła	Klakson	Reset zabezpieczenia PWM		

Ramka danych odbierana przez procesor *myRio* zawiera dane dotyczące informacji o funkcjonowaniu platformy i układów zasilających. Informacje te przedstawione są w Tab. 4.2 oraz objaśnione poniżej:

- Kierunek jazdy

Informacja pobierana z enkodera pozwala na potwierdzenie zgodności stanu zadanej wartości prędkości z rzeczywistym kierunkiem jazdy. W ramce danych przedstawiana jest jako jedna z cyfr należących do przedziału {0 ; 1 ; 2}, gdzie 0 to postój, 1 jazda w przód, a 2 to jazda w tył.

- Skrót

Informacje na temat aktualnego stanu układu skrętnego otrzymujemy na podstawie danych z dwóch sąsiadujących ze sobą liczb, liczby te są przekształconymi odczytami z potencjometru, który służy jako układ enkodera absolutnego. Pierwsza z komórek udostępnia takie informacje, to kierunek skrętu, jest on przedstawiany w postaci cyfr {1 ; 2} gdzie jeden odpowiada za skrót w lewo, a 2 w prawo. Druga otrzymana informacja to konkretna wartość skrętu, którą otrzymujemy w postaci dwucyfrowej liczby z przedziału <00 ; 99>. Liczby te są przeliczone tak, żeby reprezentować punkt neutralny dla wartości 50.

- Temperatura układów przekształtnikowych

Dla bezpieczeństwa układów sterujących w czasie rzeczywistym monitorowana jest temperatura układów napędowych. Oprócz automatycznych, wbudowanych w program zasad bezpieczeństwa, użytkownik powinien mieć możliwość kontrolowania temperatury. Z tego powodu do ramki została dodana liczba trzycyfrowa, która po podzieleniu przez 10 pozwala wyświetlić wartość temperatury w stopniach Celsjusza.

- Poziom naładowania akumulatorów

Podobnie do wartości temperatury do ramki została dodana kolejna liczba trzycyfrowa, która pozwala po przekształceniu, pokazać wartość napięcia akumulatorów w voltach. Wartość ta może zostać użyta do oszacowania zasięgu i pozostałoego czasu działania platformy.

- Blokada PWM

Wartość cyfrowa {0 ; 1} pozwala na określenie czy zaszła potrzeba uruchomienia procedury nagłego zatrzymania napędów. Jest to sygnał dla operatora, że należy zweryfikować poprawne działanie platformy, a po usunięciu awarii odpowiednim przyciskiem w interfejsie wyłączyć blokadę.

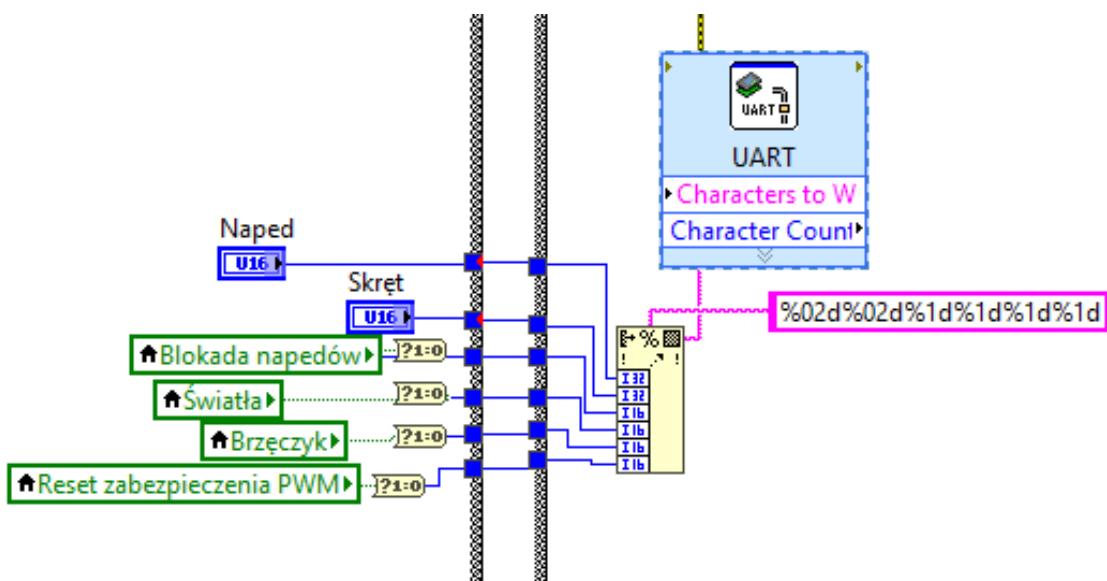
Tabela 4.2. Ideowa budowa ramki odbieranych danych.

0/1/2	1/2	0 – 9	0 – 9	0 – 9	0 – 9	0 – 9	0 – 9	0 – 9	0 – 9	1/0
Kierunek jazdy	Kierunek skrętu	Położenie osi skrętnej	Temperatura układów zasilania i sterowania		Napięcie akumulatorów		Zabezpieczenie PWM			

4.2.2. Opis programu komunikacji między STM32, a NI myRio wykonanej w LabView

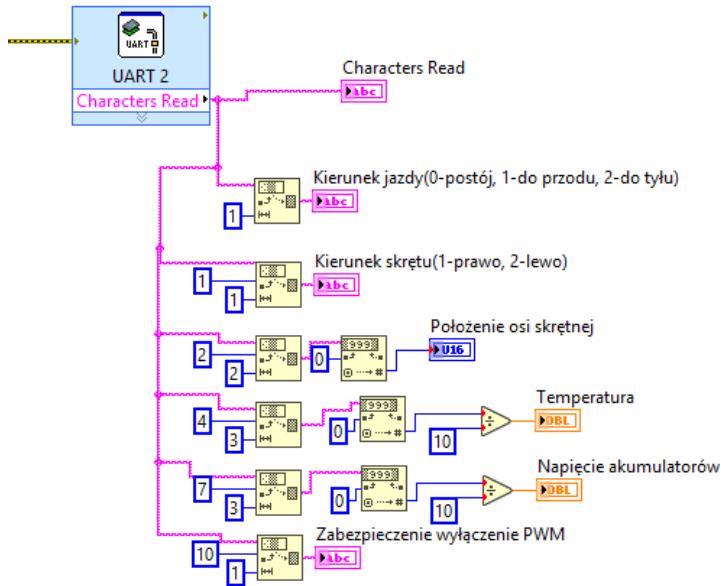
Procesor *myRio* programowany za pomocą *LabView* zawiera dedykowane biblioteki wspierające realizację dwustronnej komunikacji dla szeregu popularnych interfejsów. Predefiniowane bloki funkcyjne ustawiane są w oknach systemowych, dzięki czemu tworzenie wysyłanych ramek danych odbywa się automatycznie.

Pierwsza część programu odpowiada za wysyłanie zadanych wartości do procesora STM32. W końcowej wersji projektu pojawi się wiele różnych sposobów zadawania tych danych, natomiast na potrzeby prezentacji funkcjonalności programu pokazane zostaną tylko na bazie jednego przykładu. Dane wprowadzane są z interfejsu graficznego, gdzie zadawane są przyciskami bądź suwakami, które emulują wartości analogowe. Zmienne reprezentujące odpowiednie dane są formowane w ciąg znaków za pomocą bloku funkcyjnego. Taki sformatowany ciąg znaków przekazywany jest na blok *UART*, który buduje założoną ramkę i przesyła ją po magistrali do drugiego urządzenia. Kod wykonujący te funkcje pokazany jest na Rys. 4.1.



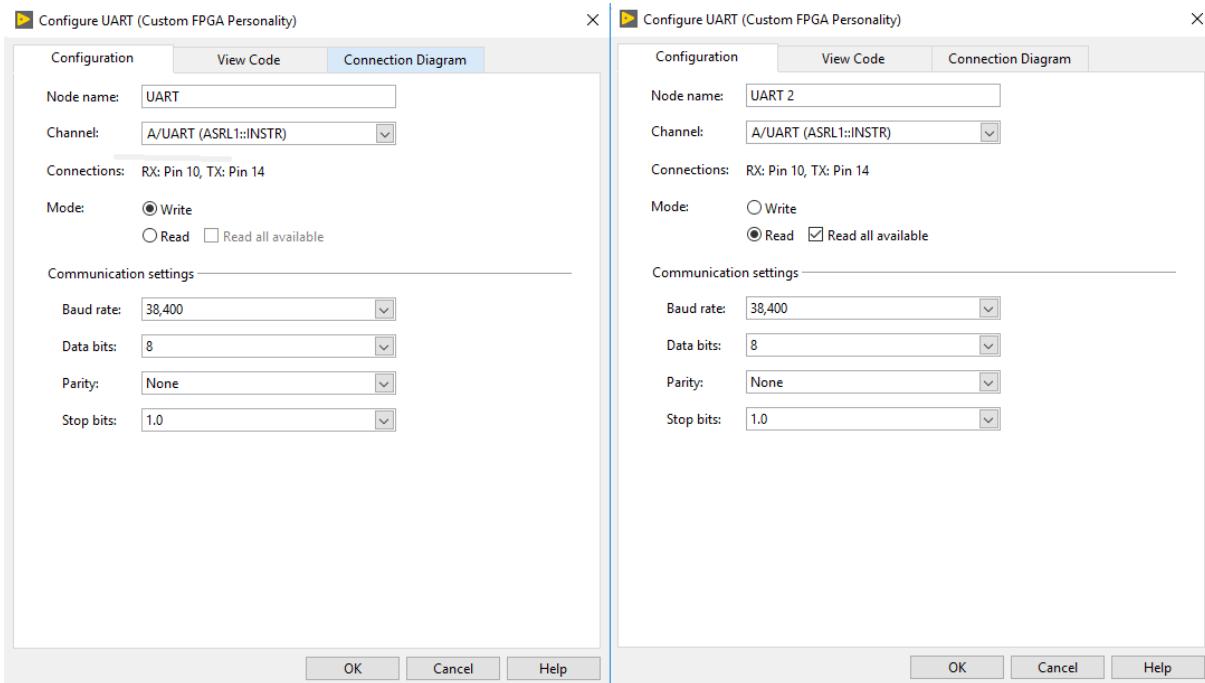
Rysunek 4.1. Układ wysyłający dane UART.

