

POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: AUTOMATYKA I ROBOTYKA (AIR)
SPECJALNOŚĆ: TECHNOLOGIE INFORMACYJNE W SYSTEMACH
AUTOMATYKI (ART)

PRACA DYPLOMOWA
INŻYNIERSKA

Aplikacja mobilna do sterowania robotem
minisumo

Mobile application for controlling a minisumo
robot

AUTOR:

Łukasz Miłaszewski

PROWADZĄCY PRACĘ:

dr inż. Łukasz Jeleń

OCENA PRACY:

Spis treści

1. Wstęp	6
1.1. Minisumo	6
1.2. Założenia	7
2. Wykorzystane technologie	8
2.1. C	8
2.1.1. HAL	8
2.2. Swift	9
2.2.1. UIKit	10
2.2.2. CoreBluetooth	10
2.2.3. CoreGraphics	10
2.2.4. CoreMotion	11
2.3. Arduino	11
3. Komunikacja	13
3.1. Moduł bluetooth	13
3.2. Realizacja komunikacji	14
4. Robot minisumo	17
4.1. Konstrukcja	17
4.1.1. Założenia	17
4.1.2. Nadwozie	18
4.1.3. Podwozie	18
4.1.4. Napęd	19
4.2. Elektronika	20
4.2.1. Założenia	21
4.2.2. Źródło zasilania	21
4.2.3. Procesor	21
4.2.4. Sensoryka	21
4.2.5. Sterownik silników	21
4.2.6. Schemat płytki z interfejsem	21
4.2.7. Schemat płytki głównej	21

4.3. Oprogramowanie	21
4.3.1. Transmisja danych	21
4.3.2. Obsługa przychodzących wiadomości	21
4.3.3. Algorytmy walki	21
5. Aplikacja mobilna	22
5.1. Platforma	22
5.2. Kompatybilność	22
5.3. Wzorzec MVC	22
5.4. Komunikacja	22
5.5. Struktura aplikacji	22
5.5.1. Widok główny	22
5.5.2. Widok sterowania automatycznego	22
5.5.3. Widok sterowania zdalnego	22
5.5.4. Widok diagnostyki	22
6. Implementacja	23
6.1. Kompilacja projektu	23
7. Podsumowanie	24
7.1. Zrealizowane założenia	24
7.2. Dalszy rozwój projektu	24
7.3. Uwagi	24
Literatura	25

Spis rysunków

1.1.	Zawody sumo.	7
2.1.	Konfiguracja peryferiów użytego procesora STM32F100C8T6B – LQFP48.	9
2.2.	Środowisko Xcode.	9
2.3.	Układ służący do konfiguracji modułu bluetooth.	11
3.1.	Schemat komunikacji między urządzeniami.	13
3.2.	Moduły Bluetooth.	14
3.3.	Schemat wiadomości zawierającej informacje odnośnie stanu sensorów.	15
3.4.	Schemat wiadomości w której zawarta jest konfiguracja robota.	15
3.5.	Schemat wiadomości zdalnego sterowania przy użyciu akcelerometru.	16
4.1.	Wykonany robot minisumo.	17
4.2.	Projekt nadwozia w środowisku Autodesk Inventor 2017.	18
4.3.	Projekt podwozia w środowisku Autodesk Inventor 2017.	19
4.4.	Napęd robota minisumo.	20

