|  |  |  |
| --- | --- | --- |
| Politechnika Koszalińska logo.png | **POLITECHNIKA KOSZALIŃSKA**  **WYDZIAŁ MECHANICZNY**  **KATEDRA MECHATRONIKI I**  **AUTOMATYKI** | D:\Desktop\Logo WM (Corel 9.jpg |

PRACA DYPLOMOWA INŻYNIERSKA

**Projekt i budowa zdalnie sterowanego pojazdu gąsienicowego wykonanego w technologii druku 3D**

**Design and construction of a remotely controlled tracked vehicle made using 3D printing technology**

Franciszek Niedzielski

U-17310

Kierunek: Mechatronika

Specjalność: Systemy monitorowania i sterowania

Promotor: dr. inż. Sebastian Pecolt

Koszalin, 2024

Spis treści

[Oświadczenie 4](#_Toc173931126)

[Streszczenie pracy dyplomowej 5](#_Toc173931127)

[Diploma thesis abstract 6](#_Toc173931128)

[Wykaz symboli i akronimów 7](#_Toc173931129)

[1. Wprowadzenie 8](#_Toc173931130)

[2. WPROWADZENIE TECHNICZNE 10](#_Toc173931131)

[2.1. Komunikacja i zdalne sterowanie 10](#_Toc173931132)

[2.1.1. Protokoły komunikacyjne 10](#_Toc173931133)

[2.1.2. Radio 10](#_Toc173931134)

[2.1.3. Radiowe protokoły komunikacji 10](#_Toc173931135)

[2.2. Silniki napędowe 11](#_Toc173931136)

[2.2.1. Silniki elektryczne 11](#_Toc173931137)

[2.2.2. Rodzaje silników elektrycznych 12](#_Toc173931138)

[2.2.3. Sterowanie silnikami elektrycznymi 13](#_Toc173931139)

[2.2.4. Zastosowanie przekładni mechanicznej 14](#_Toc173931140)

[2.3. Układ bieżny 15](#_Toc173931141)

[2.3.1. Gąsienicowy układ bieżny 15](#_Toc173931142)

[2.3.2. Budowa gąsienicowego układu bieżnego 16](#_Toc173931143)

[2.4. Sterowanie i akwizycja danych 16](#_Toc173931144)

[2.4.1. Mikrokontrolery 16](#_Toc173931145)

[2.4.2. Pytka mikroprocesorowa Arduino 17](#_Toc173931146)

[2.4.3. Mikrokomputer Raspberry Pi 17](#_Toc173931147)

[2.5. Akumulatorowe układy zasilania 18](#_Toc173931148)

[2.5.1. Rodzaje akumulatorów 18](#_Toc173931149)

[2.5.2. Sterowanie ładowaniem i rozładowywaniem 19](#_Toc173931150)

[2.6. Produkcja elementów konstrukcyjnych pojazdu w druku 3D 20](#_Toc173931151)

[2.6.1. Wytwarzanie przyrostowe (metoda addytywna) 20](#_Toc173931152)

[2.6.2. Rodzaje materiałów 21](#_Toc173931153)

[3. cel i zakres pracy 22](#_Toc173931154)

[4. Budowa pojazdu 23](#_Toc173931155)

[4.1. Założenia i wymagania projektowe, dobór komponentów 23](#_Toc173931156)

[4.1.1. Dobór silników napędowych 23](#_Toc173931157)

[4.1.2. Sterowanie silnikami napędowymi 24](#_Toc173931158)

[4.1.3. Dobór sensorów do wykrywania przeszkód 25](#_Toc173931159)

[4.1.4. Dobór sensora pozycji pojazdu 25](#_Toc173931160)

[4.1.5. Układ sterujący i dobór układów logicznych 26](#_Toc173931161)

[4.1.6. Nadajnik-odbiornik FlySky FS-iA6B 29](#_Toc173931162)

[4.1.7. Dobór systemu zasilania - akumulator 30](#_Toc173931163)

[4.1.8. Dobór systemu zasilania – sterowanie 31](#_Toc173931164)

[4.2. Projekt pojazdu w oprogramowaniu CAD 32](#_Toc173931165)

[4.2.1. Główny podział elementów 32](#_Toc173931166)

[4.2.2. Gąsienica i układ bieżny 32](#_Toc173931167)

[4.2.3. Dobór łożysk 32](#_Toc173931168)

[4.2.4. Silniki 32](#_Toc173931169)

[4.2.5. Obudowa i montaż elektroniki 32](#_Toc173931170)

[4.3. Fizyczna realizacja projektu 32](#_Toc173931171)

[4.3.1. Rodzaj produkcji i materiał elementów 32](#_Toc173931172)

[4.3.2. Produkcja elementów bieżnych (metoda addytywna) 33](#_Toc173931173)

[4.3.3. Produkcja elementów przy pomocy wycinarki laserowej 34](#_Toc173931174)

[4.4. Realizacja części logicznej projektu 34](#_Toc173931175)

[4.4.1. Generalny opis działania 34](#_Toc173931176)

[4.4.2. Program sterujący mikrokontrolerem Arduino Nano 36](#_Toc173931177)

[4.4.3. Program sterujący mikrokomputerem Raspberry Pi 37](#_Toc173931178)

[4.4.4. Serwer plików na mikrokomputerze Raspberry Pi 39](#_Toc173931179)

[4.5. Połączenia elektryczne układu 39](#_Toc173931180)

[4.5.1. Układ zasilania 39](#_Toc173931181)

[4.5.2. Generalny układ połączeń 39](#_Toc173931182)

[4.5.3. Użycie płytki drukowanej 39](#_Toc173931183)

[4.6. Badania działania pojazdu 40](#_Toc173931184)

[4.6.1. Badanie prędkości i przewidywanego zasięgu pojazdu 40](#_Toc173931185)

[4.6.2. Badanie funkcji autonomicznych 40](#_Toc173931186)

[5. Wnioski 41](#_Toc173931187)

[Bibliografia 42](#_Toc173931188)

[Spis rysunków 43](#_Toc173931189)

[Spis tabel 44](#_Toc173931190)

Oświadczenie

Załącznik Nr 1 do Regulaminu antyplagiatowego

(Zarządzenie Rektora Nr 24/2014 z 26.06.2014 r.)

**OŚWIADCZENIE**

**(STUDENTA)**

Franciszek Niedzielski

U-17310

Oświadczam, że moja praca pt.: Projekt i budowa zdalnie sterowanego pojazdu gąsienicowego wykonanego w technologii druku 3D:

1. została przygotowana przeze mnie samodzielnie\*,
2. nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (j.t. Dz. U. z 2006 r. Nr 90, poz. 631 z późn. zm.) oraz dóbr osobistych chronionych prawem;
3. nie zawiera danych i informacji, które uzyskałem w sposób niedozwolony;
4. nie była podstawą nadania dyplomu uczelni wyższej lub tytułu zawodowego ani mnie, ani innej osobie.

Ponadto oświadczam, że treść pracy przedstawionej przez mnie do obrony, zawarta na przekazywanym nośniku elektronicznym, jest identyczna z jej wersją drukowaną.

|  |  |
| --- | --- |
| ………………………………… | ……………………………….. |
| data | Podpis studenta |

\*Uwzględniając merytoryczny wkład promotora (w ramach prowadzonego seminarium dyplomowego

Streszczenie pracy dyplomowej

Koszalin, .....201..

**POLITECHNIKA KOSZALIŃSKA**

**WYDZIAŁ MECHANICZNY**

**Katedra/Zakład ................................**

**Tytuł:** Projekt i budowa zdalnie sterowanego pojazdu gąsienicowego wykonanego w technologii druku 3D

**Autor:** Franciszek Niedzielski

**Promotor:** dr inż. Sebastian Pecolt

Celem pracy jest zaprojektowanie i zbudowanie zdalnie sterowanego pojazdu gąsienicowego przy wykorzystaniu technologii druku 3D (metoda przyrostowa). Zakres pracy obejmuje, opracowanie modelu 3D w środowisku CAD, wydruk 3D części mechanicznych, dobór napędów, czujników, układu sterowania, opracowanie schematu elektronicznego, dobranie mikrokontrolera i napisanie aplikacji zarządzającej pracą oraz opracowanie systemu autonomicznego dla projektowanego pojazdu gąsienicowego. Po zbudowaniu pojazdu należy przeprowadzić weryfikację i testy działania.

**Słowa kluczowe: pojazd gąsienicowy, pojazd autonomiczny, CAD, druk 3D**

Diploma thesis abstract

Koszalin, 201..

KOSZALIN UNIVERSITY OF TECHNOLOGY

**FACULTY OF MECHANICAL ENGINEERING**

**Department of ……………**

**Title:** Design and construction of a remotely controlled tracked vehicle made using 3D printing technology

**Author:** Franciszek Niedzielski

**Supervisor:** dr inż. Sebastian Pecolt

The aim of the project is to design and built a remotely controlled tracked vehicle using 3D printing technology (additive manufacturing method). The scope of work includes developing a 3D model in CAD environment, 3D printing of mechanical parts, selection of mechanical drives, sensors, control system, schematic development, selection of a microcontroller, and writing a management application. Additionally, it involves developing an autonomous system for the designed tracked vehicle. After the construction of the vehicle, verification and performance tests are to be conducted.

**Keywords:** **tracked vehicle, autonomous vehicle, CAD, 3D printing**

Wykaz symboli i akronimów

**Symbole**

Vcc – napięcie zasilania

Vc – napięcie stanu wysokiego

Ic – prąd zasilania

**Akronimy**

PWM – modulacja szerokości pulsu

PPM – modulacja pozycji pulsu

N/O – nadajnik-odbiornik

CNC – komputerowe sterowanie urządzeniami numerycznymi

PLA – Polilaktyd

ABS – polimer akrylonitrylo-butadieno-styrenowy

I/O – wejścia i wyjścia (*input/output*)

HMI – interfejs maszyna-człowiek (*human machine inteface*)

RC – sterowanie radiowe (*Radio controlled*)

I2C – szeregowa, dwukierunkowa magistrala danych (*Inter-Integrated Circuit*)

PCB – płytka drukowana (*printed circuit board*)

1. Wprowadzenie

Zdalnie sterowane i autonomiczne pojazdy stają się coraz bardziej powszechne w różnych dziedzinach i pracach. Ostatnio można zaobserwować ich obecność na ulicach miast, gdzie autonomiczne pojazdy kurierskie są powszechnie używane do dostarczania jedzenia z restauracji. Jednak ich rola nie ogranicza się jedynie do usług gastronomicznych.

W obszarze przemysłu militarnego autonomiczne lub częściowo autonomiczne pojazdy mają długą historię zastosowań. W szczególności, są one wykorzystywane jako roboty saperskie, spełniając kluczową funkcję w neutralizacji potencjalnie niebezpiecznych sytuacji. Ich głównym zadaniem jest zbliżanie się do ładunków wybuchowych i transportowanie ich w bezpieczne miejsce. Dla saperów stanowią one bezpieczną alternatywę w porównaniu do ręcznego zbliżania się do potencjalnej bomby lub miny, umożliwiając precyzyjną i bezpieczną interwencję w sytuacjach zagrażających życiu. Autonomiczne pojazdy w służbie wojskowej reprezentują zintegrowane podejście do modernizacji i poprawy efektywności operacji, wprowadzając innowacyjne rozwiązania technologiczne w obszarze bezpieczeństwa.



1. Amerykański robot saperski TALON [5]

Pojazdy tego rodzaju posiadają znaczący potencjał wykorzystania również w innych obszarach, takich jak ratownictwo. Zdalnie sterowany pojazd może pełnić kluczową rolę jako wsparcie przy ewakuacji ludzi z płonącego budynku, zwłaszcza gdy warunki widoczności są znacznie ograniczone. Wykorzystanie zdalnie sterowanego lub autonomicznego pojazdu pozwoliłoby na skuteczne działania w sytuacjach awaryjnych.

Zdalnie sterowany pojazd mógłby zostać wysłany do wnętrza budynku, zbierając informacje dotyczące układu pomieszczeń i oceniając sytuację. Taka mapowanie pięter budynku byłoby niezwykle przydatne dla strażaków, umożliwiając im lepsze zrozumienie terenu przed wejściem do niego. W rezultacie zespół ratowniczy mógłby podejmować bardziej skoordynowane i precyzyjne działania podczas wyprowadzania ofiar pożaru, co zwiększałoby skuteczność operacji ratowniczych.

Zastosowanie zdalnie sterowanych lub autonomicznych pojazdów w obszarze ratownictwa otwiera nowe perspektywy dla poprawy bezpieczeństwa zarówno ratowników, jak i osób potrzebujących pomocy w sytuacjach kryzysowych.

1. WPROWADZENIE TECHNICZNE
   1. Komunikacja i zdalne sterowanie
      1. Protokoły komunikacyjne

Protokół komunikacyjny to zestaw ustalonych zasad i procedur, które regulują wymianę danych między urządzeniami. W celu skutecznej komunikacji, protokół musi być ściśle określony i zrozumiały dla wszystkich połączonych urządzeń. Protokoły cyfrowe operują zwykle na wartościach binarnych, co oznacza, że dane są reprezentowane w postaci bitów o wartości wysokiej (1) lub niskiej (0).

Protokoły komunikacyjne są często obsługiwane przez wyspecjalizowane komponenty lub układy scalone. UART (Universal Asynchronous Receiver/Transmitter), który jest przykładem wyspecjalizowanego układu scalonego często używanego do implementacji protokołów komunikacyjnych. UART jest interfejsem komunikacyjnym umożliwiającym przesyłanie danych między urządzeniami w sposób asynchroniczny, co oznacza, że nie wymaga wspólnej zegarowej synchronizacji między nadawcą a odbiorcą.