Druga część służy odbieraniu danych diagnostycznych. Tę rolę również spełnia gotowy blok funkcyjny służący do odczytywania ciągów znaków przesyłanych przez magistralę. Podobnie jak w poprzednim przypadku, konfiguracja funkcji jest dość prosta, a dane odbierane są w postaci ciągu znaków, który musi następnie zostać rozbity na pojedyncze składowe, które mogą zostać przeanalizowane i wyświetcone lub użyte w inny sposób. Kod wykonujący te funkcje pokazany jest na Rys. 4.2.



Rysunek 4.2. Układ odbierający i formatujący przychodzące dane UART.

Ustawienia transmisji danych zweryfikowano w seriach testów, w trakcie których upewniono się, że dane wysyłane są bezbłędnie, a czas przekazywania danych nie jest zbyt długi. Ustawienia zastosowane w komunikacji danych wewnętrz platformy zaprezentowane są na Rys. 4.3.



Rysunek 4.3. Okno konfiguracji interfejsu UART w bloku funkcyjnym NI myRIO.

4.3. Interfejs graficzny LabView

4.3.1. Założenia projektowe dla interfejsu graficznego

Interfejs graficzny zastosowany w projekcie ma za zadanie nie tylko informować użytkownika o aktualnym stanie w jakim znajduje się pojazd. Dodatkową funkcjonalnością interfejsu ma być awaryjny system sterowania wszystkimi elementami robota mobilnego.

Jako elementy informacyjne muszą wyświetlać się takie dane jak: odebrane informacje, wartości sterujące, które należy również wyświetlić je poprzez odpowiednie ikony. Poniżej opisane są dokładnie wszystkie potrzebne elementy interfejsu:

- Przetworzone dane otrzymane z rękawicy:
 - Odczytany ciąg znaków przed sformatowaniem;
 - Przetworzone na wartości liczbowe odczyty z akcelerometrów;
 - Przetworzone na wartości cyfrowe odczyty z czujników ugięcia;
- Wszystkie informacje otrzymywane z platformy:
 - Odczytany ciąg znaków przez sformatowaniem;
 - Wyświetlone w odpowiednich miejscach reszta informacji z ramki;
- Wszystkie informacje wysyłane do platformy:
 - Zadana wartość skrętu osi;
 - Zadana wartość prędkości napędów;
- Wszystkie dane dotyczące działania manipulatora:
 - Aktualnie zadawane wartości wypełnienia impulsów PWM;
 - Aktualna pozycja końcówki narzędziowej w układzie bazowym;
 - Aktualnie zdefiniowane długości przegubów.

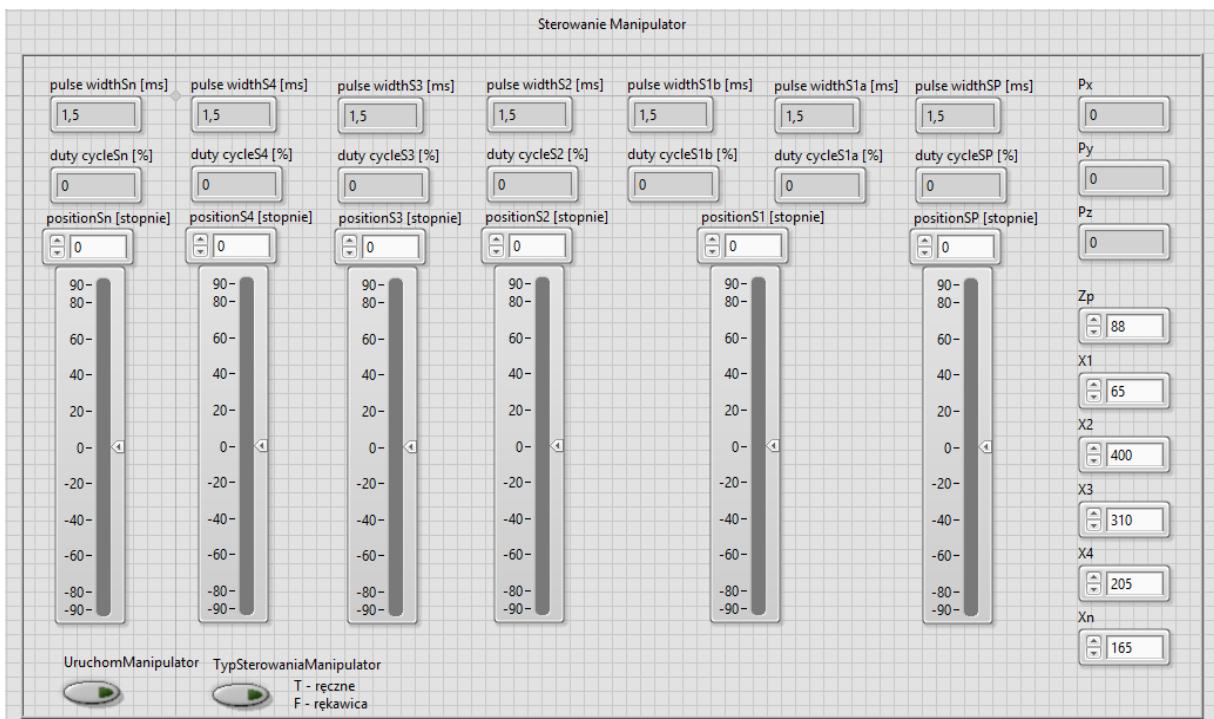
Część interfejsu odpowiedzialna za zadawanie parametrów składa głównie z przycisków oraz suwaków pozwalających określić wartości zadawanych sygnałów. Wśród tych zmiennych znajdują się informacje obsługujące takie jak:

- Sterowanie platformą:
 - Suwaki umożliwiające sterowaniem skrętem i prędkością;
 - Przyciski odpowiedzialne za światła, klakson, uruchomienie platformy, wyłączenie blokady PWM oraz określenie czy użytkownik chce sterować za pomocą rękawicy czy suwaków.
- Sterowanie manipulatorem:
 - Suwaki odpowiadające za sterowaniem każdym z serwomechanizmów;
 - Przyciski odpowiadające za uruchomienie manipulatora, wyłączenie blokady napędów oraz zmianę sposobu sterowania;
 - Pola numeryczne pozwalające na zmianę długości przegubów manipulatora.

4.3.2. Program LabView

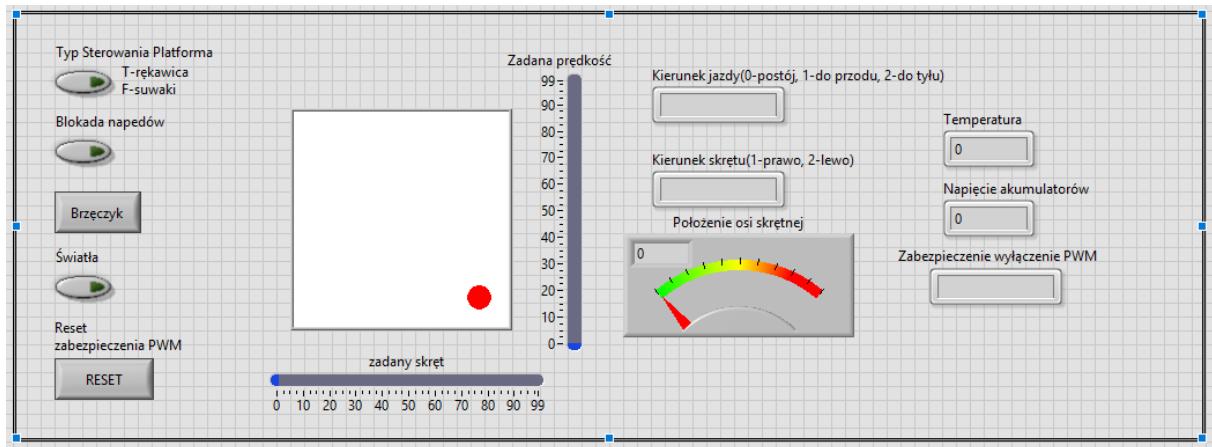
Graficzny interfejs użytkownika został podzielony na trzy części. Z uwagi na brak możliwości równoczesnego operowania platformą i manipulatorem, te dwie kategorie sterowania zostały rozdzielone i umieszczone w oddzielnych częściach ekranu. Informacje otrzymywane z rękawicy znajdują się w najmniej dostępnym miejscu z uwagi na niewielką potrzebę ciągłego ich obserwowania.

Strefa zajmująca się sterowaniem manipulatorem zawiera w sobie siedem suwaków, które w trybie sterowania ręcznego działają jako emulowane analogowe suwaki, natomiast przy sterowaniu za pomocą rękawicy, wartości wyświetlane odpowiadają aktualnemu położeniu przegubów. Nad każdym suwakiem znajdują się równocześnie pola numeryczne odpowiadające za wyświetlenie liczbowych wartości aktualnie ustawionych parametrów. Pod suwakami umieszczone są odpowiednie przyciski uruchamiające funkcje manipulatora. Najbardziej po prawej umieszczone są pola liczbowe wyświetlające parametry długości przegubów oraz położenia narzędzia. Powyżej opisana część interfejsu znajduje się na Rys. 4.4.