Spis listingów

2.1. Kod programu umożliwiającego konfigurację modułu Bluetooth HM-10. 12

Rozdział 1

Wstęp

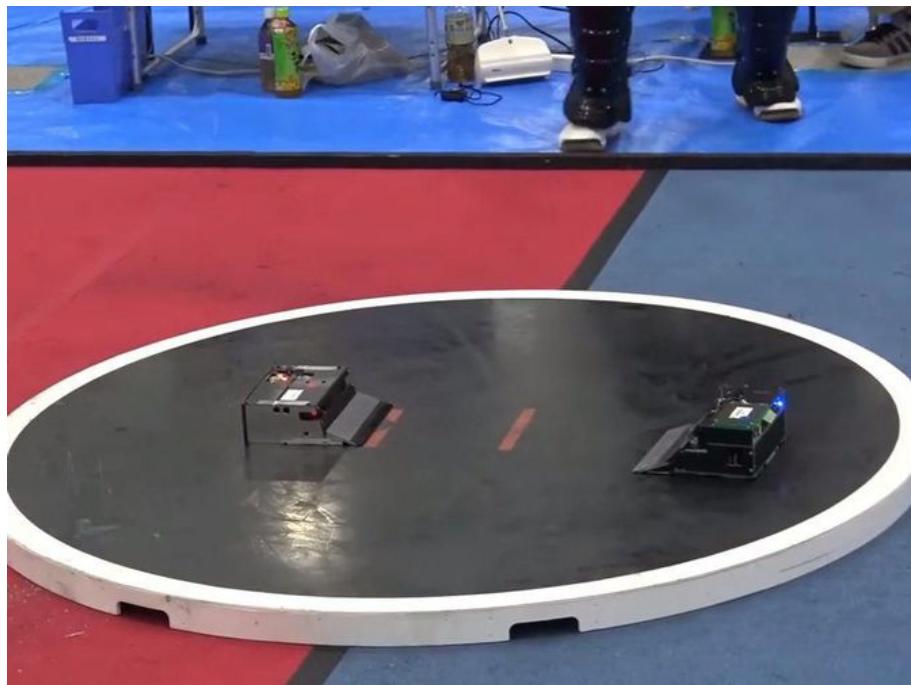
Celem niniejszej pracy jest implementacja aplikacji mobilnej służącej do sterowania robotem minisumo. W ramach pracy dyplomowej powstał samodzielnie wykonany dwukołowy robot w pełni spełniający wymagania do startu w zawodach minisumo. Dodatkowo powstała aplikacja mobilna na platformę iOS, która daje możliwość obsługi oraz konfiguracji wyżej wspomnianego robota. Dzięki niej użytkownik może wybrać jedną z wielu strategii walki, ustalić maksymalną moc silników oraz uwzględnić oczekiwanie na start za pomocą odbiornika fal podczerwonych. Dodatkowo aplikacja oferuje możliwość zdalnego sterowania robotem za pomocą akcelerometru lub wirtualnego dżojstiku oraz sprawdzenia poprawności działania sensorów i silników.

1.1. Minisumo

Minisumo jest jedną z kategorii walk robotów wzorowanych na popularnym japońskim sporcie – zapasach sumo. Tak samo jak i w prawdziwym sporcie, starcie odbywa się na okrągłym ringu. Wygrywa ten robot, który jako pierwszy wypchnie rywala z areny. Obowiązujące zasady są takie same dla każdej z kategorii (sumo, minisumo, nanosumo, pentosumo) z wyjątkiem dopuszczalnej wagi oraz rozmiaru. Dla minisumo maksymalna waga to 500 gramów, a szerokość oraz długość nie mogą przekroczyć 100 milimetrów. Dodatkowo każdy z robotów musi spełniać poniższe wymagania:

- musi być w pełni autonomiczny,
- nie może być przytwierdzony do areny,
- nie może zakłócać sterowania przeciwnika,
- musi posiadać na wyposażeniu moduł startowy, dający możliwość zdalnego uruchomienia robota przez sędzięgo,
- nie może emitować cieczy, gazów oraz nadmiernego ciepła.

Na rysunku 1.1 przedstawiono przykładową walkę robotów klasy sumo. Warto zauważyć, iż wnętrze areny jest czarne, natomiast obwód biały. Dzięki zastosowanemu kontrastowi robot wyposażony w odpowiednie czujniki jest w stanie wykryć brzeg areny.



Rys. 1.1: Zawody sumo.

1.2. Założenia

Główne założenia realizowanego projektu:

- stworzenie robota spełniającego wymagania kategorii minisumo,
- sprawna sensoryka pozwalająca na wykrycie przeciwnika oraz końca ringu,
- w pełni działająca komunikacja między robotem a aplikacją,
- aplikacja mobilna pozwalająca na konfigurację wyżej wspomnianego robota.

Etapy realizacji wyżej wymienionych założeń zostały opisane w dalszej części pracy z podziałem na tworzonego robota minisumo oraz aplikację mobilną.

Rozdział 2

Wykorzystane technologie

Projekt został zrealizowany pod systemem Windows 10 oraz Mac OS X El Capitan.