Przy projektowaniu systemów komunikacyjnych, wybór odpowiedniego protokołu jest kluczowy dla zapewnienia poprawnej transmisji danych i zrozumienia między urządzeniami. [6]

* + 1. Radio

Zdalne sterowanie opisywanego pojazdu bazuje na komunikacji radiowej. Fale radiowe (inaczej promieniowanie radiowe) to fale promieniowania elektromagnetycznego w odpowiednim zakresie częstotliwościowym. Według definicji amerykańskiej agencji kosmicznej NASA zakres ten zaczyna się na 3 kHz, a kończy na 300 GHz.

Proces komunikacji za pomocą fal radiowych obejmuje generowanie sygnału radiowego przez nadajnik radiowy, który emituje fale elektromagnetyczne w określonym zakresie częstotliwości. Odbiornik radiowy działający w tej samej częstotliwości odbiera te fale, demoduluje je i przetwarza na informacje zrozumiałe dla użytkownika lub układu interpretującego te wiadomości. [6]

* + 1. Radiowe protokoły komunikacji

Radiowe protokoły komunikacyjne są analogiczne do protokołów działających w przewodach, ale różnią się głównie formą przekazywania informacji, co wprowadza dodatkowe wyzwania związane z zakłóceniami i zasięgiem.

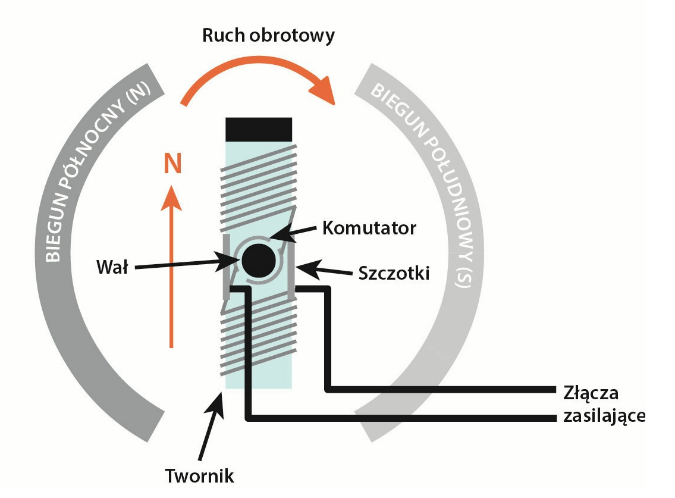
Podstawową różnicą między przewodowymi a radiowymi protokołami komunikacyjnymi jest to, że wartości bitów są przekazywane za pomocą fal radiowych, co sprawia, że są bardziej narażone na zakłócenia. Fale radiowe mogą być zakłócane przez różne czynniki, takie jak interferencje elektromagnetyczne, inne źródła fal radiowych czy obiekty na drodze sygnału, jak budynki, drzewa czy inne przeszkody. Ponadto, fale radiowe tracą moc znacznie szybciej niż impulsy elektryczne w przewodach, co wpływa na zasięg komunikacji.

Mimo tych wyzwań związanych z korzystaniem z takiej formy komunikacji, ma ona zasadniczą zaletę. Nasze urządzenie nie musi być fizycznie połączone z resztą skomunikowanych urządzeń.

* 1. Silniki napędowe
     1. Silniki elektryczne

Silnik elektryczny to urządzenie elektrotechniczne, zamieniające energię prądu elektrycznego na ruch mechaniczny, z reguły o charakterze obrotowym. Istotne parametry silników tego typu to napięcie, moc, sprawność, moment i prędkość obrotowa.

Silniki elektryczne, niezależnie od ich typu, wykorzystują zasadę elektromagnetyzmu jako podstawową siłę napędową. W każdym rodzaju silnika elektrycznego, generacja ruchu obrotowego jest wynikiem interakcji pomiędzy polem elektromagnetycznym a elementami ruchomymi silnika. Silniki te możemy podzielić na:

* obcowzbudne, w których znajdują się magnesy trwałe (lub elektromagnesy o stałej polaryzacji, niepowiązane z twornikiem, spełniające tą samą funkcję),
* samowzbudne, gdzie pole elektromagnetyczne jest wytwarzane przy pomocy elektromagnesów połączonych z twornikiem. Uzwojenie stojana i twornika są zasilane tym samym źródłem,

1. Schemat budowy silnika szczotkowego prądu stałego [7]

W przypadku obcowzbudnego silnika szczotkowego (rys.2) prądu stałego, pole elektromagnetyczne jest tworzone przez uzwojenia w tworniku. W stojanie znajdują się magnesy trwałe oddziałujące z uzwojeniami twornika. Zmiana polaryzacji pola elektromagnetycznego, niezbędna do utrzymania ruchu obrotowego wirnika, jest realizowana za pomocą komutatora. Komutator pełni funkcję mechanicznego przełącznika, który zmienia kierunek przepływu prądu w uzwojeniach twornika w odpowiednim momencie obrotu wirnika. [1][3][7]

* + 1. Rodzaje silników elektrycznych

Silniki elektryczne możemy również podzielić na zasilane prądem stałym DC i zasilane prądem przemiennym AC. Jako że zasilanie projektowanego układu bazuje na napięciu stałym, silniki zasilane prądem zmiennym zostały pominięte.

Główne rodzaje silników rozważanych w projekcie to:

* Silniki szczotkowe prądu stałego – ich główną zaletą jest niska cena i łatwość implementacji. Niestety przy wyższych prędkościach obrotowych, silniki szczotkowe mają tendencję do przegrzewania, ze względu na tarcie szczotek komutatora, przez co zakres prędkości obrotowych pracy silnika jest ograniczony. Szczotki wymagają również cyklicznej konserwacji lub wymiany,
* Silniki bezszczotkowe prądu stałego – są droższe niż silniki szczotkowe, ale cechują się większą niezawodnością, dokładnością, żywotnością i zakresem prędkości pracy. Istnieje również możliwość sterowania momentem obrotowym ze zmienną prędkością obrotową. Posiadają średnio większą moc od silników szczotkowych podobnych gabarytów. Główną wadą silników bezszczotkowych jest potrzeba zastosowania specjalnego układu sterującego, co również zwiększa koszt implementacji.
* Silniki krokowe (skokowe) – silniki tego typu różnią się znacznie w budowie od wyżej wymienionych. Są głównie stosowane przy sterowaniu precyzyjnym (np. we frezarkach CNC czy drukarkach 3d). Obrót rotora w tego typu silnikach odbywa się poprzez przeskoki o stały ustalony kąt. Silniki krokowe możemy też podzielić na unipolarne i bipolarne. Silniki bipolarne posiadają dodatkowe wejście dla uzwojenia, które dzieli to uzwojenie na dwie części. Pozwala to na dokładniejsze sterowanie silnika.

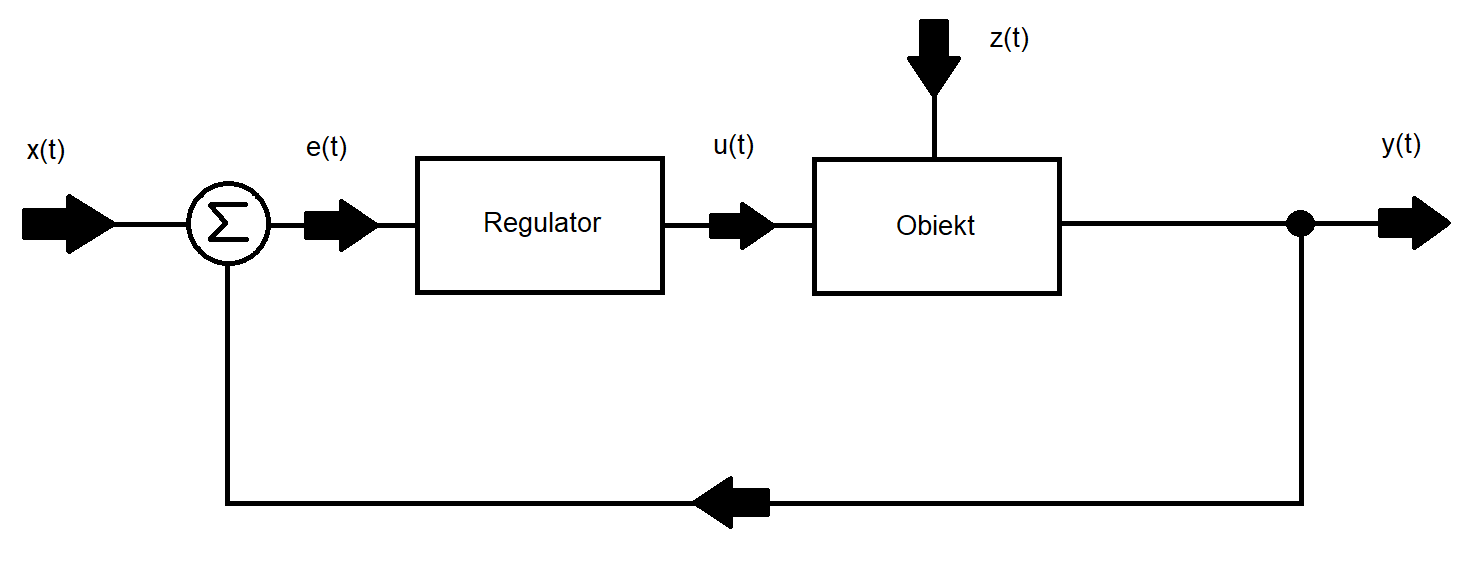
[1][3][7]

* + 1. Sterowanie silnikami elektrycznymi

Rodzaj sterowania silnikiem jest uwarunkowane rodzajem silnika i jego zastosowaniem. Celem sterowania jest uzyskanie żądanej prędkości obrotowej, momentu obrotowego i kierunku obrotu. Każdy rodzaj silnika wymaga innego, wyspecjalizowanego układu:

* Sterowanie silnikiem szczotkowym DC – zazwyczaj jest realizowane przez modulację średniej mocy za pomocą sygnału PWM, sterującym przełącznikiem mocy. Zamiana kierunków jest realizowana poprzez zamianę kierunku przepływu prądu przez silnik,
* Sterowanie silnikiem bez szczotkowym DC – wymaga dedykowanego sterownika, którego funkcją jest załączanie i rozłączanie wejść silnika, w odpowiedniej kolejności i z odpowiednią częstotliwością, aby indukować poszczególnie cewki silnika, co powoduje obrót rotora. Niektóre silniki posiadają czujniki pozycji rotora aby usprawnić sterowanie, poprzez zamknięcie systemu sterowania.
* Sterowanie silnikiem krokowym – może być wykonywany na kilka sposobów, pełnokrokowo i półkrokowo. Pełnokrokowo oznacza, że rotor przy zmianie polaryzacji uzwojenia przemieszcza się od jednego uzwojenia do drugiego. Półkroki natomiast są realizowane przez załączenie dwóch cewek na raz, aby rotor znalazł się w położeniu między dwoma pełnymi krokami. Sterowanie silnikiem jest przeważnie wykonywane przez specjalny układ elektryczny, który przyjmuje dwa sygnały: pulsy i kierunek (sygnały prostokątne). Sygnał kierunkowy, zmieniając wartość, wyznacza czy rotor będzie poruszał się zgodnie czy przeciwnie względem wskazówek zegara. Sygnał puls natomiast to kroki (lub półkroki) wykonywane przez silnik. Im większa częstotliwość pulsów tym większa prędkość obrotu.

Sterowanie silnikami różni się również w zależności od użytego silnika, lecz można wyróżnić dwa główne sposoby sterowania – bez sprzężenia zwrotnego i ze sprzężeniem zwrotnym (regulacja).



1. Układ sprzężenia zwrotnego – regulacji [opracowanie własne]

Układ z regulacją posiada funkcję korekcji błędu, co znacznie zwiększa dokładność sterowania. Niweluje to również wpływ zakłóceń na układ. Główną wadą tego rozwiązania jest zwiększenie złożoności (przez co i kosztu wykonania) urządzenia, ale i również wrażliwość utraty stabilności układu. W przypadku utraty stabilności układ wpada w niekontrolowane oscylacje, które mogą skończyć się uszkodzeniem urządzenia.

Natomiast w przypadku sterowania bez sprzężenia zwrotnego układ jest znacznie prostszy. Nie ma również ryzyka utraty stabilności, ale nie ma też informacji zwrotnej o położeniu lub stanie obiektu. []

* + 1. Zastosowanie przekładni mechanicznej

Przekładnia mechaniczna to mechanizm przenoszący ruch z jednego urządzenia na drugie, zmieniając przy tym jego parametry takie jak prędkość obrotowa czy moment obrotowy. Do typowych przekładni mechanicznych należą:

* przekładnie cięgnowe,
* przekładnie zębate,
* przekładnie cierne.

Stale zazębione przekładnie o przełożeniu redukującym prędkość obrotową prowadzące moment obrotowy do systemu bieżnego (np. kół samochodu) nazywamy przekładniami głównymi. Przekładnie główne spełniają też często funkcję zamiany kierunku obrotu, lecz w przypadku budowanego pojazdu, funkcja ta będzie realizowana przez sterownik silników. [2]

* 1. Układ bieżny
     1. Gąsienicowy układ bieżny

Pierwszy układ gąsienicowy przeznaczony do napędu pojazdów został zaprojektowany przez polskiego naukowca Józefa Marię Hoene-Wrońskiego w latach 30stych XIX wieku. Miał być to wtedy projekt konkurujący z ówczesną koleją. Rozwój gąsienicowych układów bieżnych był od tego czasu rozwijany przez różne osoby i organizacje. Obecnie pojazdy gąsienicowe znajdują szerokie zastosowanie, np. w przemyśle budowlanym czy militarnym.