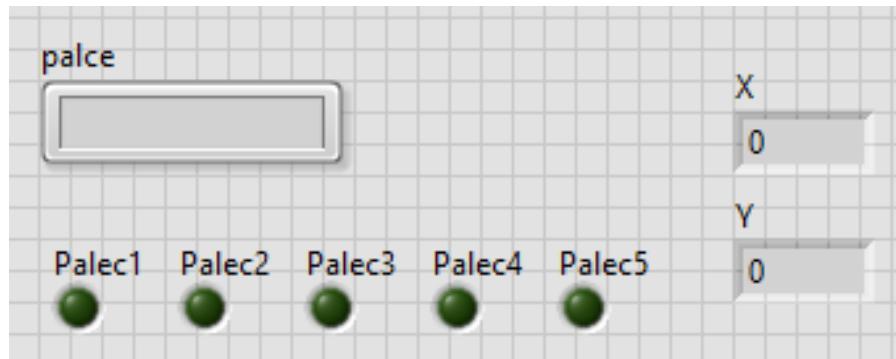
Rysunek 4.4. Interfejs obsługujący pracę manipulatora.

Strefa obsługująca sterowania oraz odczyty platformy składa się z serii przycisków obsługujących włączanie i wyłączanie funkcjonalności platformy. Następnie umieszczone są suwaki obsługujące sterowanie platformą oraz odczyty aktualnie zadawanych wartości. Najbardziej po prawej stronie znajduje się strefa odpowiedzialna za wyświetlanie informacji diagnostycznych dotyczących działania platformy. Powyżej opisana część interfejsu znajduje się na Rys. 4.5.



Rysunek 4.5. Interfejs obsługujący pracę platformy.

Ostatnim najmniejszym obszarem jest strefa odpowiedzialna za wyświetlanie informacji dotyczących danych odebranych z rękawicy. W dużym polu tekstowym wyświetlany jest pierwotny ciąg znaków, który odbierany jest z interfejsu komunikacyjnego. Poniżej znajdują się pola liczbowe i binarne wyświetlające przetworzone dane. Powyżej opisana część interfejsu znajduje się na Rys. 4.6.



Rysunek 4.6. Interfejs obsługujący odczyty z kontrolera.

4.4. Zastosowanie rękawicy jako kontrolera

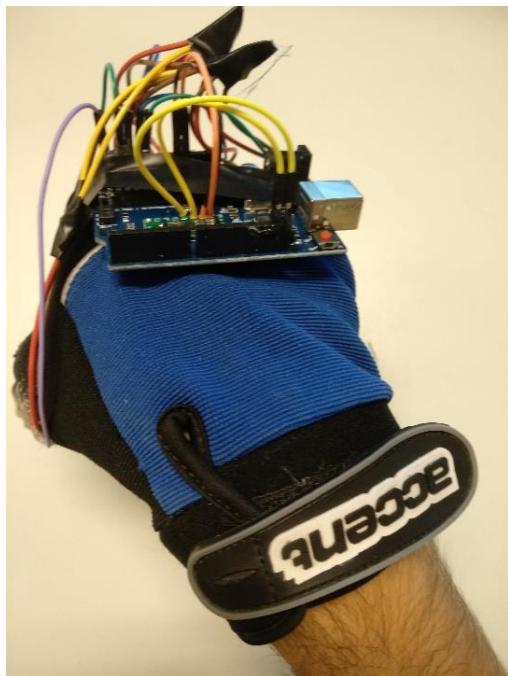
4.4.1. Zestawy gestów używane do sterowania

Sterowanie platformą odbywa się przy pomocy dwóch równoczesnych komend wydawanych przy pomocy dloni. Pierwszym typem komendy są wychylenia (skręcenia) dloni względem osi przechodzących przez jej środek. Drugim typem komendy, jaki można wydawać za pomocą rękawicy są zgięcia palców, odpowiadające binarnym przełącznikom. Zgięty palec to sygnał wysoki, prosty palec to sygnał niski. Odpowiednie sekwencje zaciśniętych palców powodują odblokowywanie się konkretnych elementów i umożliwiają sterowanie.

Jeśli zablokowane zostaną napędy manipulatora i uruchomione napędy platformy, następnie wybrany zostanie tryb sterowania rękawicą, można wydawać następujące komendy:

- Jazda do przodu

Wszystkie palce zaciśnięte, pochylenie dłoni do przodu. Gest pokazany został na Rys. 4.7.



Rysunek 4.7. Ustawienie dłoni przy jeździe do przodu.

- Jazda do tyłu

Wszystkie palce zaciśnięte, pochylenie dłoni do tyłu. Gest pokazany został na Rys. 4.8.



Rysunek 4.8. Ustawienie dłoni przy jeździe w tył.

- Skręt w prawo

Wszystkie palce zaciśnięte, skręcenie dłoni w prawo. Gest pokazany został na Rys. 4.9.



Rysunek 4.9. Ustawienie dłoni przy skręcie kołami w prawo.

- Skręt w lewo

Wszystkie palce zaciśnięte, skręcenie dłoni w lewo. Gest pokazany został na Rys. 4.10.



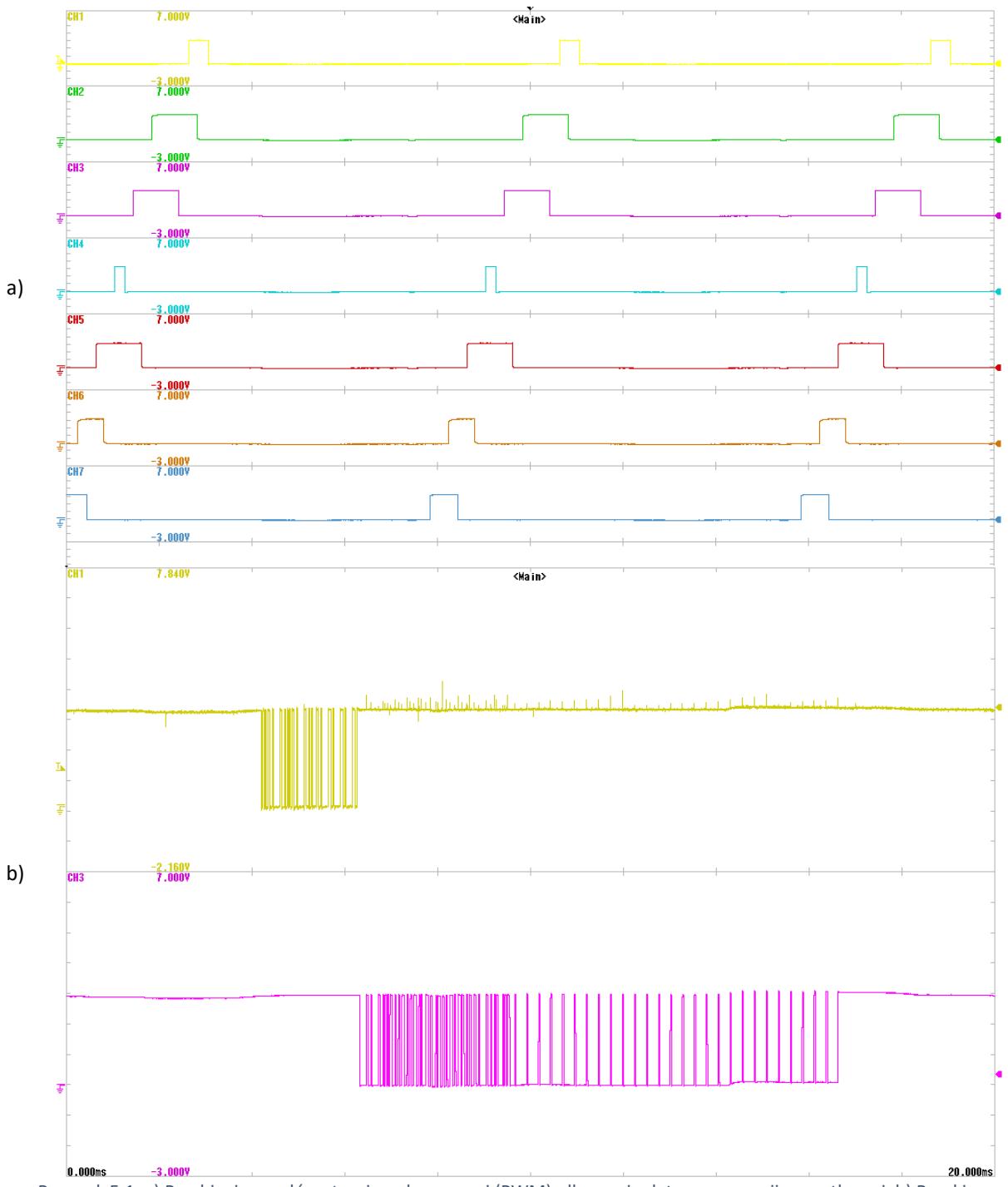
Rysunek 4.10. Ustawienie dłoni przy skręcie w lewo.

Komendy skrętu i jazdy w przód i w tył można wydawać równocześnie.

5. Badania i weryfikacja zaproponowanego rozwiązania

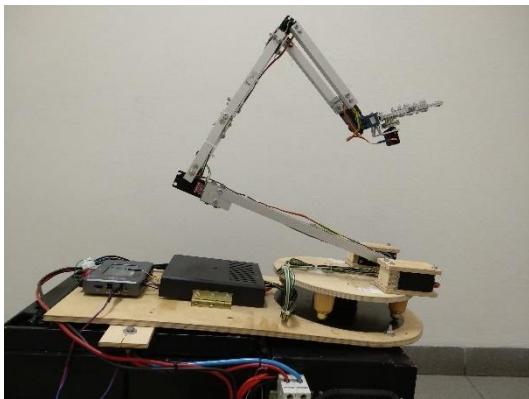
Po ukończeniu budowy i oprogramowania projektu wykonane zostały odpowiednie testy funkcjonalności oraz zweryfikowane zostały rzeczywiste założenia, które zostały opisane we wcześniejszej części pracy. Testy rozpoczęte zostały od sprawdzenia jakości logicznych sygnałów sterujących napędami oraz ramek danych wysyłanych między procesorami. Następnie przeprowadzone zostały testy mechaniczne samego manipulatora, w ramach których wykonano testy: mobilności poszczególnych przegubów, zasięgu całego robota, udźwigu przy maksymalnym wyprostowaniu oraz dokładności odczytów kinematyki prostej manipulatora. Na końcu wykonane zostały zbiorcze testy kontrolowania robota, zarówno przy pomocy interfejsu graficznego jak i rękawicy akcelerometrycznej. W trakcie tych procedur został sprawdzony zasięg bezprzewodowego interfejsu komunikacyjnego.