2.1. C

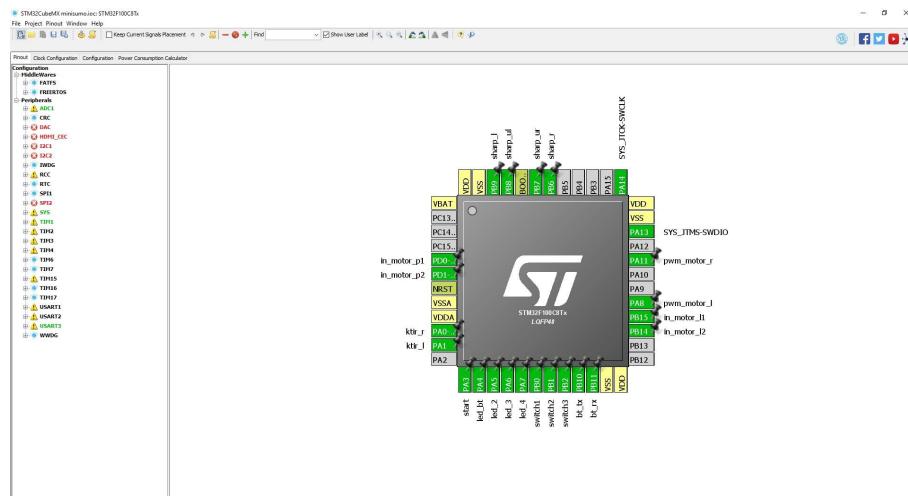
Z racji, iż sercem robota jest procesor z rodziny STM, wybór technologii został ograniczony do języka C lub C++. Wybrano język C z powodów optymalizacyjnych oraz małego stopnia skomplikowania programu.

Oprogramowanie zostało stworzone w środowisku Eclipse z dodatkiem AC6 wspierającym platformę STM32.

2.1.1. HAL

HAL (Hardware Abstraction Layer) jest biblioteką będącą wysokopoziomowym interfejsem służącym do konfiguracji peryferiów mikrokontrolera. Zdecydowano się na wyżej wspomnianą bibliotekę z powodu bardzo przejrzystej dokumentacji oraz łatwości użytkowania. Dodatkowo użyto środowiska CubeMX, które udostępnia graficzny interfejs, który pozwala na stosunkowo łatwą oraz intuicyjną konfigurację procesora oraz wygenerowanie projektu w języku C wraz z użyciem bibliotek HAL.

Rysunek 2.1 przedstawia konfigurację peryferiów użytego procesora w środowisku CubeMX.



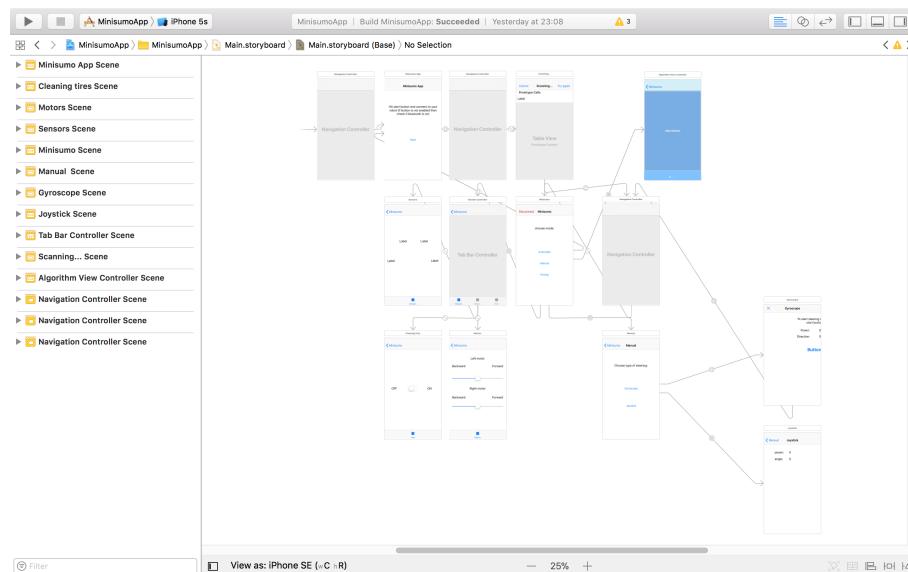
Rys. 2.1: Konfiguracja peryferiów użytego procesora STM32F100C8T6B – LQFP48.

2.2. Swift

Swift jest językiem natywnym (następcą języka Objective–C) zaprezentowanym przez Apple Inc. w 2014 roku. Wykorzystywany jest do tworzenia oprogramowania na platformy macOS oraz iOS. W pracy dyplomowej użyto wersji języka 3.0, ponieważ była to najnowsza wersja wspierana przez docelowe urządzenie, którym był telefon iPhone 5.

Środowiskiem użyтыm do tworzenia aplikacji mobilnej w technologii Swift był Xcode, którego dużym atutem jest występowanie graficznego interfejsu umożliwiającego tworzenie widoków aplikacji. Dzięki temu tworzenie aplikacji jest bardziej intuicyjne oraz pozwala na sprawne wprowadzanie zmian w tworzonych widokach.