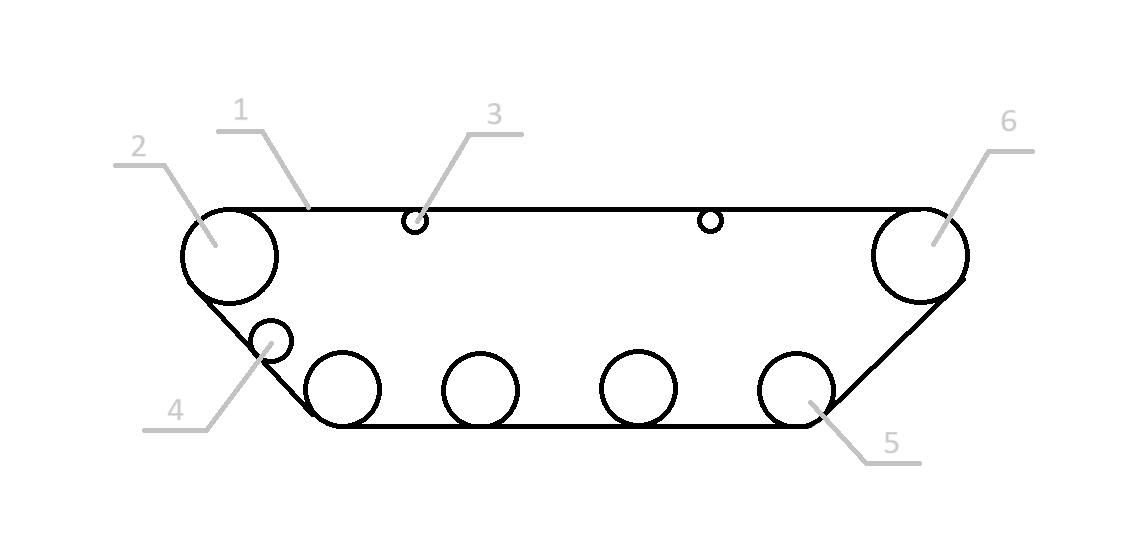
Zalety gąsienicowego układu bieżnego:

* równomierne rozłożenie ciężaru - Konstrukcja gąsienic pozwala na rozłożenie ciężaru pojazdu na większej powierzchni, minimalizując nacisk na podłoże. Jest to korzystne, zwłaszcza w warunkach, gdzie koła mogą mieć trudności, takich jak miękkie lub niestabilne tereny.
* lepsza manewrowość w trudnym terenie - Gąsienice umożliwiają efektywniejsze poruszanie się po nierównym i trudnym terenie, co jest szczególnie ważne w zastosowaniach wojskowych i inżynieryjnych.
* zwiększona stabilność - Dzięki gąsienicom pojazdy mogą lepiej utrzymywać stabilność, co jest istotne przy przewożeniu ciężkich ładunków lub wykonywaniu precyzyjnych operacji.

Wady gąsienicowego układu bieżnego:

* kompleksowość konstrukcji i utrzymania - Układy te wymagają bardziej skomplikowanego projektowania i regularnego konserwowania, co prowadzi do zwiększenia kosztów operacyjnych.
* większa masa i hałas: Gąsienice zwiększają masę pojazdu oraz generują więcej hałasu podczas jazdy, co może być problematyczne w pewnych środowiskach.
* ograniczenia prędkości i efektywności paliwowej: Układy gąsienicowe zwykle oferują niższe prędkości maksymalne oraz posiadają mniejszą sprawność względem układów kołowych.

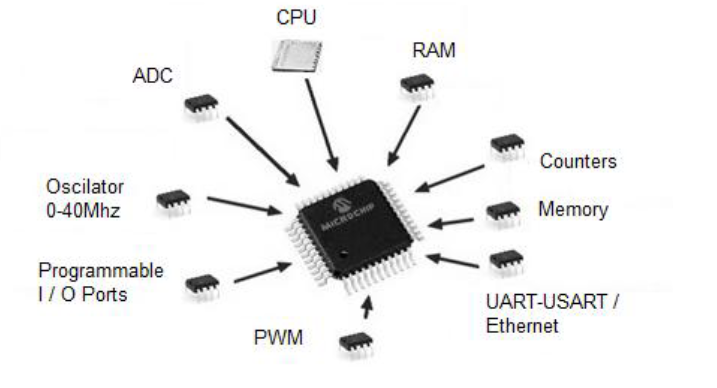
[2][4]

* + 1.  Budowa gąsienicowego układu bieżnego

1. Schemat budowy typowego gąsienicowego układu bieżnego [opracowanie własne]

Gąsienicowy układ bieżny składa się z kół napędowych (2), kół kierunkowych (6), kół nośnych (5), kół napinających (4) i gąsienicy (1). Gąsienica to łańcuch składający się z pojedynczych ogniw, które mają kontakt z podłożem. [4]

* 1. Sterowanie i akwizycja danych
     1. Mikrokontrolery

Mikrokontroler to układ scalony, posiadający procesor, pamięć operacyjną i układ wejść/wyjść. Dzisiejsze mikrokontrolery posiadają często również podzespoły służące do komunikacji i programowania, przetworniki analogowo-cyfrowe, generatory sygnałów (np. PWM). Są one wykorzystywane w szerokim spektrum zastosowań. Mogą służyć do sterowania i kontroli procesów produkcyjnych, sterowania urządzeniami AGD czy nawet jako proste komputery.

1. Układ nowoczesnego mikrokontrolera i jego komponenty

Mikrokontrolery zaczęły być używane na masową skalę w latach 70. Pierwsze układy były 4-bitowe, ale szybko pojawiły się układy 8-bitowe. Obecnie większość mikrokontrolerów dla prostych zastosowań operuje na 8 bitach, ale znane są również układy 16-, 32- i 64-bitowe. [10]

* + 1. Pytka mikroprocesorowa Arduino

Rodzina płytek mikroprocesorowych Arduino nazywanych potocznie mikrokontrolerami Arduino powstała we Włoszech w 2009 roku. Pierwsza płytka została zaprojektowana z myślą o nauczaniu programowania, elektroniki i automatyki. Autorzy płytki zostali wtedy nazwani przez prasę branżową „techno-hipisami” w związku z otwartą polityką oprogramowania (z ang. *open-source*). Z pierwszej, podstawowej płytki Arduino UNO zrodziły się konstrukcje o podobnej budowie ale innym zastosowaniu, mniejsze lub większe, z wbudowanymi modułami lub z redukcją elementów celem niższego poboru energii. Wszystkie płytki cechują się jednak użyciem tej samej infrastruktury programowania z wykorzystaniem mikrokontrolerów AVR. [11].

Każda płytka Arduino składa się z mikroprocesora, układu zasilania, wejść/wyjść cyfrowych, wejścia programatora (najczęściej w postaci portu USB, lecz mniejsze modele wymagają dedykowanych programatorów), przetworników ADC, generatorów sygnału PWM i diod LED sygnalizujących obecny stan połączenia szeregowego. [12]

Płytki Arduino są programowane poprzez wbudowany lub oddzielny programator, łączący się z jednostką PC poprzez port COM, najczęściej przez połączenie USB. Dedykowanym środowiskiem programistycznym jest oprogramowanie Arduino IDE, gdzie użytkownik może napisać program dla mikroprocesora, pobrać biblioteki, wybrać port COM oraz wybrać model programowanej płytki. Wybór odpowiedniego portu i modelu Arduino jest kluczowy do połączenia.

Program w Arduino IDE pisany jest w języku programowania wywodzącym się, i silnie bazującym na językach C i C++. Języki te zostały zmodyfikowane celem stworzenia języka bardziej przystępnego do formy płytki mikroprocesorowej.

* + 1. Mikrokomputer Raspberry Pi

Płytki z serii Raspberry Pi to w pełni funkcjonalne komputery zamknięte w formie pojedynczej płytki PCB. Podobnie do płytek Arduino, są to urządzenia *open-source*. Umożliwiają one korzystanie z tych samych funkcji i programów (z pewnymi wyjątkami) co tradycyjne komputery PC, choć mają ograniczoną moc obliczeniową. Mikrokomputery Raspberry Pi posiadają system operacyjny, wielordzeniowy procesor, pamięć RAM i flash jak i szereg portów zapożyczonych z komputerów PC. Płytki posiadają również wyjścia GPIO, umożliwiające komunikację w protokołach SPI i I2C, sterowanie prostymi urządzeniami cyfrowymi takimi jak sensory czy diody LED, oraz odczytywanie z nich danych w postaci sygnałów cyfrowych. Piny GPIO nie posiadają przetworników ADC, w przeciwieństwie do podobnych urządzeń. Wiele modeli Raspberry Pi posiada również porty DSI (szeregowy interfejs wyświetlacza) co umożliwia podłączanie wyświetlaczy bezpośrednio do płytki, z pominięciem portów HDMI i DisplayPort.

Na mikrokomputerach Raspberry Pi funkcjonuje dedykowany system operacyjny Raspbian. Jest to system bazujący na systemie Linux Debian, przystosowany do infrastruktury procesorów ARM, znajdujących się na płytkach Raspberry Pi.

* 1. Akumulatorowe układy zasilania
     1. Rodzaje akumulatorów

Głównymi rodzajami akumulatorów są:

Akumulatory kwasowo-ołowiowe, to najstarsze i prawdopodobnie najbardziej rozpowszechnione rodzaje akumulatorów w pojazdach spalinowych. Cechuje je duża masa w stosunku do utrzymywanego ładunku i możliwość zasilania dużymi prądami. Niestety ładowanie ich dużymi prądami prowadzi do szybszego zużycia. Napięcie znamionowe pojedynczego ogniwa to 2.1-2.15 V.

Akumulatory zasadowe niklowo-kadmowe (NiCd), to mniej popularny rodzaj akumulatorów, stosowany głównie w urządzeniach przenośnych o dużym prądzie zasilania. Główną wadą tego typu akumulatorów jest efekt pamięciowy. Cechuje się to zmniejszeniem pojemności akumulatora w przypadku, gdy jest on ładowny zanim został w pełni rozładowany (gdy ogniwo posiada napięcie 0.9-1 V). Napięcie znamionowe ogniwa to 1.2 V.

Akumulatory zasadowe niklowo-wodorkowe (NiMH). Są podobne do akumulatorów niklowo-kadmowych, posiadają to samo napięcie znamionowe 1.2 V. Posiadają natomiast większą gęstość energetyczną i większą pojemność. Ich główne wady to mniejsza trwałość i szybsze samorozładowanie w porównaniu do niklowo-kadmowych. Nie występuje w nich natomiast efekt pamięciowy.

Akumulatory litowo-jonowe (Li-Ion) posiadają największe z wymienionych akumulatorów napięcie znamionowe, wynoszące 3.6 V. Brak występowania efektu pamięciowego. Główną wadą tego typu akumulatorów jest ich łatwopalność i ryzyko samozapłonu w przypadku przegrzania. Może to nastąpić podczas ładowania lub podczas pobierania zbyt dużego prądu z akumulatora. Rozszczelnienie obudowy może również prowadzić do samozapłonu. Akumulatory tego typu wymagają korzystania ze specjalnego układu sterującego ładowaniem i rozładowywaniem.

Akumulatory litowo-polimerowe (Li-Po) działają podobnie do litowo-jonowych, lecz różnią się postacią elektrolitu. Elektrolit Li-Po to polimer w postaci częściowo stałej (żelowej). Użycie elektrolitu w takiej postaci sprawia, że akumulator jest bardziej stabilny i mniej wrażliwy na uszkodzenie mechaniczne. Napięcie znamionowe to 3.7 V.

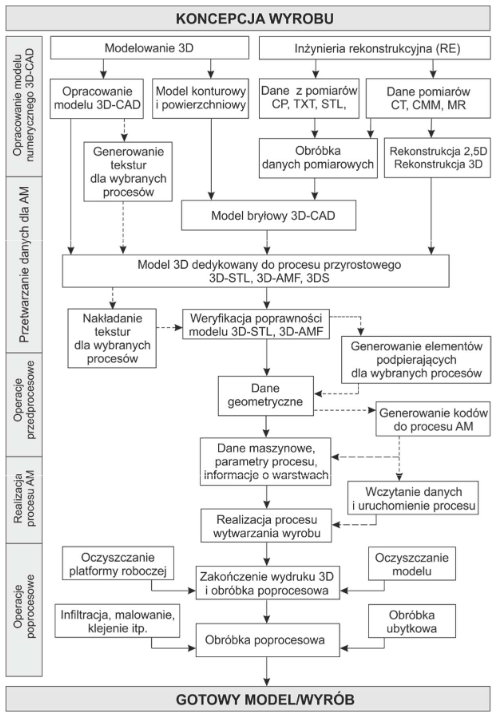
[1][8]

* + 1. Sterowanie ładowaniem i rozładowywaniem

W nowoczesnych aplikacjach wykorzystujących akumulatory, kluczową rolę odgrywają układy sterujące procesami ładowania i rozładowywania. Te systemy są zintegrowane z akumulatorami w celu optymalizacji ich pracy i zwiększenia żywotności. Szczególnie w przypadku baterii litowo-jonowych (Li-Ion) i litowo-polimerowych (Li-Po), obecność inteligentnych układów sterowania jest nie tylko pożądana, ale często konieczna z uwagi na ich delikatną naturę i potencjalne ryzyko uszkodzeń.