Pierwszy rodzaj testów odpowiedzialnych za sprawdzenie jakości sygnałów polegał na użyciu oscyloskopu do zaobserwowania przebiegów PWM, które mają sterować napędami oraz przebiegu sygnałów które reprezentują binarnie przesyłane ramki z danymi. Sygnały muszą być przesyłane bez zakłóceń i nie powinny mieć dużych przeregulowań. Przebiegi pokazane są na Rys. 5.1. Jak widać przebiegi są czytelne i nieodkształcone, a ewentualne zakłócenia są najprawdopodobniej wprowadzane przez urządzenia pomiarowe. W przypadku komunikacji UART, ramki zostały skonstruowane w taki sposób, aby ramka pochodząca z STM32 następowała jako odpowiedź dla ramki wysyłanej z *myRio*, można to zaobserwować na Rys. 5.1 b) jako przesunięcie ramek w czasie względem siebie.



Rysunek 5.1. a) Przebiegi sygnałów sterujących serwami (PWM), dla manipulatora w pozycji początkowej. b) Ramki komunikacji UART.

Druga seria testów mająca za zadanie sprawdzić mobilność pojedynczych układów napędowych została wykonana poprawnie. Przesuwając odpowiednimi suwakami w interfejsie graficznym wywołano ruchy kolejnych przegubów, co pozwoliło przejść do kolejnych faz testów mechanicznych. Pozycja początkowa ora końcowa zostały przedstawione na Rys. 5.2.



a)



b)

Rysunek 5.2. a) Ułożenie początkowe. b) Ułożenie manipulatora po wykonanym teście ruchomości przegubów.

Trzecia faza testowa miała za zadanie sprawdzić maksymalny zasięg robota i porównać go z założonymi na początku wartościami. Jeszcze przed rozpoczęciem testów oczywistym było, że robot znacznie przekroczył zakładaną długość ramienia, końcowa konstrukcja manipulatora ma długość 1200 mm. Natomiast najdalszy punkt na podłożu do jakiego sięgało ramię robota znajdował się około 1000 mm od bazy platformy. Cały zasięg przedstawiony jest na Rys. 5.3.



Rysunek 5.3. Demonstracja całkowitego zasięgu opracowanego manipulatora.

Czwarta seria testów miała na celu sprawdzenie w rzeczywistości teoretycznych wyliczeń mechanicznych dotyczących momentów poszczególnych napędów, manipulator musi mieć możliwość uniesienia lekkich przedmiotów przy maksymalnym wyprostowaniu. Niestety, robot nie był w stanie podnieść przy pełnym wyprostowaniu przegubów założonego ciężaru 500 g. Na podstawie badań eksperymentalnych stwierdzono, że udźwig robota w najbardziej niekorzystnym położeniu (wyprostowane ramię) to 200 g. Założone 500 g ramię robota mogło podnieść tylko przy częściowym ugięciu. Druga część testów została przeprowadzona bez obciążenia. W trakcie tej próby sprawdzono, czy ramię może się swobodnie poruszać przy wszystkich przegubach w pozycji wyprostowanej. Na Rys. 5.3 pokazano częściowo wyprostowane ramię z obciążeniem, natomiast na Rys. 5.4 pokazano ruch wyprostowanego ramienia.



Rysunek 5.4. Manipulator z obciążeniem.



Rysunek 5.5. a) Manipulator w pozycji początkowej przed testem. b) Podniesione i wyprostowane ramię manipulatora.

Piąty z testów mechanicznych manipulatora polegał na ustawieniu manipulatora w trzech różnych pozycjach oraz wykonaniu pomiarów odległości końcówki chwytaka od początku układu bazowego za pomocą miarki. Pomiary te zostały później porównane z odczytanymi z interfejsu wartościami wyliczonymi przez program. Dane rzeczywiste są danymi odczytanymi z miarki, a dane odczytane to te pochodzące z programu. Pomiary przedstawione zostały w Tab. 5.1.

Tabela 5.1. Zestawienie pomiarów kinematyki prostej robota.

Próba	X rzeczywisty	Y rzeczywisty	Z rzeczywisty	X odczytany	Y odczytany	Z odczytany
1	209	0	515	201	0	522
2	315	54	291	321	51	302
3	1130	-815	0	1100	-810	-10

Z powodu niewystarczającej sztywności konstrukcji i precyzji układów enkoderowych, odczyty odbiegają od wartości rzeczywistych i nie są całkowicie zadowalające. Natomiast można założyć, że obliczenia były wykonane prawidłowo z uwagi na brak dużych odchyleń.

Ostatnia seria testów miała za zadanie sprawdzić równocześnie zasięg komunikacji między komputerem sterującym, a platformą, przy jednoczesnym sprawdzeniu działania sterowania akcelerometrycznego. Pierwszy z dwóch prowadzonych równolegle testów wykazał, że zasięg komunikacji bezprzewodowej w terenie bez przeszkód to około 52m. Sterowanie rękawicą akcelerometryczną było płynne i po początkowych drobnych problemach z nawiązaniem komunikacji nie przysparzało dalszych problemów. Na Rys. 5.6 pokazana jest platforma w trakcie jazdy.



a)



b)



c)

Rysunek 5.6. Opracowany robot mobilny w trakcie jazdy: a) koła skręcone w lewo; b) koła wyprostowane; c) koła skręcone w prawo.

6. Podsumowanie i wnioski

W wyniku realizacji powyższej pracy powstał robot pięcioosiowy klasy 5R, wyposażony w mechaniczny chwytak obsługiwany przez serwomechanizmy elektryczne. Robot oraz program skonstruowane są w taki sposób, że w opracowanym systemie możliwa jest realizacja odwrotnej kinematyki robota oraz równań dynamicznych. Do sterowania i obliczeń można również zastosować programy zewnętrzne łączące się z *myRio* np. *MATLAB* lub *SIMULINK*. Jako drugą część pracy wykonano system sterownia całym pojazdem przy pomocy interaktywnej rękawicy. Rękawica może zostać użyta do dowolnego innego projektu, a odczyty z żyroskopów mogą zostać zastosowane do sterowania prędkością poruszania się robota.

Długi czas trwania projektu pozwolił nie tylko na zbudowanie i ukończenie pracy w stopniu zadowalającym, ale również na analizę zaproponowanych rozwiązań. Wiele z nich udało się zrealizować bardzo dobrze, jednak niektóre z podjętych decyzji projektowych okazały się nieoptimalne. Poniżej opisane ostaną części projektu, które zdaniem autora zrealizowano prawidłowo oraz te, które wymagają dalszego rozwoju.

Rozwiązania zrealizowane prawidłowo:

- Dobór napędów oraz zasilania.

Testy funkcjonalne wykazały, że układy napędowe manipulatora wraz z odpowiednimi układami zasilania zostały dobrane prawidłowo. Manipulator może wykonywać wszystkie założone ruchy bez obciążenia oraz z obciążeniem, które zostało opisane w założeniach. Wynika z tego, że metody matematyczne użyte do zaprojektowania manipulatora zostały również dobrane odpowiednio.

- Wykonanie kinematyki prostej manipulatora.

Testy funkcjonalne manipulatora wykazały, że odpowiednie obliczenia kinematyczne są wykonywane poprawnie dla każdej z pozycji manipulatora. Końcówka zawsze znajduje się w położeniu odpowiednim dla wartości zadanych, niestety z powodu nieidealnej sztywności manipulatora, odczyt położenia obarczony jest stosunkowo dużym błędem pomiarowym. Jednak pomimo niepewności pomiarowej oraz niedokładności urządzenia można założyć, że obliczenia zostały wykonane dobrze, a na otrzymanych równaniach można zrealizować zagadnienie odwrotnej kinematyki robota. Po poprawieniu konstrukcji całość powinna zwracać dużo lepsze wyniki.

- Realizacja interfejsów sterowania oraz komunikacji.

Dłuższy czas użytkowania pokazał, że zbudowany interfejs jest responsywny i czytelny. Główny problem, z który należało rozwiązać były opóźnienia w komunikacji, które mogły występować przy zastosowanej strukturze wielu szeregowych interfejsów. Testy wykazały, że cały system nie posiada opóźnień ani nie zrywa połączenia w trakcie pracy urządzenia.

- Wykonanie i oprogramowanie rękawicy sterującej.

Komponenty dobrane do budowy i oprogramowania rękawicy akcelerometrycznej zapewniają odpowiednią dokładność oraz responsywność. System kontrolera nie ulega awariom, a komunikacja jest stabilna. Umożliwia ona sterowanie platformą w intuicyjny i łatwy sposób.

Rozwiązania, które wymagają dopracowania i poprawek:

- Konstrukcja mechaniczna manipulatora.

Budowa manipulatora przebiegała w kilku fazach, w trakcie których zmieniały się koncepcja oraz wymagania konstrukcyjne. Spowodowało to, że początkowo podjęte decyzje projektowe nie sprawdziły się w zrealizowanej formie. Wpływęło to na obniżenie sztywności konstrukcji oraz spowodowało zmniejszenie dokładności osiąganych pozycji. Niestety w momencie, w którym takie wady zostały wykryte, projekt był w fazie, w której nie była możliwa zmiana konstrukcji.

- Brak dodatkowych funkcjonalności manipulatora.