Ilustracja 2.2 ukazuje środowisko Xcode wraz z widokami aplikacji oraz zależnościami między nimi.



Rys. 2.2: Środowisko Xcode.

2.2.1. UIKit

UIKit jest platformą, która zapewnia infrastrukturę dla aplikacji. Udostępnia architekturę widoku do implementacji interfejsu, obsługę zdarzeń, obsługę wielopunktowego dotyku, zarządza zasobami oraz interakcjami pomiędzy aplikacją a użytkownikiem. Dodatkowymi funkcjami są obsługa dokumentów, aplikacji oraz jej rozszerzeń, zarządzanie tekstem i wyświetlaniem.

UIView

Widoki są podstawowymi elementami składowymi interfejsu użytkownika aplikacji, a klasa `UIView` definiuje zachowania wspólne dla wszystkich widoków. Obiekt widoku renderuje zawartość w obrębie prostokąta obwiedni i obsługuje wszelkie interakcje z tą zawartością.

UITableView

`UITableView` wyświetla listę pozycji w pojedynczej kolumnie. Jest podklassą `UIScrollView`, która pozwala użytkownikowi przewijać tabelę, z tą różnicą, że przewijanie dozwolone jest wyłącznie w pionie. Komórki zawierające poszczególne elementy tabeli są obiektami `UITableViewCell`. `UITableView` używa wspomnianych obiektów do rysowania widocznych wierszy tabeli. Komórki mogą posiadać tytuły treści, obrazy oraz widoków akcesoriów. Widoki akcesoriów mogą być również elementami sterującymi, takimi jak przełączniki i suwaki.

2.2.2. CoreBluetooth

Platforma `CoreBluetooth` dostarcza klasy niezbędne do komunikacji aplikacji z urządzeniami wyposażonymi w bezprzewodową technologię Bluetooth.

CBPeripheral

Klasa `CBPeripheral` reprezentuje zdalne urządzenia peryferyjne, których połączenie aplikacja wykrywa za pośrednictwem instancji `CBCentralManager`. Urządzenia peryferyjne są identyfikowane za pomocą unikalnych identyfikatorów (UUID) reprezentowanych przez obiekty NSUUID.

CBCentralManager

Obiekty `CBCentralManager` służą do zarządzania podłączonymi urządzeniami peryferyjnymi (reprezentowanymi przez obiekty `CBPeripheral`). Pozwalają na skanowanie oraz wykrywanie zdalnych urządzeń.

2.2.3. CoreGraphics

Platforma `CoreGraphics` oparta jest na zaawansowanym mechanizmie rysowania `Quartz`. Zapewnia niskopoziomowe renderowanie 2D, zarządzanie wzorcami, gradientami oraz danymi

obrazu. Dodatkowo pozwala na tworzenie oraz maskowanie obrazu, a także wyświetlanie i analizowanie dokumentów PDF. W pracy dyplomowej *CoreGraphics* zostało wykorzystane do stworzenia wirtualnego dżojstiku służącego zdalnemu sterowaniu robotem minisumo.