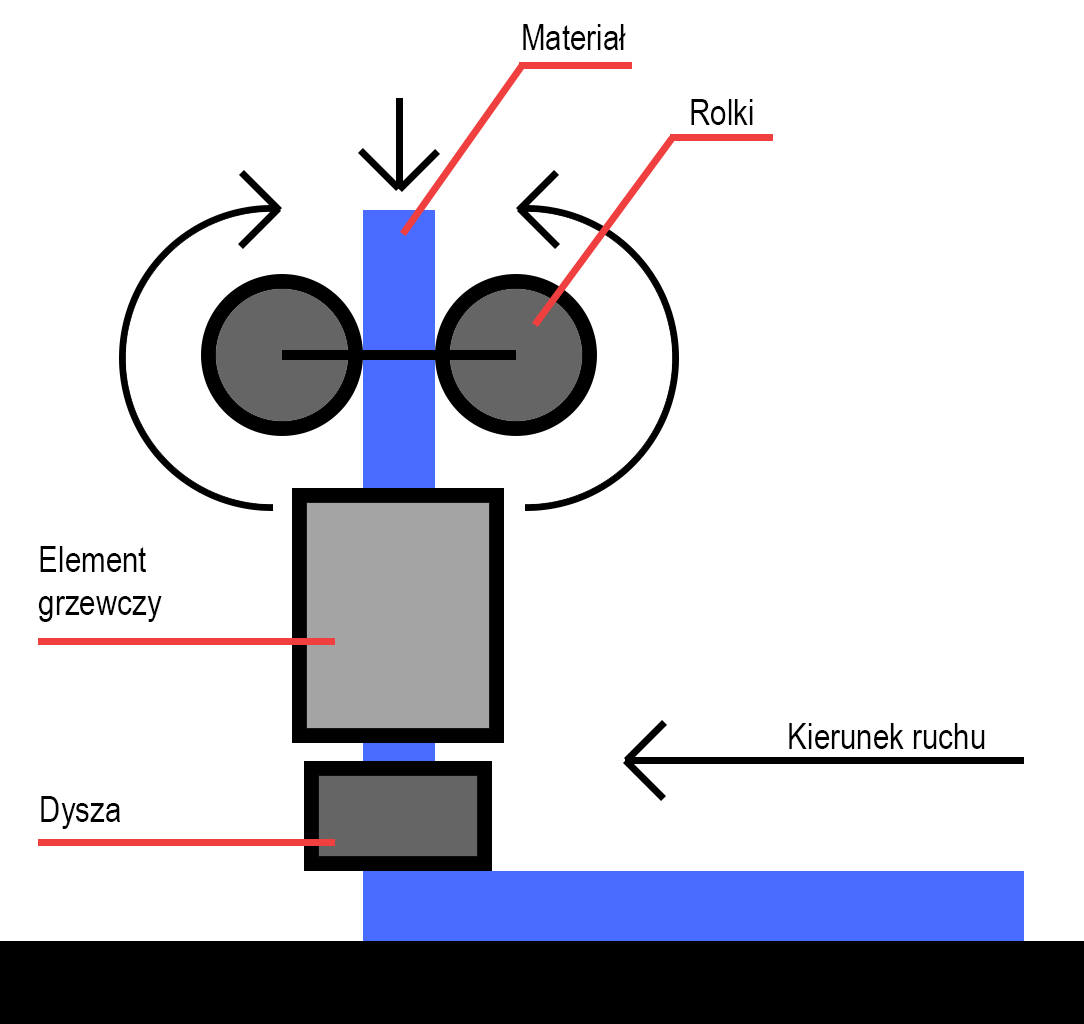
Głównym celem tych systemów jest zapewnienie równowagi pomiędzy maksymalizacją wydajności baterii a jej bezpieczeństwem. Osiąga się to przez monitorowanie i regulację kluczowych parametrów, takich jak napięcie, prąd, temperatura oraz poziom naładowania baterii. [1]

* 1. Produkcja elementów konstrukcyjnych pojazdu w druku 3D
     1. Wytwarzanie przyrostowe (metoda addytywna)

Wytwarzanie przyrostowe to jedna z głównych metod drukowania w trzech wymiarach. Polega na wyprodukowaniu rzeczywistego, fizycznego kształtu elementu, na podstawie modelu 3D. Cały proces produkcji wytwarzania przyrostowego składa się z wielu etapów, tak jak przedstawiono na rys. 6.

1. Schemat procesu wytwarzania przyrostowego [9]

Sam etap wytwarzania fizycznego obiektu odbywa się za pomocą maszyny przyrostowej (drukarki 3D). Maszyna za pomocą ekstrudera nakłada pojedyncze warstwy materiału na stół roboczy. Ekstruder składa się ze specjalnej dyszy, nagrzewanej do temperatury topnienia używanego materiału, który natomiast jest podawany do niej za pomocą rolek. Prędkość obrotu rolek jest ściśle sterowana przez układ sterujący maszyną. Jest ona zmienna w zależności od zapotrzebowania, im większa prędkość, tym więcej materiału jest wyprowadzane na stół roboczy.

Kształt elementu powstaje dzięki poruszaniu się ekstudera nad stołem roboczym. Nakładana jest w ten sposób każda kolejna warstwa elementu. Jest to realizowane najczęściej za pomocą plotera 3D, który manipuluje ekstruderem w przestrzeni trójwymiarowej. Nie jest to natomiast regułą, wiele maszyn przyrostowych używa również ramion roboczych. Zwiększa to znacznie szybkość produkcji, ale również zwiększa cenę takiej maszyny.

1. Schemat działania ekstrudera maszyny przyrostowej [opracowanie własne]
   * 1. Rodzaje materiałów

Najczęściej stosowanymi rodzajami materiałów są polimery topliwe. Najczęściej wykorzystuje się polimery plastikowe Polilaktyd (PLA) i akrylonitrylo-butadieno-styrenowy (ABS). Dobór materiału jest kluczowy, aby produkowana część posiadała oczekiwane właściwości fizyko-chemiczne.

Metoda przyrostowa może być też wykorzystywana z materiałami metalowymi. Potrzebna jest wtedy znacznie większa temperatura, alby przygotować materiał do nałożenia na stół roboczy.

1. cel i zakres pracy

Celem niniejszej pracy jest projekt i budowa zdalnie sterowanego pojazdu gąsienicowego, wykonanego w konstrukcji druku 3D. Pojazd ma być również w stanie nawigować zamknięty teren (np. piętro budynku) bez pomocy operatora.

Do zdalnego sterowania układu zostanie wykorzystany nadajnik-odbiornik (z ang. transceiver) FlySky FS-IA6B, działający na częstotliwości 2,4 GHz. Natomiast system autonomiczny będzie realizowany za pomocą mikrokomputera RaspberryPi. Do komunikacji z urządzeniem FlySky i RaspberryPi, jak i do sterowaniem napędami pojazdu zostanie wykorzystana płytka mikrokontrolerowa Arduino.

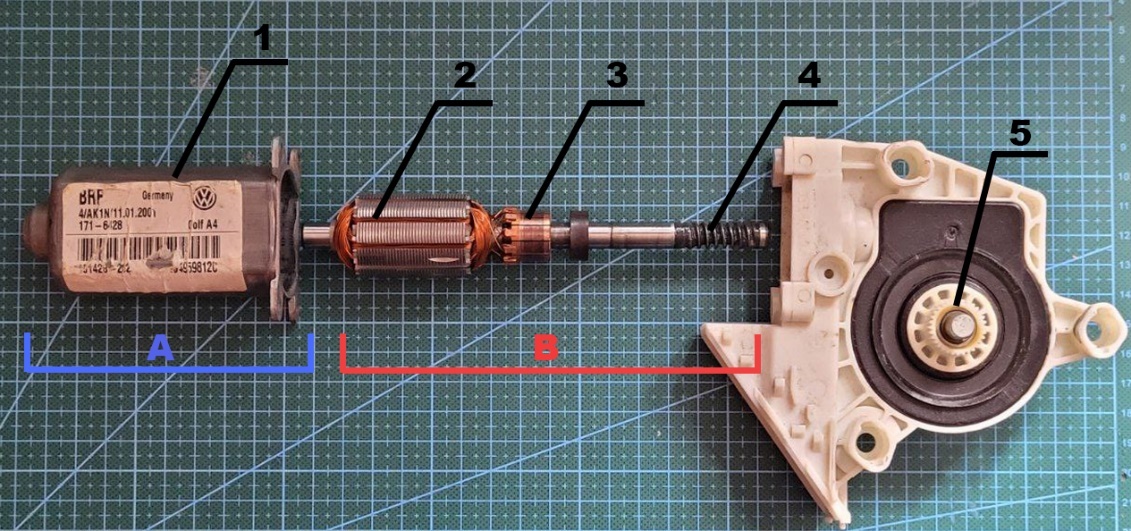
Do zakresu pracy należy:

* projekt pojazdu gąsienicowego, działającego dobrze warunkach terenowych i miejskich, w oprogramowaniu CAD
* przygotowanie projektu do druku 3D
* opracowanie systemu komunikacyjnego nadajniko-odbiornika z resztą układu
* przygotowanie programu autonomicznego sterowania pojazdem
* dobór mikrokontrolera, silników (napędowych i sterujących), układu zasilania, czujników i odpowiednich standaryzowanych elementów mechanicznych
* sporządzenie schematu elektrycznego całego układu

1. Budowa pojazdu
   1. Założenia i wymagania projektowe, dobór komponentów

Celem projektu jest projekt i budowa zdalnie sterowanego pojazdu gąsienicowego, z funkcją autonomicznego mapowania pomieszczeń (labiryntów).

* + 1. Dobór silników napędowych

Do napędzania pojazdu zostały wybrane silniki DC z wbudowaną przekładnią ślimakową. Oryginalnym zastosowaniem zestawu było podnoszenie i opuszczanie szyby w samochodach Golf IV Volkswagen o numerach katalogowych (prawy i lewy) 1J4959811C i 1J4959812C.

1. Rozebrany silnik DC używany w pracy, wraz z opisem elementów [opracowanie własne]

Powyżej przedstawiono budowę silnika z przekładnią wykorzystywanego w projekcie. Literą A oznaczono stojan silnika, natomiast literą B rotor. Numerami oznaczono następujące elementy: 1 – osłona silnika z magnesami trwałymi; 2 – twornik silnika z cewkami uzwojenia; 3 – komutator silnika; 4 – ślimak przekładni, 5 – wał wyjściowy silnika.

Silniki nie posiadają oficjalnej dokumentacji, więc należy wykonać badania poboru prądu i prędkości obrotowej przy stałym napięciu i obciążeniu. Jako napięcie zasilania przyjęto 12 volt i silnik został obciążony momentem 0.15 Nm, za pomocą odważnika o wadze 1.5 kg na kole o promieniu 10 cm.

Pomiar poboru prądu został wykonany amperometrem cyfrowym o błędzie pomiarowym ± 5 %. Wartość średnia 10 pomiarów wynosi 2,55 amerów. Dokładność pomiaru wynosi:

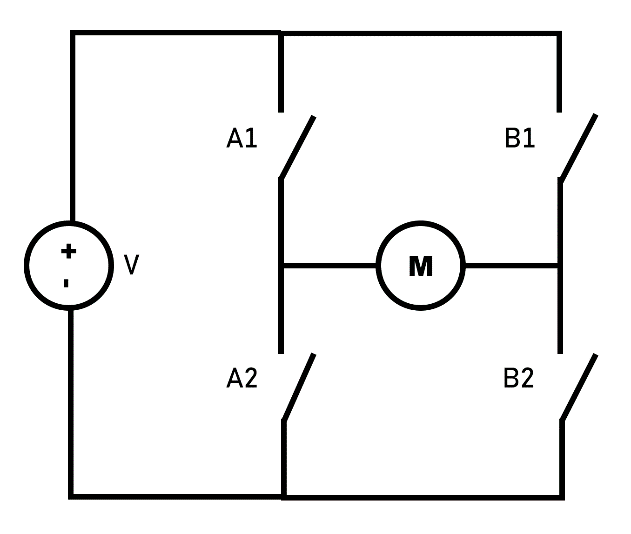
(1)

Możemy zatem przyjąć, że średni pobór prądu wynosi 2,55 ± 0,13 A.

Pomiar prędkości obrotowej został wykonany przy pomocy stopera. W ciągu 60 sekund silnik wykonał 75 pełnych obrotów.

* + 1. Sterowanie silnikami napędowymi

Do sterowania silnikami został zastosowany jednokanałowy moduł sterownika IBT\_2, przeznaczony do sterowania silnikami DC. Układ działa na bazie dwóch pół-mostków BTS7960. Dwa pół-mostki tworzą układ analogiczny do pełnego mostka H, co pozwala na kontrolę prędkością i kierunkiem obrotu silnika. Moduł nie posiada aktywnego chłodzenia, jedynie chłodzenie pasywne w postaci radiatora. W projekcie użyte zostaną dwa sterowniki, po jednym na sterowany silnik.