Pomimo długiego czasu trwania projektu, większość czasu została poświęcona wykonaniu podstawowych funkcjonalności sterowania manipulatorem oraz platformą. Pierwsze plany zawierały wykonanie dodatkowo modelu manipulatora w programie *Matlab* oraz *LabView*, przy użyciu odpowiednich bibliotek. Pojawiające się jednak na bieżąco wyzwania oraz problemy nie pozwoliły na rozpoczęcie prac nad tymi funkcjonalnościami. Bez odpowiedniego modelu, który umożliwiłby realizację zagadnień kinematyki odwrotnej manipulatora, sterowanie poszczególnymi przegubami jest niepraktyczne. Wykonywanie dokładnych ruchów PTP (ang. „*point to point*”) oraz zaimplementowanie interpolacji i ruchu liniowego bardzo ułatwiałoby sposób manipulowania robotem. Opisane braki w funkcjonalności sterowania manipulatorem uniemożliwiły zastosowanie sterowania akcelerometrycznego do poruszania manipulatorem. Niestety bez zaimplementowanej kinematyki odwrotnej dla manipulatora nie ma możliwości użytkownego zastosowania rękawicy.

Projekt platformy mobilnej z manipulatorem był niezwykle dużym i rozbudowanym zagadnieniem, które może być rozwijane i kontynuowane.

Pierwszym i najważniejszym krokiem, który należy podjąć w ramach rozwoju projektu jest usztywnienie konstrukcji poprzez poprawienie budowy podstawy manipulatora, oraz usztywnienie łączów przegubów. Zastosowanie druku 3D oraz frezowanych aluminiowych łączów i elementów wzmacniających pozwoli uzyskać większą sztywność konstrukcji. Drugą bardzo ważną czynnością, którą należy wykonać i dopracować w ramach kontynuowania projektu jest zastosowanie odpowiednich bibliotek (np. tych dostarczanych przez środowiska *Matlab* oraz *LabView*) do realizacji równań odwrotnej kinematyki manipulatora. Robot o 5stopniach swobody wymaga nieliniowego podejścia do rozwiązywania równań kinematycznych, ponieważ posiada więcej niż jedno rozwiązanie kinematyki odwrotnej dla prawie wszystkich punktów w swojej przestrzeni roboczej. Zastosowanie wspomnianych wcześniej bibliotek pozwoli na łatwiejsze i szybsze obliczanie i dobranie końcowej pozycji każdego z przegubów. Kolejny krok rozwoju projektu to implementacja ruchów PTP (ang. „*point to point*”) oraz liniowych. Opisane wyżej biblioteki pozwalają na automatyczne generowanie tego typu ruchów, więc po wykonaniu odpowiednich modeli fizycznych ten krok powinien zostać zrealizowany automatycznie przez zastosowane oprogramowanie. Dalszym etapem rozwoju projektu byłoby przystosowanie rękawicy akcelerometrycznej do sterowania manipulatorem stosując odwrotne zagadnienie kinematyki.

Na podstawie wykonanych testów oraz ilości pracy włożonej w realizowany projekt autor przyjął, że przyjęte założenia początkowe zostały spełnione i projekt zrealizowano pomyślnie. Tak rozbudowane zagadnienie wymaga dalszych prac i poprawek, jednak w ramach realizacji pracy opracowano w pełni funkcjonalny manipulator, zdobywając bardzo szeroką wiedzę z zakresu robotyki oraz elektroniki.

7. Bibliografia

- [1] Westinghouse, [Online]. Available: <http://cyberneticzoo.com/robots/1927-televox-wensley-american/>.
- [2] D. Bednarski, Praca Dyplomowa Magisterska "Opracowanie platformy sterująco-pomiarowej wraz z interfejsem użytkownika dla platformy mobilnej", Warszawa, 2017.
- [3] M. Pluta, Praca Dyplomowa Magisterska "Opracowanie przekształtnika energoelektronicznego współpracującego z silnikiem prądu stałego platformy mobilnej", Warszawa, 2017.
- [4] ABB, ABB IRB1200, [Online]. Available: <https://new.abb.com/products/robotics/pl/roboty-przemyslowe/irb-1200>. [Data uzyskania dostępu: 14 Styczeń 2019].
- [5] KUKA, KUKA LBR IIWA, [Online]. Available: <https://www.robotyka.com/produkt.php/produkt.4891/LBR%20iiwa>. [Data uzyskania dostępu: 14 Styczeń 2019].
- [6] Kawasaki, Kawasaki MIRHook 100, [Online]. Available: <https://www.astor.com.pl/produkty/robotyzacja/roboty-mir.html>. [Data uzyskania dostępu: 14 Styczeń 2019].
- [7] KUKA, Specyfikacja techniczna KUKA YouBot 480, [Online]. Available: <https://www.electromate.com/assets/catalog-library/pdfs/maxon/ScientificResearchRobot.pdf>. [Data uzyskania dostępu: 14 Styczeń 2019].
- [8] SparkFun, uArm Swift Pro, [Online]. Available: <https://www.sparkfun.com/products/14342> . [Data uzyskania dostępu: 14 Styczeń 2019].
- [9] [Online]. Available: <http://store.linksprite.com/diy-4-axis-servos-control-palletizing-robot-arm-model-for-arduino-uno-mega2560/>. [Data uzyskania dostępu: 14 Styczeń 2019].
- [10] J. Misiak, Mechanika ogólna tom 1 Statyka i Kinematyka, WTN, 2009.
- [11] B. Modelarski. [Online]. Available: <https://www.gimmik.net/blog/budowa-serwomechanizmu/>. [Data uzyskania dostępu: 14 Styczeń 2019].
- [12] Botland, Specyfikacja techniczna Power HD 1235MG, [Online]. Available: <https://www.pololu.com/file/0J706/HD-1235MG.pdf>. [Data uzyskania dostępu: 14 Styczeń 2019].
- [13] JX, JX PDI 6221MG, [Online]. Available: <http://www.jx-servo.com/English/Product/0951873936.html>\. [Data uzyskania dostępu: 14 Styczeń 2019].
- [14] Servo S3315D, [Online]. Available: <https://pl.aliexpress.com/item/F16692-HDKJ-S3315D-15Kg-60g-Robot-Torque-Metal-Gear-Digital-Servo-Set-180-Degree-Wide-Angle/32596852089.html?spm=a2g0s.9042311.0.0.27425c0fywhOhv>. [Data uzyskania dostępu: 14 Styczeń 2019].
- [15] Botland, SparkFun RoboticClaw MKII, [Online]. Available: <https://botland.com.pl/pl/chwytki-uchwyty-gimbale/1628-chwytek-metalowy-robotic-claw-mkii.html>. [Data uzyskania dostępu: 14 Styczeń 2019].
- [16] Botland, Specyfikacja techniczna SO5NF STD, [Online]. Available: http://botland.com.pl/index.php?controller=attachment&id_attachment=482. [Data uzyskania dostępu: 14 Styczeń 2019].
- [17] Botland, Specyfikacja Techniczna Pololu D24V150F7, [Online]. Available: <https://botland.com.pl/pl/przetwornice-step-down/7615-pololu-d24v150f7-przetwornica-step-down-75v-15a.html>. [Data uzyskania dostępu: 14 Styczeń 2019].
- [18] Botland, Spesyfikacja techniczna XL4015, [Online]. Available: <http://i-makers.info/resource/XL4015%20datasheet.pdf>. [Data uzyskania dostępu: 14 Styczeń 2019].
- [19] Castorama, Specyfikacja techniczna rura czworokątna Cezar, [Online]. Available:

- <https://www.castorama.pl/produkty/narzedzia-i-artykuly/artykuly-metalowe/okucia-budowlane/profile/rura-czworokatna-cezar-1-m-25-x-25-x-1-5-mm-aluminium-srebrne.html?id=1018704>. [Data uzyskania dostępu: 14 Styczeń 2019].
- [20] T. Szkodny, Podstawy Robotyki Wydanie 2, Wydawnictwo Politechniki Śląskiej , 2012.
- [21] Z. Struzik, Praca Dyplomowa Magisterska "Manipulator Przeznaczony Do Celów Dydaktycznych", Wrocław, 2005.
- [22] Arduino, Specyfikacja techniczna Arduino MEga, [Online]. Available: <http://www.mantech.co.za/datasheets/products/a000047.pdf>. [Data uzyskania dostępu: 14 Styczeń 2019].
- [23] D. STM, Specyfikacja techniczna STM32 F411, [Online]. Available: https://www.st.com/content/ccc/resource/technical/document/user_manual/e9/d2/00/5e/15/46/44/0e/DM00148985.pdf/files/DM00148985.pdf/jcr:content/translations/en.DM00148985.pdf. [Data uzyskania dostępu: 14 Styczeń 2019].
- [24] S. CubeMx, Specyfikacja techniczna STM CubeMX, [Online]. Available: https://www.st.com/content/ccc/resource/technical/document/user_manual/10/c5/1a/43/3a/70/43/7d/DM00104712.pdf/files/DM00104712.pdf/jcr:content/translations/en.DM00104712.pdf. [Data uzyskania dostępu: 14 Styczeń 2019].
- [25] N. Instruments, Specyfikacja techniczna NI myRio, [Online]. Available: <http://www.ni.com/pdf/manuals/376047c.pdf>. [Data uzyskania dostępu: 14 Styczeń 2019].
- [26] PWM, [Online]. Available: <https://stapulawski.wordpress.com/2017/03/19/w10iotrpi3-pwm/>. [Data uzyskania dostępu: 14 Styczeń 2019].
- [27] ArduinoRC, Emulowany Pad Android, [Online]. Available: <https://play.google.com/store/apps/details?id=eu.jahnestacado.arduino&hl=pl>. [Data uzyskania dostępu: 14 Styczeń 2019].
- [28] Xbox, Pad Xbox, [Online]. Available: <https://www.x-kom.pl/p/76633-pad-microsoft-pad-xbox-360-wireless-controller-xbox.html#modal:galeria>. [Data uzyskania dostępu: 14 Styczeń 2019].
- [29] Carefour, PS3, [Online]. Available: <https://www.carrefour.pl/quer-bezprzewodowy-pad-pc-ps3-dual-shock>. [Data uzyskania dostępu: 14 Styczeń 2019].
- [30] RC, Kontroler RC, [Online]. Available: <http://sklepavatar.pl/pl/drony/aparatury-rc/>. [Data uzyskania dostępu: 14 Styczeń 2019].
- [31] Robot daVinci, [Online]. Available: https://en.wikipedia.org/wiki/Da_Vinci_Surgical_System. [Data uzyskania dostępu: 14 Styczeń 2019].
- [32] B. Lofvendahl, Augmented Reality Applications for Industrial Robos, [Online]. Available: <https://www.diva-portal.org/smash/get/diva2:706980/FULLTEXT01.pdf>. [Data uzyskania dostępu: 14 Styczeń 2019].
- [33] Surgery Robot daVinci, [Online]. Available: <http://fortune.com/2016/04/20/surgery-robot-company/>. [Data uzyskania dostępu: 14 Styczeń 2019].
- [34] Wikipedia, Nintendo Power Glove, [Online]. Available: https://en.wikipedia.org/wiki/Power_Glove. [Data uzyskania dostępu: 14 Styczeń 2019].
- [35] S. T. B. Ź. Andrzej Michalski, "Laboratorium miernictwa wielkości niewielkiej elektrycznych" Wydanie 2, Warszawa: OWPW, 1999.
- [36] Sparkfun, Sparkfun Czujnik Ugięcia 73x63mm, [Online]. Available: <https://botland.com.pl/pl/czujniki-nacisku/1640-czujnik-ugiecia-73x63mm-sparkfun.html>. [Data uzyskania dostępu: 14 Styczeń 2019].
- [37] Sparkfun, Specyfikacja techniczna Sparkfun LSM6DS3, [Online]. Available: <https://botland.com.pl/pl/akcelerometry/4573-lsm6ds3-3-osiowy-akcelerometr-i-zyroskop-i2cspi-modul>

sparkfun.html. [Data uzyskania dostępu: 14 Styczeń 2019].

- [38] Pololu, Specyfikacja techniczna Pololu LSM6DS33, [Online]. Available: <https://botland.com.pl/pl/akcelerometry/5331-pololu-lsm6ds33-3-osiowy-akcelerometr-i-zyroskop-i2cspi.html>. [Data uzyskania dostępu: 14 Styczeń 2019].
- [39] MINI-M0, Specyfikacja techniczna MINI-M0 for STM32, [Online]. Available: <https://www.tme.eu/pl/Document/df78cdbb6eb37ea47036021bf436230e/mikroe1367.pdf>. [Data uzyskania dostępu: 14 Styczeń 2019].
- [40] Fernel, Specyfikacja techniczna Arduino Micro, [Online]. Available: <http://www.farnell.com/datasheets/1685581.pdf>. [Data uzyskania dostępu: 14 Styczeń 2019].
- [41] Arduino, Specyfikacja techniczna Arduino UNO ref3, [Online]. Available: <https://datasheet.octopart.com/A000066-Arduino-datasheet-38879526.pdf>. [Data uzyskania dostępu: 14 Styczeń 2019].
- [42] Pololu, Specyfikacja techniczna biblioteki LSM6.h, [Online]. Available: <https://github.com/pololu/lsm6-arduino>. [Data uzyskania dostępu: 14 Styczeń 2019].
- [43] Arduino, Specyfikacja techniczna biblioteki Wire.h, [Online]. Available: <https://www.arduino.cc/en/reference/wire>. [Data uzyskania dostępu: 14 Styczeń 2019].

8. Spis rysunków

Rysunek 1.1. Uproszczony schemat komunikacji w zrealizowanym projekcie.	12
Rysunek 2.1. ABB IRB1200 [4].	15
Rysunek 2.2. KUKA LBR IIWA [5].	15
Rysunek 2.3. KAWASAKI MIRHook 100 [6].	16
Rysunek 2.4. KUKA YouBot 480 [7].	16
Rysunek 2.5.uArm Swift Pro [8].	17
Rysunek 2.6.DIY 4-Axis Servos Control Palletizing Robot Arm Model [9].	18
Rysunek 2.7. Platforma mobilna wykorzystana w realizowanym projekcie.	19
Rysunek 2.8. Poglądowy rysunek manipulatora.	20
Rysunek 2.9.Schemat ostatniego przegubu.	21
Rysunek 2.10. Schemat trzeciego przegubu.	22
Rysunek 2.11. Schemat drugiego przegubu.	22
Rysunek 2.12 Schemat pierwszego przegubu	23
Rysunek 2.13 Przekrój budowy serwomechanizmu stosowanego w modelarstwie [11].	24
Rysunek 2.14.Serwomechanizm Power HD 1235MG [12].	25
Rysunek 2.15. Serwomechanizm JX PDI-6221MG [13].	26
Rysunek 2.16. Serwomechanizm ServoS3315D zastosowany dla przegubu 4 [14].	26
Rysunek 2.17. Chwytek RoboticClaw MKII [15].	27
Rysunek 2.18. Serwomechanizm DSG S90 S05NF – medium [16].	28
Rysunek 2.19. Pololu D24V150F7 [17].	29
Rysunek 2.20.Przetwornica XL4015 [18].	29
Rysunek 2.21. Dolna część podstawy wraz z umieszczoną wyżej obrotową platformą.	30
Rysunek 2.22. Połączenie dolnej i górnej części podstawy robota	31
Rysunek 2.23. Mocowanie napędu przegubu pierwszego.	31
Rysunek 2.24. Układ napędowy przegubu pierwszego.	32
Rysunek 2.25. Układ napędowy przegubu drugiego.	32
Rysunek 2.26. Układ napędowy przegubu trzeciego.	33
Rysunek 2.27. Układ napędowy czwartego przegubu.	33
Rysunek 2.28. Aluminiowy profil [19].	34
Rysunek 2.29. Przykład połączenia usztywniającego ognisko.	34
Rysunek 2.30. Gotowa konstrukcja razem z układami zasilania i sterowania.	34
Rysunek 2.31. Poglądowy rysunek manipulatora z przesunięciami.	35
Rysunek 2.32. Układy kinematyczne w manipulatorze.	35
Rysunek 2.33.Skrypt służący do analitycznego wyznaczenia pozycji manipulator.	39
Rysunek 2.34. Położenie wyświetlane w interfejsie.	39
Rysunek 2.35.Arduino Mega [22].	40

Rysunek 2.36. STM3F411 Discovery [23].....	41
Rysunek 2.37. NI myRio [25].	42
Rysunek 2.38. Opis wejść i wyjść NI myRio [25].	42
Rysunek 2.39. Nowy projekt.	43
Rysunek 2.40.a) Okno interfejsu. b) Okno diagramu logicznego.....	44
Rysunek 2.41. Sygnał PWM o różnym wypełnieniu [26].	45
Rysunek 2.42. Zaimplementowane przekształcenia zadawanego kąta na wypełnienie sygnału PWM.....	46
Rysunek 2.43. Suwaki pozwalające na sterowanie manipulatorem.....	46
Rysunek 2.44. Przyciski obsługujące manipulator.....	46
Rysunek 3.1. Emulowany pad Android [27].	47
Rysunek 3.2. Kolejno: Kontroler RC, Kontroler PS3 oraz Kontroler Xbox360 [28] [29] [30].	48
Rysunek 3.3. Panel sterujący robota DaVinci [33].....	49
Rysunek 3.4. Nintendo Power Glove [34].	50
Rysunek 3.5. Rękawica interaktywna - sterownik platformy.	50
Rysunek 3.6. Czujnik ugięcia SparkFun [36].	52
Rysunek 3.7. SparkFun LSM6DS3[34].....	53
Rysunek 3.8. Pololu LSM6DS33 [38].....	54
Rysunek 3.9. MINI-M0 FOR STM32 [39].	56
Rysunek 3.10. Arduino Micro [40].	57
Rysunek 3.11. Arduino UNO ref 3 [41].	58
Rysunek 3.12 Graficzna reprezentacja wysyłanej ramki danych	58
Rysunek 3.13. Kod programu obsługującego czujniki ugięcia.	59
Rysunek 3.14. Kod programu obsługującego akcelerometr I2C.....	60
Rysunek 3.15. Kod programu przesyłającego wartości do portu USB.....	61
Rysunek 3.16. Struktura odbioru danych z rękawicy.	61
Rysunek 3.17. Struktura analizy danych i zapisu do serwera zmiennych.....	62
Rysunek 3.18. Interfejs obsługujący interaktywną rękawicę.	62
Rysunek 4.1. Układ wysyłający dane UART.	65
Rysunek 4.2. Układ odbierający i formatujący przychodzące dane UART.....	66
Rysunek 4.3. Okno konfiguracji interfejsu UART w bloku funkcyjnym NI myRIO.....	66
Rysunek 4.4. Interfejs obsługujący pracę manipulatora.	68
Rysunek 4.5. Interfejs obsługujący pracę platformy.	69
Rysunek 4.6. Interfejs obsługujący odczyty z kontrolera.	69
Rysunek 4.7. Ustawienie dloni przy jeździe do przodu.	70
Rysunek 4.8. Ustawienie dloni przy jeździe w tył.....	70
Rysunek 4.9. Ustawienie dloni przy skręcie kołami w prawo.....	71
Rysunek 4.10. Ustawienie dloni przy skręcie w lewo.	71