1. Schemat logiczny mostka H, lub dwóch pół-mostków [opracowanie własne]
2. Parametry modułu IBT\_2 - sterownik silników DC

|  |  |  |  |
| --- | --- | --- | --- |
| **Symbol** | **Parametr** | **Wartość** | **Jednostka** |
| Vcc | Napięcie zasilania sterownika | 5 | V |
| Vm | Napięcie zasilania silnika | 5 - 27 | V |
| Vc | Napięcie logiczne (stan wysoki) | 5 | V |
| Imax | Maksymalny prąd wyjściowy | 30 | A |
| f | Maksymalna częstotliwość sterowania | 25 | kHz |

Sterownik IBT\_2 został wybrany z uwagi na jego zdolność do obsługi wysokiego prądu sterowania silnikami, który znacznie przewyższa zapotrzebowanie prądowe zastosowanych silników. Zapewni to niskie zużycie sterowników i zapobiegnie ewentualnemu przegrzewaniu, ze względu na mniejsze obciążenie. Maksymalna częstotliwość sterowania jest również większa niż sterowana przez wybrany mikrokontroler, co zapewni bezbłędne sterowanie sygnałem PWM.

* + 1. Dobór sensorów do wykrywania przeszkód

Pojazd powinien być w stanie wykrywać podstawowe przeszkody na swojej drodze jak i określić odległość od nich. Powinien również być w stanie określić odległość od ściany po jego prawej stronie i ustawić się do niej prostopadle. Możemy wyznaczyć wymagania:

* sensor powinien działać w warunkach utrudnionej widoczności,
* sensor powinien być w stanie wykryć przeszkody na trasie pojazdu,
* sensor powinien być w stanie określić odległość od przeszkody z rozdzielczością minimalną 10 mm,
* sensor powinien mieć zakres pomiarowy co najmniej 1 m, aby pojazd miał wystarczający czas do reakcji na przeszkodę,
* minimalna częstotliwość pomiaru 500 Hz,
* możliwie niska cena modułu.

Uwzględniając te wymagania, można wyróżnić sensor HC-SR04. Jest to odległościowy czujnik ultradźwiękowy, badający odległość pośrednio, mierząc czas od wysłania impulsu dźwiękowego do jego powrotu. Impuls dźwiękowy odbija się od przeszkody i powraca do sensora. Znając prędkość dźwięku w ośrodku (powietrze – 340 m/s) można wyznaczyć odległość sensora od obiektu.

1. Parametry sensora HC-SR04

|  |  |  |  |
| --- | --- | --- | --- |
| **Symbol** | **Parametr** | **Wartość** | **Jednostka** |
| Vcc | Napięcie zasilania | 5 | V |
| Vc | Napięcie logiczne (stan wysoki) | 5 | V |
| I | Pobór prądu | 15 | mA |
| Smin | Minimalna odległość pomiaru | 20 | mm |
| Smax | Maksymalna odległość pomiaru | 2000 | mm |
| fi | Częstotliwość impulsu dźwiękowego | 40 | kHz |
| ti | Długość pojedynczego impulsu | 10 - zmienny\* | µs |

\*-zależny od układu sterującego. Długość impulsu jest zależna od czasu w jakim pin Trig znajduje się w stanie wysokim. Zalecana wartość to 10 µs.

Odległość od przeszkody zostanie wyznaczona za pomocą wzoru:

(2)

Gdzie S to odległość w od badanego obiektu [m], ti to długość impulsu [s], vs to prędkość dźwięku w powietrzu [m/s].

* + 1. Dobór sensora pozycji pojazdu

Pojazd powinien mieć funkcjonalność określenia obecnego kierunku jazdy w trakcie działania. Jest to potrzebne ze względu na możliwie zróżnicowany teren, po którym będzie się przemieszczał i wykonywał manewr skręcania (obrotu). Przy otwartym systemie sterowania, bez informacji zwrotnej o pozycji, nie można dokładnie wysterować obrotu pojazdu. Dla przykładu, zakładając że pojazd potrzebuje 5 s na obrót na podłożu betonowym, nie oznacza że ta sama operacja zajmie taki sam czas na żwirze lub piasku. Nie można zastosować sterowania czasowego. Należy wykorzystać zamknięty system sterowania, gdzie informacją zwrotną, będzie obecny kierunek zwrotu pojazdu.

Wymagania stawiane sensorowi:

* pomiar sensora nie powinien być zakłócany przez działanie pojazdu,
* pomiar powinien być wykonywany z minimalną dokładnością 1° obrotu
* częstotliwość pomiaru nie powinna być mniejsza niż 5 Hz
* możliwie niska cena sensora.

Uwzględniając te wymagania do pomiaru kierunku zwrotu pojazdu został wybrany moduł kompasu – magnetometr QMC5883L. Czujnik może wykonywać pomiar natężenia pola magnetycznego w 3 osiach, co pozwala na wyznaczenie pozycji czujnika (i analogicznie pojazdu). Czujnik wykorzystuje protokół komunikacyjny I2C, co pozwala na bardzo szybką wymianę danych, znacznie większą niż wymagana.

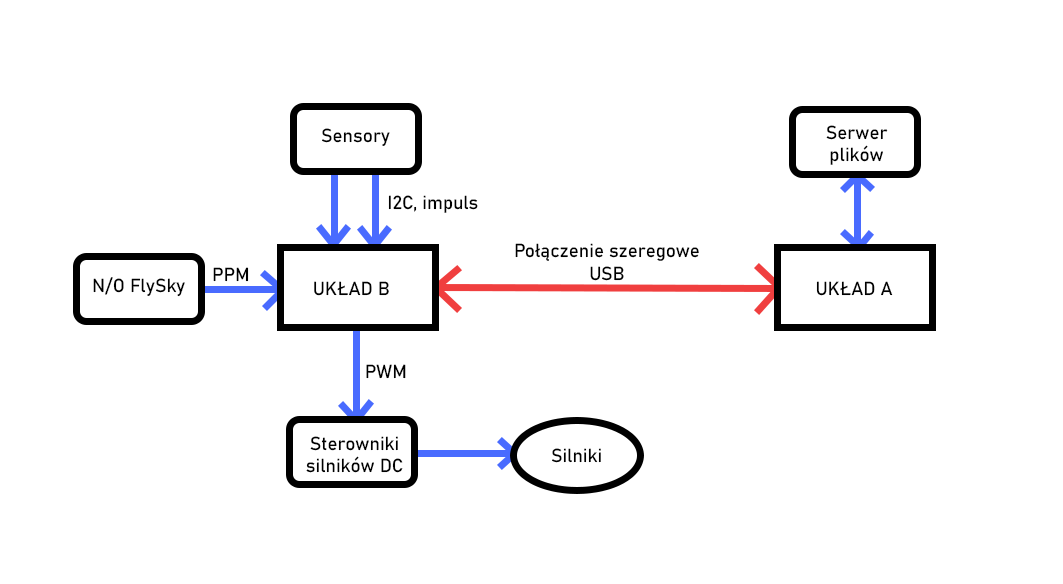
1. Parametry magnetometru QMC5883L

|  |  |  |  |
| --- | --- | --- | --- |
| **Symbol** | **Parametr** | **Wartość** | **Jednostka** |
| Vcc | Napięcie zasilania | 3.3-5 | V |
| Vc | Napięcie logiczne | takie jak Vcc | V |
| I | Pobór prądu | < 1 | mA |
| x | Rozdzielczość pomiaru | 12 | bit |
| f | Częstotliwość pomiaru | 75-160 | Hz |
| - | Wymiary | 18 x 18 | mm |

* + 1. Układ sterujący i dobór układów logicznych

Układ sterujący będzie się składał z dwóch osobnych układów logicznych, komunikujących się ze sobą za pomocą magistrali szeregowej. Jeden z układów (układ B) ma za zadanie kontroli sygnałów I/O (silniki, sensory itp.) i podejmowaniem decyzji w trybie autonomicznym, natomiast drugi układ ma zajmować się zapisywaniem stanu maszyny w czasie i przekazywaniem tych informacji operatorowi.

Układ B powinien być w stanie komunikować się z drugim kontrolerem za pomocą połączenia szeregowego. Równolegle powinien generować i odbierać sygnały PWM i impulsowe, aby kontrolować silniki, odczytywać informację z N/O oraz sensorów.

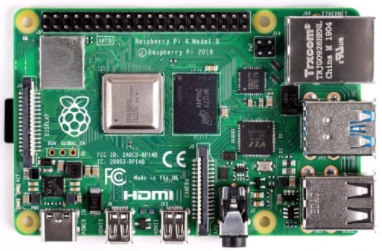
Z uwzględnieniem tych wymagań jako układ B została wybrana płytka mikroprocesorowa Arduino Nano bazująca na mikrokontrolerze ATmega328. Działa w oparciu o napięcie 5 V (stan wysoki), może być zasilana portem USB, posiada 2 terminale szeregowe i 20 piny I/O (w tym 8 wbudowanych przetworników ADC).

1. Schemat logiczny układu sterującego [opracowanie własne]

Największymi zaletami użycia płytki Arduino Nano jest prostota implementacji i programowania oraz niska cena urządzenia. Do programowania zostanie wykorzystane oprogramowanie Arduino IDE, dedykowane do płytek tego rodzaju. Arduino Nano posiada również wbudowane generatory sygnału PWM co znacznie ułatwia kontrolę i sterowanie silnikami DC poprzez sterowniki IBT\_2.

1. Parametry płytki mikroprocesorowej Arduino Nano

|  |  |
| --- | --- |
| **Parametr** | **Opis** |
| Napięcie zasilania | 5 V przez port USB lub 7-15 V przez pin 30 |
| Pobór prądu | ~15 mA |
| Mikrokontroler | ATmega328 8bit AVR, 16MHz |
| Pamięć SRAM | 2 kB |
| Pamięć flash | 32 kB (2kB stale zarezerwowane dla bootloader) |
| Pamięć EEPROM | 1 kB |
| Cyfrowe interfejsy komunikacyjne | UART, SPI (master/slave), I2C |
| Wyjścia/wejścia cyfrowe | 20 |
| Wyjścia analogowe (PWM) | 6 |
| Wejścia analogowe (ADC) | 8 |
| Złącza wejściowe | Mini-B USB, goldpin raster 2,54 mm (0.1 cala) |
| Przyciski wbudowane | REBOOT |
| Temperatura pracy | -40 °C / 85 °C |
| Wymiary | 43.2 x 18 x 16.6 mm |

Jako układ A został wybrany mikrokomputer Raspberry Pi 4B z tarczą wyświetlacza dotykowego, ze względu na łatwość implementacji HMI. Mikrokomputer tego typu posiada względnie dużą moc obliczeniową, podobną do komputera PC, co umożliwia przyszłą rozbudowę funkcji układu całego pojazdu.

1. Raspberry Pi 4 model B [www.botland.com.pl]

Raspberry Pi 4 B występuje w czterech wariantach, różniących się ilością pamięci operacyjnej RAM LPDDR4: 1GB, 2GB, 4GB, 8GB. Uwzględniając wydajność i cenę każdego z wariantów, wybrany został wariant 4GB. Taka ilość pamięci operacyjnej powinna być wystarczająca do przedstawionych w tej pracy zadań, ale i powinna umożliwić przyszłą rozbudowę układu o dodatkową funkcję, które mogą wymagać większej ilości pamięci operacyjnej.

1. Parametry mikrokomputera Raspberry Pi 4B – 4GB

|  |  |
| --- | --- |
| **Parametr** | **Opis** |
| Napięcie zasilania | 5 V przez wejście USB C lub odpowiednie piny GPIO |
| Pobór prądu | do 3 A |
| Procesor | Broadcom BCM2711 64-bit, rdzeń Quad-Core ARM Cortex-A72 |
| Taktowanie procesora | 1.5 GHz |
| Pamięć RAM | 4GB LPDDR4 |
| Pamięć flash | Brak – wymaga karty microSD |
| Połączenia bezprzewodowe | WiFi dualBand 2,4GHz/5GHz, Bluetooth 5.0 LE |
| Złącza | GPIO 40-pin raster 2.54 mm, 2x microHDMI, 2x USB 3.0, 2x USB 2.0, Ethernet 1000 Mbit/s, CSI, DSI, mcroSD, Jack 3.5 mm |
| System operacyjny | Linux Raspbian, Windows 10 loT |
| Interfejsy komunikacyjne | UART, SPI, I2C, GPIO |
| Wymiary | 85 x 56 x 17 mm |

* + 1. Nadajnik-odbiornik FlySky FS-iA6B

N/O FS-iA6B produkcji firmy FlySky to uniwersalny układ do zdalnej kontroli pojazdów, samolotów czy innych urządzeń RC. Do kontroli modułu używany jest pilot, który musi zostać wcześniej z nim sparowany. Przy budowie tego projektu został użyty pilot 6-kanałowy z możliwością odbierania informacji telemetrycznych (napięcie baterii, RPM silnika, ciśnienie i temperatura). FS-iA6B posiada sześć kanałów, których wartości mogą być przekazywane do komponentów za pomocą sygnałów PPM (np. serva) lub za pomocą interfejsu I-BUS. Zasięg radiowy tego układu nie przekracza 1.5 km w linii prostej (bez przeszkód i silnych zakłóceń).