Rysunek 5.1. a) Przebiegi sygnałów sterujących serwami (PWM), dla manipulatora w pozycji początkowej. b) Ramki komunikacji UART.....	73
Rysunek 5.2. a) Ułożenie początkowe. b) Ułożenie manipulatora po wykonanym teście ruchomości przegubów.....	74
Rysunek 5.3. Demonstracja całkowitego zasięgu opracowanego manipulatora.	74
Rysunek 5.4. Manipulator z obciążeniem.	75
Rysunek 5.5. a) Manipulator w pozycji początkowej przed testem. b) Podniesione i wyprostowane ramię manipulatora....	75
Rysunek 5.6. Opracowany robot mobilny w trakcie jazdy: a) koła skręcone w lewo; b) koła wyprostowane; c) koła skręcone w prawo.....	76
Rysunek 10.1. Zmodyfikowana podstawa serwomechanizmu.	86
Rysunek 10.2. Zastosowane przerobione serwomechanizmu 1.	86
Rysunek 10.3. Zastosowane przerobione serwomechanizmu 2.	87
Rysunek 10.4. Schemat płytki PCB sterowania manipulatorem.	87
Rysunek 10.5. Dolna strona zaprojektowanej płytki.....	88
Rysunek 10.6. Górna strona zaprojektowanej płytki.....	88
Rysunek 10.7. Schemat połączeń elektronicznych w interaktywnej rękawicy.	89
Rysunek 10.8. Kod użyty do sterowania rękawicą.....	91

9. Spis Tabel

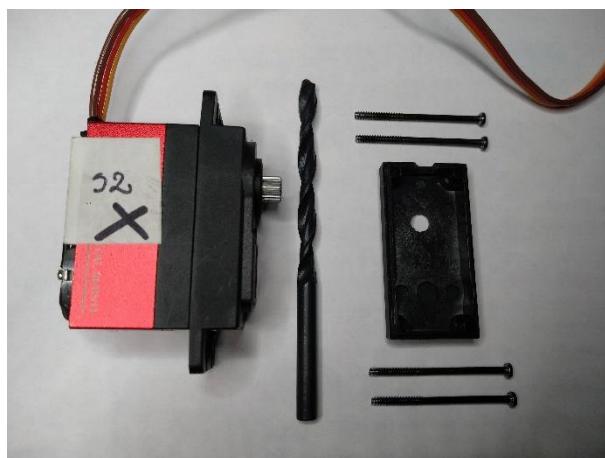
Tabela 2.1. Opis nazewnictwa ogniw manipulatora.....	20
Tabela 2.2. Opis nazewnictwa przegubów manipulatora.....	20
Tabela 2.3. Parametry mechaniczne manipulatora.....	23
Tabela 2.4. Parametry serwonapędu Power HD 1235 MG.....	25
Tabela 2.5. Parametry serwonapędu JX PDI-6221MG.....	26
Tabela 2.6. Parametry serwonapędu Servo S3315D.	27
Tabela 2.7. Parametry chwytaka Robotic Claw MKII.....	27
Tabela 2.8. Parametry serwonapędu DSG S90 S05NF.....	28
Tabela 2.9. Parametry przetwornicy Pololu D24V15F7.....	29
Tabela 2.10. Parametry przetwornicy XL4015.	29
Tabela 2.11Parametry notacji DH opracowanego manipulatora	36
Tabela 2.12. Wymagania wejść i wyjść procesora głównego.	40
Tabela 2.13. Parametry kontrolera Arduino Mega.	40
Tabela 2.14. Parametry kontrolera STM32F411 Discovery.	41
Tabela 2.15. Parametry kontrolera NI myRio.....	42
Tabela 3.1. Parametry Czujnika Ugięcia SparkFun.	52
Tabela 3.2 Parametry akcelerometru LSM6DS3.....	53
Tabela 3.3. Parametry akcelerometru LSM6DS33.....	54
Tabela 3.4. Wymagania wejść i wyjść procesora rękawicy.	55
Tabela 3.5. Parametry kontrolera MINI-M0 FOR STM32.	56
Tabela 3.6. Parametry kontrolera Arduino Micro.	57
Tabela 3.7. Parametry kontrolera Arduino UNO ref 3.....	58
Tabela 4.1. Ideowa budowa ramki wysyłanych danych.	64
Tabela 4.2. Ideowa budowa ramki odbieranych danych.....	65
Tabela 5.1. Zestawienie pomiarów kinematyki prostej robota.....	75

10. Dodatek

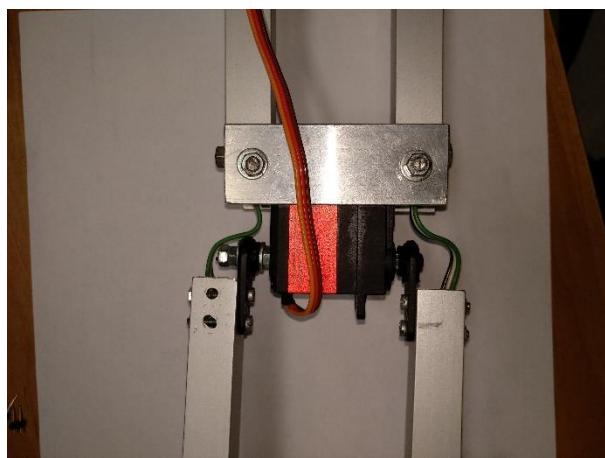
- Modyfikacje serwomechanizmu jednostronnego na dwustronne

Jeszcze w fazie projektowej konstrukcji pojawił się problem związany z jednostronną osią wybranych serwonapędów dla przegubów drugiego i trzeciego. Brak dwustronnej osi powodował skręcanie się konstrukcji oraz znaczne osłabienie sztywności całego robota. Z tego powodu należało wykonać operację dodania drugiej osi do obu napędów. Osie te miały za zadanie wspierać konstrukcję w newralgicznych punktach i nie pozwolić na działanie sił skręcających. Modyfikacja ta wymuszona była przez brak dostępnych na rynku gotowych rozwiązań, które miałyby wystarczające możliwości momentu napędowego.

Dodanie osi polegało na zdjęciu pokrywki z dolnej części mechanizmu i wywierceniu w niej dziury dokładnie w osi działania rzeczywistego napędu. Następnie przełożono przez wywiercony otwór odpowiedniej grubości śrubę i orczyk, na którym została zamontowana druga strona przegubu. Rysunki 10.1, 10.2 i 10.3 pokazują odpowiednio zmodyfikowany serwomechanizm i złożoną stronę wspierającą przegubu.



Rysunek 10.1. Zmodyfikowana podstawa serwomechanizmu.

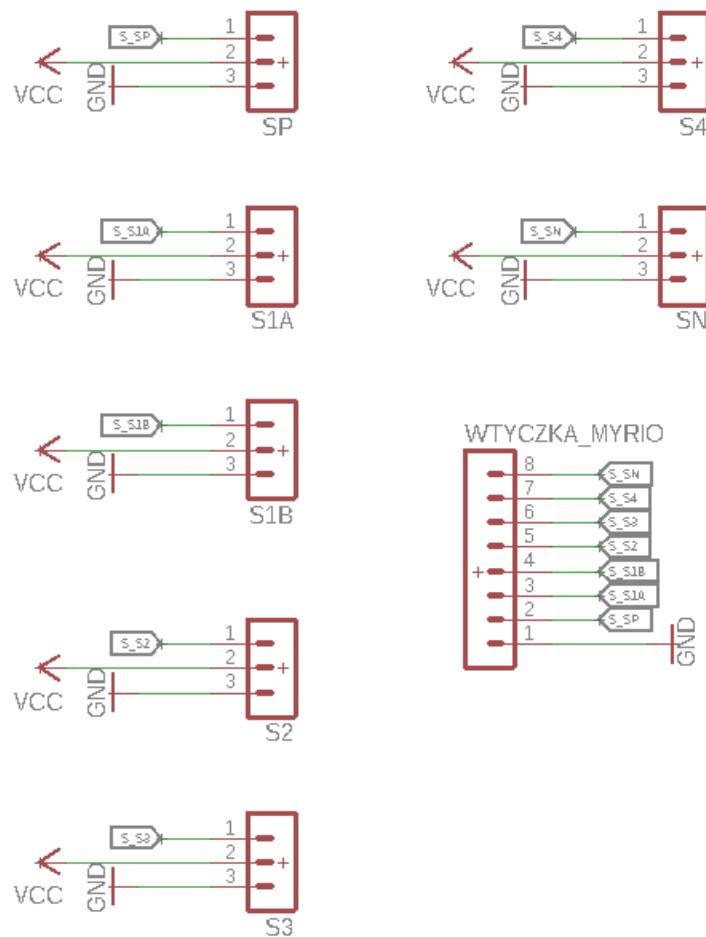


Rysunek 10.2. Zastosowane przerobione serwomechanizmu 1.

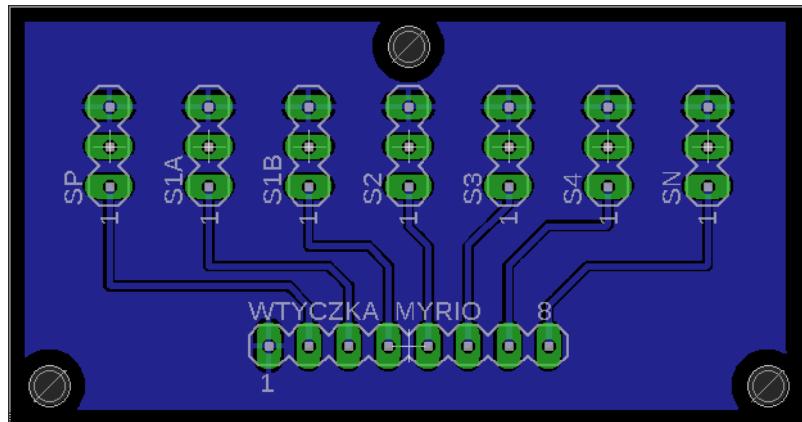


Rysunek 10.3. Zastosowane przerobione serwomechanizmu 2.

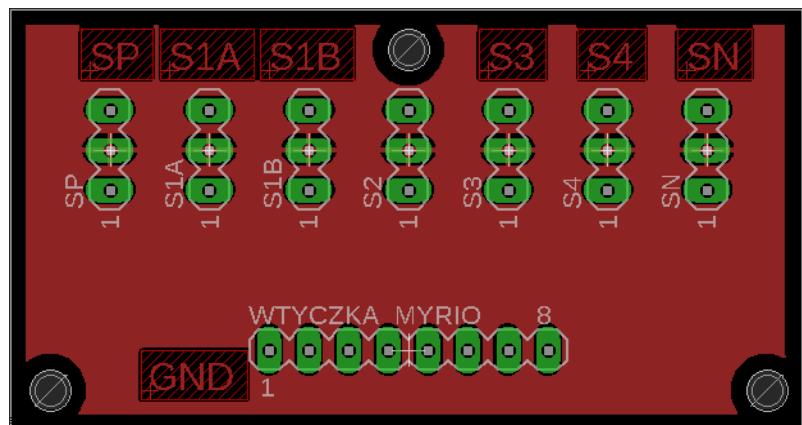
- Schematy elektryczne sterowania manipulatorem Rys. 10.4 oraz warstwy górna i dolna zaprojektowanych płytka rysunki 10.5 oraz 10.6.



Rysunek 10.4. Schemat płytki PCB sterowania manipulatorem.

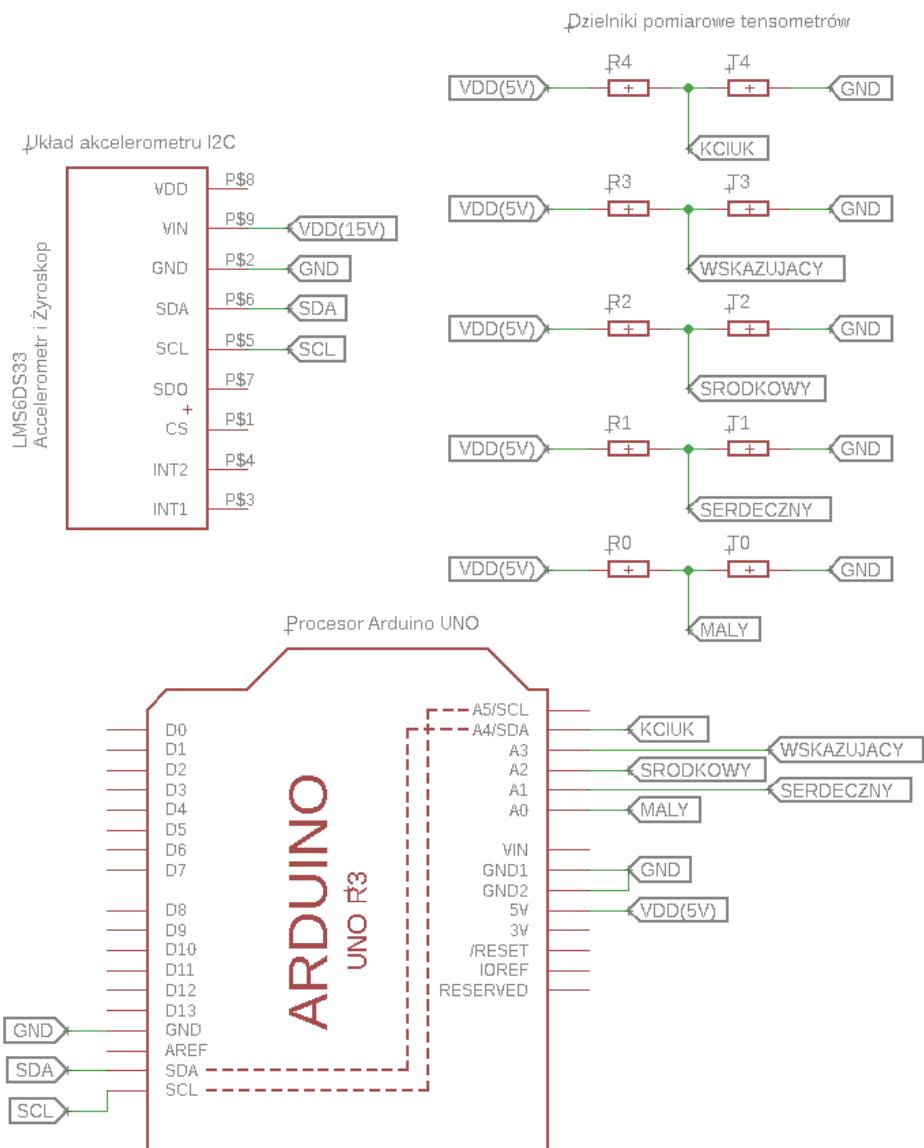


Rysunek 10.5. Dolna strona zaprojektowanej płytki.



Rysunek 10.6. Góra strona zaprojektowanej płytki.

- Schemat połączenia elektrycznego układów pomiarowych w rękawicy Rys. 10.7.



Rysunek 10.7. Schemat połączeń elektrycznych w interaktywnej rękawicy.

- Całkowity kod użyty do sterowania rękawicą Rys. 10.8.

```

// biblioteka I2C
#include<Wire.h>
//biblioteka żyroskop i akcelerometr
#include<LSM6.h>
// zdefiniowanie wejść analogowych
static int flexSensorPin_1 = A0;
static int flexSensorPin_2 = A1;
static int flexSensorPin_3 = A2;
static int flexSensorPin_4 = A3;
static int flexSensorPin_5 = A5;
// zdefiniowanie akcelerometru
LSM6 imu;
char reportAcc[80];
char reportGyro[80];
void setup(){
    // otwarcie komunikacji szeregowej USB
    Serial.begin(9600);
    // otwarcie komunikacji I2C
    Wire.begin();
    // jeśli nie znajdziemy urządzeń LMS6 to wysyłamy komunikat
    if (!imu.init()){
        Serial.println("Failed to detect and initialize IMU!");
        while (1);
    }
    imu.enableDefault();
}
void loop(){
    // uruchomienie funkcji odczytu z czujników
    // w powtarzającej się pętli
    readFlexSensors();
    readAccel();
    // wysłanie danych do komputera
    Serial.println();
    // oczekanie przed kolejnym wysłaniem
    delay(100);
}
void readFlexSensors(){
    // inicjalizacja wejść analogowych jako czujnika ugięcia
    int flexSensorReadingsPinky = analogRead(flexSensorPin_1);
    int flexSensorReadingsRing = analogRead(flexSensorPin_2);
    int flexSensorReadingsMiddle = analogRead(flexSensorPin_3);
    int flexSensorReadingsIndex = analogRead(flexSensorPin_4);
    int flexSensorReadingsThumb = analogRead(flexSensorPin_5);
    // uruchmienie zmennych lokalnych
    // które reprezentują stan logiczny czujników ugięcia
    bool flexSensorReadingsPinkyLogic;
    bool flexSensorReadingsRingLogic;
    bool flexSensorReadingsMiddleLogic;
    bool flexSensorReadingsIndexLogic;
    bool flexSensorReadingsThumbLogic;
    // przetworzenie analogowego czujnika na stan logiczny 1/0
    if(flexSensorReadingsPinky<390){
        flexSensorReadingsPinkyLogic = 1;
    }else{
        flexSensorReadingsPinkyLogic = 0;
    }
    if(flexSensorReadingsRing<350){
        flexSensorReadingsRingLogic = 1;
    }else{
        flexSensorReadingsRingLogic = 0;
    }
    if(flexSensorReadingsMiddle<350){
        flexSensorReadingsMiddleLogic = 1;
    }else{
        flexSensorReadingsMiddleLogic = 0;
    }
    if(flexSensorReadingsIndex<350){
        flexSensorReadingsIndexLogic = 1;
    }else{
        flexSensorReadingsIndexLogic = 0;
    }
    if(flexSensorReadingsThumb<780){
        flexSensorReadingsThumbLogic = 1;
    }else{
        flexSensorReadingsThumbLogic = 0;
    }
}

```

```

// podanie odczytanych stanów na odpowiednie miejsca
// w ramce danych przesyłanych do komputera
Serial.print(flexSensorReadingsPinkyLogic);
Serial.print(",");
Serial.print(flexSensorReadingsRingLogic);
Serial.print(",");
Serial.print(flexSensorReadingsMiddleLogic);
Serial.print(",");
Serial.print(flexSensorReadingsIndexLogic);
Serial.print(",");
Serial.print(flexSensorReadingsThumbLogic);
Serial.print(",");
}
// odczyt danych z akcelerometru przez magistralę I2C
void readAccel(){
    // uruchomienie odczytu z urządzenia LSM6
    imu.read();
    // odczytanie odpowiednich zmiennych z zadanejgo urządzenia
    int pozycjaX = map(imu.a.x, -16900, 16900, -99, 99);
    int pozycjaY = map(imu.a.y, -16900, 16900, -99, 99);
    // zabezpieczenia
    if (pozycjaX > 35 && pozycjaX < 55){
        pozycjaX = 45;
    }
    if (pozycjaY > 35 && pozycjaY < 55){
        pozycjaY = 45;
    }
    if (pozycjaX > 99){
        pozycjaX = 99;
    }elseif (pozycjaX < 0){
        pozycjaX = 0;
    }
    if (pozycjaY > 99){
        pozycjaY = 99;
    }elseif (pozycjaY < 0){
        pozycjaY = 0;
    }
    // podanie zmiennych do ramki danych
    Serial.print(pozycjaX);
    Serial.print(",");
    Serial.print(pozycjaY);
}
// odczyt danych z żyroskopu
// NIE UŻYWANY W TEJ WERSJI STEROWANIA
void readGyros(){
    imu.read();
    sprintf(reportGyro, sizeof(reportGyro), "x:%6d y:%6d z:%6d", imu.g.x, imu.g.y, imu.g.z);
    Serial.print(reportGyro);
    Serial.print(" ");
}

```

Rysunek 10.8. Kod użyty do sterowania rękawicą.