1. FS-iA6B FlySky [www.rc-team.pl]
2. Parametry modułu FS-iA5B

|  |  |  |  |
| --- | --- | --- | --- |
| **Symbol** | **Parametr** | **Wartość** | **Jednostka** |
| Vcc | Napięcie zasilania | 4 - 6.5 | V |
| Ic | Pobór prądu | - | - |
| CH | Ilość kanałów wyjściowych | 6 | - |
| f | Pasmo częstotliwości | 2,405 -2,475 | GHz |
| Sp | Zasięg pracy | do 1800 | m |
| Pp | Moc nadajnika | 20 | dBm |

Producent nie podaje średniego lub maksymalnego poboru prądu. Ta wartość jest potrzebna do wyznaczenia mocy całego układu pojazdu i obliczenia długości działania przy zasilaniu akumulatorowym. Aby wyznaczyć tą wartość został wykonany pomiar multimetrem cyfrowym. Badanie wykonano za pomocą multimetru FY76 z błędem pomiarowym ± 1% + 5. Po wykonaniu dziesięciu pomiarów, wartość średnia wynosiła 43,21 mA, przy napięciu zasilania Vcc = 5 V. Dokładność pomiaru:

Możemy zatem przyjąć, że pobór prądu wynosi 43,21 ± 5,43 mA. Dla obliczeń mocy układu pojazdu zostanie przyjęta największa przybliżona możliwa wartość prądu 48,64 mA.

Połączony z FS-iA6B będzie pilot firmy FlySky FS-i6. Jest to 6-kanałowy pilot dedykowany do sterowania samolotami, helikopterami i podobnymi modelami latającymi. Można go jednak skonfigurować, aby działał jako pilot sterujący do pojazdu jeżdżącego. Każdy z kanałów może zostać dowolnie przypisany do aparatury pilota takich jak joysticki, potencjometry czy przełączniki dwu i trzy położeniowe.

* + 1. Dobór systemu zasilania - akumulator

Pojazd będzie zasilany za pomocą układu akumulatorowego. Rozważone zostały dwia główne rodzaje akumulatorów wraz z wymaganym systemem sterowania. Układ bazujący na akumulatorach żelowych i układ bazujący na akumulatorowych Li-Po.

Akumulator żelowy to rodzaj akumulatora kwasowo-ołowiowego, z elektrolitem w formie żelu. Głównymi zaletami takiego rozwiązania jest prostota kontroli układu zasilania, niska cena i większe bezpieczeństwo względem układu bazującego na ogniwach Li-Po. Wadami takiego rozwiązania jest mniejsza gęstość energetyczna akumulatora żelowego od ogniw Li-Po, z czym wiąże się większa masa takiego układu.

Układ bazujący na ogniwach Li-Po cechuje się większą gęstością energetyczną, przez co mniejszą masą. Ogniwa Li-Po mogą również dostarczyć większą moc, tzn. mogą być rozładowywane większymi prądami, bez uszkodzenia akumulatora. Wadami takiego układu jest wysoka cena ogniw li-po, większa złożoność sterowania i przede wszystkim niestabilność ogniw Li-Po. W przypadku przebicia warstwy ochronnej akumulatora, może dojść do samozapłonu ogniwa.

Aby wybrać układ należy wykonać obliczenia zapotrzebowania energetycznego całego pojazdu. Obliczenie maksymalnej mocy pobieranej przez pojazd zostało przedstawione na poniższym równaniu:

gdzie: In to prąd pobierany przez dany element, Un to napięcie danego elementu.

Uwzględniając współczynnik bezpieczeństwa, przyjęty jako 0.1 wartość końcowa wynosi Pmax = 37,22 W. Przewidując przyszłą rozbudowę pojazdu, wymagana moc maksymalna układu zasilania została przyjęta jako Pvcc = 50 W, co daje zapas ~22 W.

Po przyjęciu 50 W jako moc układu zasilania (w formie pracy ciągłej, nie impulsowej jak przy zapłonie silnika spalinowego), możemy odrzucić akumulator żelowy jako formę zasilania. Dostępne akumulatory, o gabarytach i masie mieszczących się w pojeździe nie są w stanie dostarczyć tak dużej mocy w trybie ciągłym. Należy zatem wykorzystać akumulator bazujący na ogniwach Li-Po. Wiąże się to z większym kosztem produkcji, lecz zredukuje to finalną masę pojazdu.

Maksymalny prąd rozładowywania w przypadku akumulatorów Li-Po jest podawany jako „battery rating” z jednostką C. Aby otrzymać wartość dla danej baterii w amperach należy wyliczyć tą wartość ze wzrou:

Został wybrany akumulator Li-Po o napięciu 14,8 V, pojemności 6000 mAh i klasie 60C. Można z tego wyliczyć maksymalny prąd i moc rozładowywania:

Wyznaczona wartość znacznie przekracza wymaganą moc zasilania. Do tego akumulatora został natomiast dobrany układ zasilający o mniejszej mocy, co oznacza, że maksymalna pobierana moc, nie osiągnie tak dużych wartości.

Można również obliczyć przewidywany czas działania pojazdu, bez ładowania akumulatora.

* + 1. Dobór systemu zasilania – sterowanie

Układ zasilający powinien posiadać 2 główne funkcje:

* Jasne sygnalizowanie dźwiękowe w przypadku spadku napięcia akumulatora poniżej wartości minimalnej,
* Zamiana wartości napięcia z akumulatora na 12 V i 5 V,

Biorąc pod uwagę te wymagania, wybrano układ BLDBVD2. Jest to układ zabezpieczający przed nadmiernym rozładowywaniem pojedynczych ogniw akumulatorów Li-Po. Układ posiada woltomierz z wyświetlaczem segmentowym, brzęczyk alarmowy i przycisk do regulacji wartości granicznych napięcia. Gdy napięcie któregoś z ogniw spadnie poniżej dopuszczalnej wartości, układ włącza alarm nadając sygnał dźwiękowy z brzęczyka.

Przetwornica generująca 12V musi mieć moc przynajmniej 18W, aby zasilić silniki, jednak należy przyjąć pewien współczynnik bezpieczeństwa zwiększający te wartość. Analogicznie dla przetwornicy generującej 5V, wartość wyliczona to 15,84 W.

* 1. Projekt pojazdu w oprogramowaniu CAD

Do zaprojektowania pojazdu zostało wybrane oprogramowanie SolidWorks2023, ze względu na szerokie zastosowanie w nowoczesnym przemyśle i możliwość wykorzystania funkcji symulacyjnych tego środowiska.

Pojazd można podzielić na pięć głównych grup elementów: gąsienice i koła, układ zawieszenia, obudowa, silniki i sprzęgła, oraz uchwyt dla układów elektronicznych.

* + 1. Gąsienica i koła

Gąsienice i koła gąsienicowe mogą być porównane do przekładni łańcuchowej

* + 1. Układ zawieszenia
    2. Dobór łożysk
    3. Silniki i sprzęgła
    4. Obudowa i montaż elektroniki
  1. Fizyczna realizacja projektu

Produkcja elementów pojazdu powstała dzięki pomocy Uczelnianego Centrum Druku 3D na Politechnice Koszalińskiej, które udostępniło maszyny przyrostowe oraz ploter laserowy.

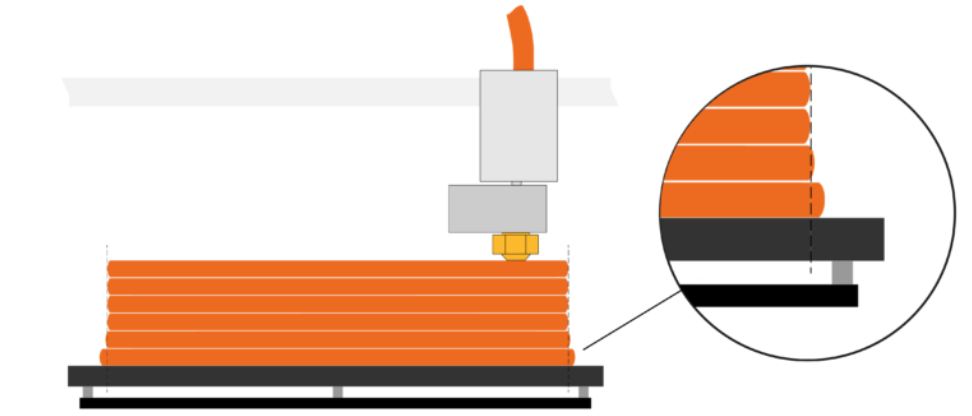
* + 1. Rodzaj produkcji i materiał elementów

Elementy o uproszczonej, płaskiej, geometrii, które mogą być przedstawione jako dwuwymiarowy rysunek, zostały wykonane przy pomocy wycinarki laserowej. W ten sposób zostały przygotowane elementy obudowy pojazdu jak i boczne belki trzymające elementy układu bieżnego. Zastosowany materiał to PMMA (potocznie plexiglas, szkło akrylowe), ze względu na swoją wysoką wytrzymałość, dostępność na rynku i podatność na cięcie laserowe.

Elementy o złożonej geometrii zostały wyprodukowane za pomocą metody addytywnej, przy użyciu polimeru PLA. Materiał ten został wybrany ze względu na wytrzymałość, dobrą odporność na ścieranie i kompatybilność z używanymi drukarkami 3d. Metoda addytywna pozwala na dokładną realizacje fizyczną modeli powstałych w oprogramowaniu CAD.

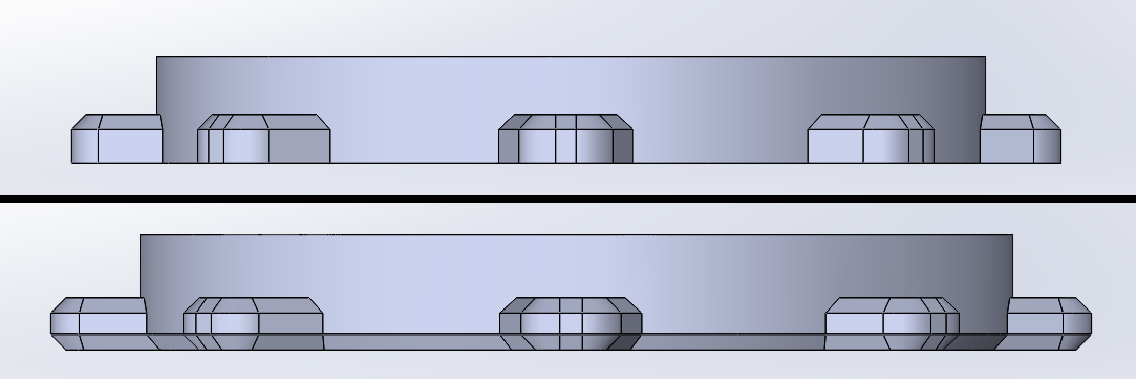
* + 1. Produkcja elementów bieżnych (metoda addytywna)

Pierwszym elementem przekazanym do produkcji było koło łańcuchowe, prowadzące gąsienice pojazdu. Kod G-Code został przygotowany w oprogramowaniu PrusaSlicer. Model koła został przygotowany w sposób, aby nie wymagane były podpory wspomagające produkcję.

Wyprodukowany element posiadał wadę. Przy druku powstał artefakt, nazywany potocznie „słoniową stopą”. Jest to zjawisko rozpływania się dolnej warstwy materiału, która zaczyna wystawać poza zarys elementu.

1. Powstawanie słoniowej stopy przy produkcji addytywnej [help.prusa3d.com]

Aby zapobiec powstawaniu tej niedoskonałości do modelu została dodana faza na dolnej ścianie 1 mm na 45°. Po zastosowaniu tej operacji, następne modele nie posiadały tego artefaktu.

1. Widok od boku koła, na górze bez operacji fazowania, na dole z operacją [opracowanie własne]
   * 1. Produkcja elementów przy pomocy wycinarki laserowej

Część elementów projektu została zaprojektowana, z myślą o produkcji przy pomocy wycinania laserowego. Wszystkie elementy wycinane są wykonane z płyt materiału PMMA, o grubości 3mm.

Elementy tego typu cechują się większą wytrzymałością strukturalną niż elementy wykonywane metodą addytywną, które posiadają warstwy, podatne na ścianie. PMMA jest jednolitym przeźroczystym materiałem polimerowym, szeroko stosowanym w przemyśle, głównie jako materiał do konstrukcji pokryw, obudów i ekranów.

W kwestii opisywanego projektu, decyzja o zastosowaniu elementów z tego materiału została podjęta ze względu na opisane wcześniej cechy tego materiału, jak i z powodu geometrii elementów. Obudowa i belki nośne pojazdu wykonane są z PMMA. Taka konstrukcja obudowy również umożliwia obserwację wewnętrznych elementów pojazdu, co jest bardzo przydatną cechą przy prototypowaniu konstrukcji.

Aby przygotować elementy do produkcji, musiały zostać one wyeksportowane z oprogramowania CAD do formy dwuwymiarowej. W oprogramowaniu są to obiekty trójwymiarowe, natomiast oprogramowanie plotera poruszającego laserem wymaga pliku dwuwymiarowego (rysunku), więc elementy przechodzą proces uproszczenia. Eksportowana jest tylko ściana, której kontury można bezpośrednio przekształcić na ścieżki ruchu plotera.

Szybkość ruchu głowicy laserowej i użyta moc zasilania zostały dobrane przez operatora maszyny w pracowni.

* 1. Realizacja części logicznej projektu
     1. Generalny opis działania

Pojazd posiada dwa tryby użytkowania – sterowanie radiowe i sterowanie autonomiczne.

Sterowanie radiowe realizowane jest przy pomocy N/O FlySky FS-iA6B i Arduino Nano. Dzięki temu trybowi operator pojazdu może sterować pojazdem za pomocą pilota. Sygnał z N/O typu PPM jest przekazywany do mikrokontrolera Arduino, który interpretuje te sygnały i przetwarza na wartości liczbowe. Następnie są one przekształcane na odpowiednie wartości dla prędkości silnika prawego i lewego. W zależności od tych prędkości pojazd będzie wykonywał skręt w odpowiednim kierunku lub jechał w linii prostej (do przodu lub do tyłu). Wartości prędkości są przekształcane na sygnał PWM, który odbierają sterowniki silników DC.

Sterowanie autonomicznie działa analogicznie, lecz sygnały wartości prędkości silników są wyznaczane przez algorytm jazdy autonomicznej. Sterowanie jest realizowane przy pomocy mikrokomputera Raspberry Pi, mikrokontrolera Arduino Nano oraz odbiciowych czujników odległości i magnetometru (kompasu). W trakcie jazdy autonomicznej pojazd zapisuje przejechaną mapę drogi w formie graficznego pliku .png. Po zakończeniu przejazdu plik można pozyskać z karty pamięci urządzenia lub poprzez połączenie z siecią WiFi mikrokomputera.

Algorytm jazdy autonomicznej steruje pojazdem bez wkładu operatora. Przed uruchomieniem tego trybu należy ustawić pojazd prawą stroną (w odległości ok. 10 cm) do ściany. Po włączeniu trybu autonomicznego pojazd przechodzi w stan sterowania przez algorytm.

Algorytm pierw wykonuje pomiar odległości z boku i przodu pojazdu za pomocą czujników odbiciowych. Jeżeli odległość na przodzie pojazdu jest większa niż 15 cm i odległość z boku pojazdu jest mniejsza niż 15 cm, pojazd jedzie do przodu. W trakcie jazdy pojazd wykonuje pomiary odległości i liczy czas przejazdu. Ponieważ przemieszcza się ze stałą prędkością, zmierzony czas może być bezpośrednio przeliczony na przebytą odległość. Gdy pomiar wykonywany w trakcie jazdy wykryje ścianę przed pojazdem lub brak ściany po boku, następuje zatrzymanie. Wysyłana jest wtedy komenda z Arduino Nano do Raspberry Pi przez port szeregowy, informująca o przejechanej odległości w linii prostej. Następnie wykonywany jest obrót o 90°. Kierunek obrotu uwarunkowany jest geometrią otoczenia – jeżeli wykryta ściana znajduje się przed pojazdem, skręci on w lewo, natomiast jeżeli wykryto brak ściany po boku, wykona skręt w prawo. W trakcie obrotu, wykonywane są pomiary magnetometrem ze stałym interwałem. Gdy wartość 90° zostanie osiągnięta, pojazd się zatrzymuje. Po wykonaniu obrotu wysyłana jest komenda do Raspberry Pi o kierunku wykonanego obrotu i pojazd wraca do jazdy w nowo obranym kierunku. Algorytm zapętla się.

Poniższa tabela zawiera każdy rodzaj komendy protokołu komunikacyjnego układu, wraz z opisem i funkcją. Komunikacja jest jednostronna, informacje są przesyłane z układu B (Arduino Nano) do układu A (Raspberry Pi). System komend był wzorowany na kodzie GCode używanym w ploterach, drukarkach 3d czy maszynach CNC – krótkie i proste komendy, w postaci kodu, z wartościami liczbowymi.

1. Spis komend komunikacyjnych układu

|  |  |  |
| --- | --- | --- |
| **Nazwa** | **Komenda** | **Opis** |
| Start | START | Komenda wysyłana przy rozpoczęciu jazdy autonomicznej, sygnalizująca start działania algorytmu |
| Obrót w prawo | R90 | Jest wysyłana po wykonaniu operacji obrotu o 90° w prawo, sygnalizuje że należy zmienić kierunek następnej rysowanej linii (drogi) |
| Obrót w lewo | L90 | Jest wysyłana po wykonaniu operacji obrotu o 90° w lewo, sygnalizuje że należy zmienić kierunek następnej rysowanej linii (drogi) |
| Przejazd | F[int] | Informuje o odległości przejechanej od ostatniego obrotu do nowego obrotu. Liczba całkowita po literze F jest wartością przejechanej odległości. Przykład: F100 – pojazd przejechał 100 jednostek |
| Koniec przejazdu | END | Jest wysyłana kiedy operator zakończy przejazd autonomiczny. Uruchamia procedurę zapisu pliku mapy |

* + 1. Program sterujący mikrokontrolerem Arduino Nano

Mikrokontroler Arduino Nano został zaprogramowany w oprogramowaniu Arduino IDE w zmodyfikowanym języku programistycznym C++. Jest to najbardziej popularna forma programowania tego typu mikrokontrolerów. Program jest wgrywany przez wbudowany programator połączony portem USB z komputerem.

Cały program nie został zawarty w pełni w pracy, ze względu na zachowanie odpowiedniej długości tekstu, ale całość można znaleźć w załącznikach podłączonych do pracy. Opisane zostały najważniejsze części programu.

Program używa dwie biblioteki – Wire.h i HMC5883L\_Simple.h. Biblioteka Wire.h jest odpowiedzialna za komunikację przy pomocy interfejsu I2C. Interfejs komunikacyjny I2C (Inter-Integrated Circuit) to dwukierunkowa magistrala szeregowa umożliwiająca komunikację z szeroką gamą czujników i urządzeń elektronicznych. Wire.h jest biblioteką udostępnianą przez korporację Arduino.

Biblioteka HMC5883L\_Simple.h odpowiada natomiast za obsługę i sterowanie magnetometrem QMC5883L użytego w projekcie. Autorem biblioteki jest James Sleeman i jest ona dostępna publicznie na portalu Github.com na podstawie licencji MIT. Pełny adres strony znajduje się w bibliografii. Biblioteka ta umożliwia odczytywanie surowych danych z magnetometru i przetwarzanie ich na dane liczbowe, które mogą zostać użyte w kodzie.

Główne funkcje programu:

* **Funkcja readChannel(int)** – jest odpowiedzialna za odczyt wartości z surowego sygnału PPM wysyłanego przez N/O FlySky. Funkcja przyjmuje argument w postaci adresu pinu, z którego należy odczytać wartość sygnału. Wynikiem działania funkcji jest wartość całkowitą (zmienna int) w przedziale od -100 do 100.
* **Funkcja cleanRead()** – wykonuje odczyt danych z wszystkich używanych kanałów za pomocą funkcji readChannel(), przycinając wartości i eliminując zakłócenia. Osiąga to przez zastosowanie martwej strefy wokół wartości zerowej. Ponieważ wartości odczytywane przez funkcję readChannel() mogą oscylować od -5 do 5 mimo, że wartość sterowana na pilocie wynosi zero, wartości w przedziale od -15 do 15 są ignorowane i interpretowane jako zero.
* **Funkcja driveMotors(int,int)** - odpowiedzialna jest za kontrolę prędkości i kierunku silników napędowych pojazdu. Przyjmuje jako argumenty dwie wartości całkowite reprezentujące prędkość dla każdego silnika (M1speed, M2speed). Funkcja najpierw ogranicza te wartości do zakresu od -100 do 100. Następnie sprawdza, czy kierunek obrotu silników zmienia się względem poprzedniego cyklu. W przypadku zmiany kierunku, funkcja zatrzymuje silniki na 50 ms, aby uniknąć uszkodzenia. Dla prędkości dodatniej, funkcja steruje odpowiednimi pinami PWM, aby obracać silnik do przodu, a dla prędkości ujemnej – do tyłu. Jeśli prędkość wynosi zero, oba piny sterujące generują sygnał PWM o zerowym wypełnieniu, zatrzymując silniki. Na końcu, zapisuje aktualne wartości prędkości jako poprzednie wartości dla kolejnego cyklu i wprowadza 50 ms opóźnienia dla stabilizacji sterowania.
* **Funkcja readDistance(int)** – jest odpowiedzialna jest za pomiar odległości przy pomocy ultradźwiękowego czujnika odległości. Przyjmuje jako argument numer pinu (echo), na którym odbierany jest sygnał odbity od przeszkody. Funkcja rozpoczyna pomiar przez ustawienie pinu TRIG\_PIN na stan niski na 2 mikrosekundy, a następnie na wysoki stan (HIGH) na 10 mikrosekund, aby wygenerować impuls ultradźwiękowy. Następnie ponownie ustawia pin TRIG\_PIN na niski stan. Funkcja używa funkcji pulseIn() do pomiaru czasu trwania sygnału wysokiego (HIGH) na pinie echo, co odpowiada czasowi powrotu impulsu odbitego od przeszkody. Czas ten jest następnie przeliczany na odległość w centymetrach za pomocą wzoru (1). Funkcja zwraca obliczoną odległość jako wartość całkowitą.
* **Funkcja turn(int)** – jest odpowiedzialna za operację obrotu o 90 ° w trybie jazdy autonomicznej. Funkcja realizuje to dzięki danych z kompasu cyfrowego. Pierw przyjmuje argument 1 (prawo) lub -1 (lewo). Następnie określa docelowy kąt obrotu, dodając lub odejmując 90° (w zależności od argumentu) do aktualnej wartości kąta, odczytanej z kompasu. Po wykonaniu tej operacji, funkcja uruchamia pętlę w której pojazd skręca ze stałą prędkością i odczytuje zmieniające się wartości z kompasu, aż do momentu kiedy wartość znajdzie się w zakresie ± 5° od wartości docelowej. Kiedy pojazd osiągnie tą wartość, silniki są zatrzymywane i operacja obrotu jest zakończona.

W pętli inicjacyjnej void setup(), wykonywanej tylko raz po uruchomieniu urządzenia, otwierane jest połączenie I2C z magnetometrem, deklarowane są piny wejściowe i wyjściowe, oraz aktywowane jest połączenie szeregowe z Raspberry Pi. Po wykonaniu tych instrukcji urządzenie przechodzi do głównej pętli programu void loop(). Jest wtedy aktywowana pętla while sterowana wartością kanału trzeciego. W zależności od wartości kanału trzeciego, sterowanego trójpozycyjnym przełącznikiem na pilocie, wybierany jest tryb pracy: autonomiczny lub pełne zdalne sterowanie.

W trybie pełnego zdalnego sterowania pojazd jest kontrolowany przez operatora pojazdu przez pilota. Program wywołuje pierw funkcję cleanRead() żeby odczytać wartości z kanałów radiowych. Kanał pierwszy (ruchy góra-dół gałki pilota) jest powiązany z jazdą do przodu i do tyłu natomiast kanał drugi (ruchy lewo-prawo gałki pilota) jest powiązany ze skręcaniem. W kodzie jest to realizowane przez odczyt wartości, i jeżeli tylko wartość kanału pierwszego jest różna od zera, wartość ta jest bezpośrednio przypisywana do silników, poprzez użycie jej jako argumentu do funkcji driveMotors(). Jeżeli natomiast wartość kanału 2 nie jest zerowa, jest ona dodawana do zmiennej prędkości silnika 1 i odejmowana od zmiennej prędkości silnika 2. Dzięki tej operacji uzyskiwać jest różnica prędkości między silnikami, co powoduje skręcanie pojazdu. Należy zauważyć, że jeżeli zmienna prędkości silników jest równa zero, pojazd wykona obrót w miejscu.

W trybie autonomicznym ruchem pojazdu steruje algorytm jazdy autonomicznej. Jest on zrealizowany przez serię instrukcji „if”.

* + 1. Program sterujący mikrokomputerem Raspberry Pi

Program wykonywany na mikrokomputerze Raspberry Pi został napisany w języku Python, w środowisku programistycznym Thony, bezpośrednio na urządzeniu, w systemie Rasbian. Program wymaga połączenia portem USB z mikrokontrolerem Arduino Nano do poprawnego funkcjonowania. Pełny kod programu znajduje się w załącznikach podłączonych do pracy.

W programie użyto trzech pięciu bibliotek: pygame, serial, time, datetime i os. Biblioteka pygame pozwala na tworzenie okien programu, obiektów graficznych i powiązanych z tym funkcji. Biblioteka serial jest odpowiedzialna za komunikację za pomocą portu szeregowego (w przypadku tego kodu z płytką Arduino Nano). Biblioteki time i datatime pozwalają na odczyt godziny i daty z urządzenia i są używanie do nadawania nazw zapisywanym plikom, na podstawie czasu zapisu. Biblioteka os pozwala na obsługę ścieżek i lokalizacji plików w systemie.

Główne funkcje programu:

* **Funkcja sanitize\_command(string)** – zadaniem funkcji jest oczyszczenie, walidacja i odpowiednia edycja komend otrzymywanych z portu szeregowego tak, aby mogły być poprawnie interpretowane przez resztę programu. Funkcja pierw czyści białe znaki (takie jak zakończenia linii) z otrzymanej komendy, następnie sprawdza jej rodzaj. Jeżeli jest to komenda przejazdu, oddziela z niej całkowitą wartość liczbową i przypisuje do zmiennej distance. Nie zwraca żadnych wartości.
* **Funkcja draw\_distance\_bar(int)** – funkcja służy do graficznego przedstawiania pełnej wartości przebytej przez pojazd w trakcie jednego przejazdu autonomicznego (między wysłaniem komend START i END). Jest to realizowane przez rysowanie pasa na dole ekranu, z tekstem informującym o pełnej wartości przejechanej przez pojazd. Aby ułatwić odczytywanie operatorowi, rysowany pas jest w kolorze czerwonym, natomiast tekst w białym.
* **Funkcja save\_map()** – funkcja ma zadanie zapisać mapę powstałą podczas przejazdu do pliku graficznego .png, nadania mu odpowiedniej nazwy składającej się z daty i godziny zapisu, aby zapewnić unikalność nazw plików. Przed wykonaniem zapisu funkcja sprawdza czy istnieje przypisany folder zapisu (shared). Jeżeli nie znajduje folderu o tej nazwie, tworzy go. Następnie generuję nazwę pliku i zapisuje aktualny stan okna programu jako plik .png w przypisanym folderze. Wyświetla informację na konsoli o zapisaniu mapy.
* **Funkcja reset\_map()** – zadaniem funkcji jest reset okna mapy wraz z jego zawartością i ustawienie wszystkich zmiennych na wartości początkowe. Osiąga to przez wypełnienie okna programu białym kolorem, przypisaniem wartości 0 do zmiennej total\_distance, oraz ustawienia pozycji początkowej na środek ekranu. Wywołuje następnie funkcję draw\_distance\_bar() aby narysować pas informujący o przebytej odległości.

Rysowanie linii map odbywa się w pętli „while running:”, gdzie zmienna running jest przedtem ustawiana na TRUE, co powoduje nieskończone iteracje pętli dopóki wartość zmiennej nie zostanie zmieniona na zaprzeczenie. Wartość zmiennej może zostać zmieniona na FALSE poprzez naciśniecie na przycisk zamykania okna programu.

W pętli program otrzymuje oczyszczoną komendę zaczynającą się od 'F', przelicza wskazaną odległość na piksele, mnożąc ją przez pixel\_per\_cm, a następnie dodaje tę wartość do całkowitej przebytej odległości, aktualizując zmienną total\_distance. Na podstawie aktualnego kierunku (direction), program rysuje linię od bieżącej pozycji (x, y) do nowej pozycji (new\_x, new\_y), zależnie od kierunku: w górę, dół, lewo lub prawo. Po wykonaniu tej operacji program aktualizuje nową pozycję (x, y). W przypadku otrzymania komendy obrotu w lewo (L90) lub w prawo (R90), program zmienia kierunek na podstawie mapy direction\_map, która określa nowy kierunek po wykonaniu obrotu.

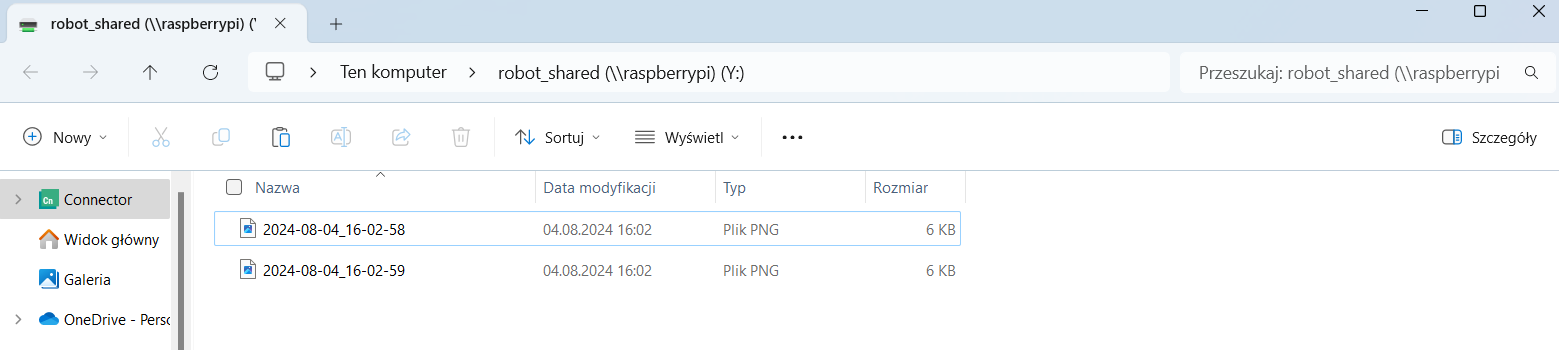
* + 1. Serwer plików na mikrokomputerze Raspberry Pi

Aby udostępniać pliki map, które generuje program w języku python, na mikrokomputerze Raspberry Pi został uruchomiony serwer plików. Aby ułatwić połączenie z serwerem, na mikrokomputerze założono również hotspot sieci Wi-Fi. Użytkownik może połączyć się z siecią Wi-Fi i następnie uzyskać dostęp do serwera plików.

Aby uruchomić serwer pierw został zainstalowany zdalny serwer plików Samba i jego zależności. Zostało to zrealizowane w terminalu systemu za pomocą komendy sudo apt install samba-common-bin. Inicjalizacja tej komendy instaluje oprogramowanie z upoważnieniami administratora. Następnie utworzono folder dla plików które będą udostępniane przez serwer (pliki graficzne map). Wybrano lokalizację /robot/shared/ żeby nie interferować z działaniem innych funkcji systemu. Po dodaniu folderu który będzie udostępniany przez serwer należy dodać konfigurację serwera samba. Do pliku /etc/samba/smb.conf dodano konfigurację o nazwie „robot\_shared” wpisując następujące instrukcje do pliku:

* [robot\_shared] – nazwa dodawanej konfiguracji
* path=/robot/shared – ścieżka do udostępnianego folderu
* writeable=No – zablokowano możliwość zapisywania plików przez użytkownika
* create mask=0755
* directory mask=0755 – dodano maskę pozwalającą na połączenie dla niezalogowanych użytkowników

Zablokowanie dodawania plików do serwera przez użytkownika serwera zostało zablokowane, ponieważ serwer ma tylko udostępniać pliki wygenerowane przez program, więc możliwość dodawania plików przez użytkownika jest zbędna. Serwer nie wymaga zalogowania się przez użytkownika ponieważ aby połączyć się z serwerem, należy znać hasło do sieci bezprzewodowej Wi-Fi, więc dodatkowe zabezpieczenie na poziomie serwera nie jest wymagane. Po skonfigurowaniu serwera, został uruchomiony z uprawnieniami administratora.

Aby połączyć się z serwerem, należy pierw połączyć się z siecią Wi-Fi o nazwie „robot\_PK” (hasło = admin123), i dodać lokalizację sieciową serwera. Można to uzyskać w systemie operacyjnym Windows dzięku funkcji „Mapuj dysk sieciowy” i podając adres \\raspberrypi\robot\_shared\ (oznacza to: \\adres hosta\nazwa udostępnionej lokalizacji\).

1. Udostępnione pliki graficzne map serwera, w eksploratorze plików Windows
   1. Połączenia elektryczne układu
      1. Układ zasilania

Pojazd posiada dwa napięcia zasilania, ze względu na różne zapotrzebowanie poszczególnych elementów układu.

Elementy logiczne: mikrokomputer Raspberry Pi, mikrokontroler Arduino Nano, sterowniki silników DC, N/O FlySky, odbiciowe sensory odległości i magnetometr są zasilane napięciem 5 volt. Napięcie to jest generowane przez samodzielny moduł zasilania XY-3606, którego moc maksymalna wynosi 25 watów. Przyjmuje on napięcie z akumulatora Li-Po, które w miarę używania urządzenia spada przez rozładowywanie akumulatora. Zakres napięcia wynosi więc około od 13.7 V do 10.7 V według producenta. Dla uproszczenia napięcie baterii jest rozważane jako 12 V.

Drugą grupą zasilanych elementów są silniki DC, które przyjmują prąd bezpośrednio z baterii poprzez sterowniki silników DC. Sterowniki modulują średnią wartość napięcia, poprzez wykorzystanie PWM. Silniki mogą więc uzyskać, w zależności od wysterowanej wartości, od 30% do 100% napięcia akumulatora. Napięcie akumulatora nie jest stałe i spada w miarę rozładowywania.

* + 1. Generalny układ połączeń

Rysunki schematów połączeń elektrycznych znajdują się w załączniku do pracy. Generalny spis najważniejszych połączeń układu:

* Bateria Li-Po (12 Volt) jest połączona ze sterownikami IBT\_2 i układem XY-3606 zasilającym elementy logiczne układu (5 Volt);
* Układ XY-3606 zasila Raspberry Pi, FS-iA6B, Arduino Nano oraz wszystkie sensory pojazdu;
* Arduino Nano jest połączone z Raspberry Pi za pomocą USB;
* Arduino Nano steruje układami IBT\_2, za pomocą sygnałów PWM, które są bezpośrednio połączone z silnikami DC;
* Arduino Nano odbiera sygnały impulsowe z sensorów HC-SR04 oraz komunikuje się z magnetometrem QMC5883L z wykorzystaniem protokołu I2C;
* FS-iA6B jest połączony z Arduino Nano i przesyła wartości z kanałów radiowych za pomocą sygnału PPM;
* Do ogniw Li-Po jest podłączony układ bezpieczeństwa BLDBVD2, oraz woltomierz cyfrowy z wyświetlaczem segmentowym.
  + 1. Użycie płytki drukowanej

W projekcie zastosowano płytkę drukowaną PCB, aby ułatwić realizację połączeń układu. Rysunek techniczny oraz schemat elektryczny płytki znajdują się w załącznikach podłączonych do pracy.

* 1. Badania działania pojazdu
     1. Badanie prędkości i przewidywanego zasięgu pojazdu
     2. Badanie funkcji autonomicznych

1. Wnioski

Bibliografia

1. Marian Doległo, 2016, Podstawy elektrotechniki i elektroniki ISBN: 978-83-206-1968-3
2. Seweryn Orzełowski, 2010, Budowa podwozi i nadwozi samochodowych
3. Konrad Zajkowski, Stanisław Duer, Stanisław Sokołowski, 2013, Laboratorium z podstaw Elektrotechniki i Elektroniki
4. Continuous Track, 2023, https://en.wikipedia.org/wiki/Continuous\_track (14.12.2023)
5. TALON Tracked Military Robot, 2020, https://www.army-technology.com/projects/talon-tracked-military-robot/?cf-view (04.01.2024)
6. Space Communications and Navigation, 2023, https://www.nasa.gov/directorates/space-operations/space-communications-and-navigation-scan-program/scan-outreach/fun-facts/#radio-waves-what (04.01.2024)
7. Silniki prądu stałego, 2021, https://strefainzyniera.pl/artykul/1033/silniki-pradu-stalego (06.01.2024)
8. Akumulatory litowo-polimerowe (Li-Poly), 2018, https://wamtechnik.pl/produkty/technologie-litowe/akumulatory-litowo-polimerowe-li-poly/ (06.01.2024)
9. Prof. dr hab. inż. Grzegorz Budzik, Dr inż. Joanna Woźniak, Dr inż. Łukasz Przeszłowski, 2022, Druk 3D jako element przemysłu przyszłości. Analiza rynku i tendencje rozwoju
10. Yılmaz GÜVEN, Ercan COŞGUN, Sıtkı KOCAOĞLU, Harun GEZİCİ, Eray YILMAZLAR, 2017, Understanding the Concept of Microcontroller Based Systems To Choose The Best Hardware For Applications
11. Arduino Team, One board to rule them all: History of the Arduino UNO, 2021, https://blog.arduino.cc/2021/12/09/one-board-to-rule-them-all-history-of-the-arduino-uno/ (03.08.2024)
12. Dokumentacja techniczna Arduino Nano, Product Reference Manual, 2023, Arduino
13. Zbigniew Goryca, 2012, METODY STEROWANIA SILNIKÓW BLDC, Prace Naukowe Instytutu Maszyn, Napędów i Pomiarów Elektrycznych Politechniki Wrocławskiej

Spis rysunków

[Rys. 1. Amerykański robot saperski TALON 7](#_Toc174208960)

[Rys. 2. Schemat budowy silnika szczotkowego prądu stałego [7] 10](#_Toc174208961)

[Rys. 3. Układ sprzężenia zwrotnego – regulacji [opracowanie własne] 13](#_Toc174208962)

[Rys. 4. Schemat budowy typowego gąsienicowego układu bieżnego [opracowanie własne] 15](#_Toc174208963)

[Rys. 5. Układ nowoczesnego mikrokontrolera i jego komponenty 15](#_Toc174208964)

[Rys. 6. Schemat procesu wytwarzania przyrostowego [9] 19](#_Toc174208965)

[Rys. 7. Schemat działania ekstrudera maszyny przyrostowej [opracowanie własne] 20](#_Toc174208966)

[Rys. 8. Rozebrany silnik DC używany w pracy, wraz z opisem elementów [opracowanie własne] 22](#_Toc174208967)

[Rys. 9. Schemat logiczny mostka H, lub dwóch pół-mostków [opracowanie własne] 23](#_Toc174208968)

[Rys. 10. Schemat logiczny układu sterującego [opracowanie własne] 26](#_Toc174208969)

[Rys. 11. Raspberry Pi 4 model B [www.botland.com.pl] 27](#_Toc174208970)

[Rys. 12. FS-iA6B FlySky [www.rc-team.pl] 28](#_Toc174208971)

[Rys. 13. Powstawanie słoniowej stopy przy produkcji addytywnej [help.prusa3d.com] 32](#_Toc174208972)

[Rys. 14. Widok od boku koła, na górze bez operacji fazowania, na dole z operacją [opracowanie własne] 32](#_Toc174208973)

Spis tabel

[Tab. 1. Parametry modułu IBT\_2 - sterownik silników DC 24](#_Toc173931093)

[Tab. 2. Parametry sensora HC-SR04 25](#_Toc173931094)

[Tab. 3. Parametry magnetometru QMC5883L 26](#_Toc173931095)

[Tab. 4. Parametry płytki mikroprocesorowej Arduino Nano 27](#_Toc173931096)

[Tab. 5. Parametry mikrokomputera Raspberry Pi 4B – 4GB 28](#_Toc173931097)

[Tab. 6. Parametry modułu FS-iA5B 29](#_Toc173931098)

[Tab. 7. Spis komend komunikacyjnych układu 36](#_Toc173931099)