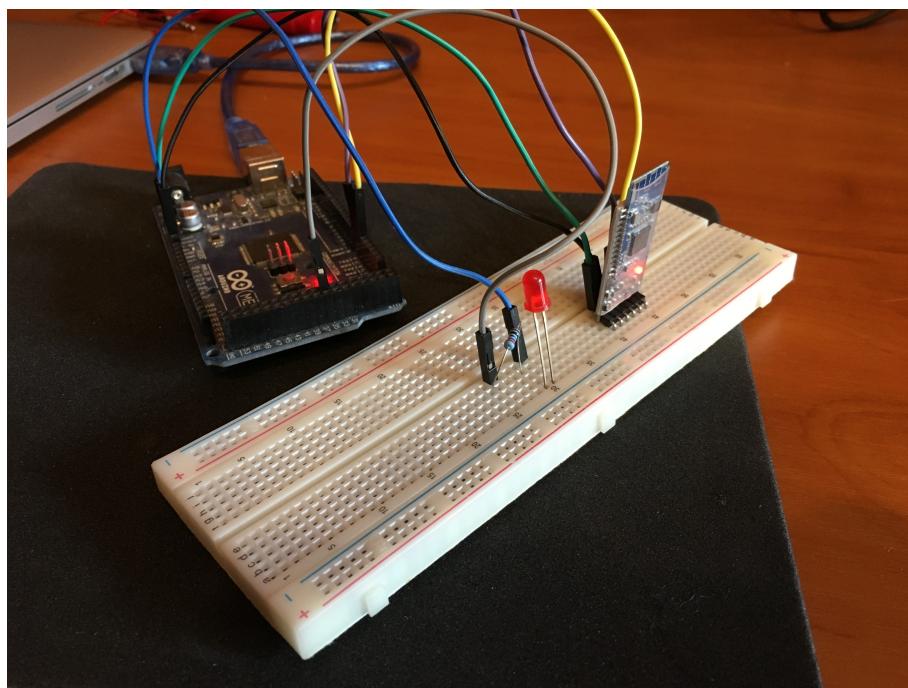
2.2.4. CoreMotion

Dostarcza dane dotyczące ruchu urządzenia na podstawie wbudowanego akcelerometru, żyroskopu, krokomierza, magnetometru, barometru itp. W aplikacji użyto wspomnianej platformy do sterowania ruchem robota za pomocą akcelerometru.

2.3. Arduino

Arduino jest platformą programistyczną przeznaczoną dla mikrokontrolerów z wbudowaną obsługą układów wejścia oraz wyjścia. Język programowania *Arduino* oparty jest na języku C/C++. Głównym atutem omawianej platformy jest szeroka społeczność, dokładna dokumentacja oraz łatwość prototypowania prostych układów. Jednakże omawiania platformy nie jest zalecana w przypadku złożonych projektów ze względu na wysokopoziomowość oraz niestabilność pracy.

Platforma została użyta do konfiguracji modułu Bluetooth za pomocą komend AT. Skonfigurowano takie parametry jak nazwa urządzenia oraz liczbę symboli na sekundę (ang.*baud rate*). Do tego celu użyto mikrokontrolera *Arduino Mega 2560* oraz płytka stykowej wraz z przewodami połączeniowymi. Konfiguracja modułu była jednorazową czynnością, dlatego zdecydowano się na platformę *Arduino* ze względu na wyżej wspomnianą szybkość oraz prostotę obsługi.



Rys. 2.3: Układ służący do konfiguracji modułu bluetooth.

Ilustracja 2.3 obrazuje układ Arduino Mega 2560 wraz z płytą stykową oraz podłączonym modułem bluetooth. Dodatkowo układ wyposażono w diodę mającą na celu sygnalizację poprawności wgrywania komend.

```
1 #include <SoftwareSerial.h>
2
3 int LEDPIN = 13;
4
5 void setup() {
6     pinMode(LEDPIN, OUTPUT);
7     Serial.begin(9600); // communication with computer
8     Serial3.begin(9600); // communication with HM-10
9 }
10
11 void loop() {
12     if (Serial3.available() ) {
13         char dioda = Serial3.read();
14         Serial.write(dioda);
15         if (atoi(&dioda)==1) {
16             digitalWrite(LEDPIN, HIGH);
17         }
18
19         if (atoi(&dioda)==0) {
20             digitalWrite(LEDPIN, LOW);
21         }
22     }
23
24     if (Serial.available() ) { Serial3.write( Serial.read() ); }
25 }
```

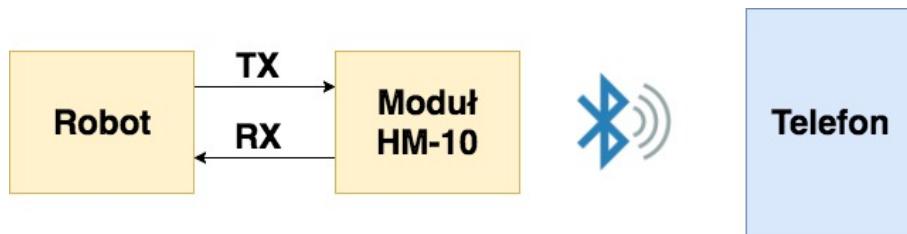
Listing 2.1: Kod programu umożliwiającego konfigurację modułu Bluetooth HM-10.

Listing 2.1 przedstawia kod programu zapewniającego komunikację między komputerem, a modułem Bluetooth przy użyciu mikrokontrolera *Arduino Mega 2560*. Za pomocą terminalu udostępnionego przez środowisko *Arduino* możliwa jest komunikacja oraz konfiguracja modułu HM-10. Dodatkowo program został rozszerzony o zapalanie oraz gaszenie diody LED w celu sprawdzenia poprawności połączenia. Jeżeli przy użyciu wyżej wspomnianego terminala zostanie wysłana wartość 1, powinna zapalić się dioda. W przypadku wysłania wartości 0, dioda powinna zgasnąć.

Rozdział 3

Komunikacja

Komunikacja między urządzeniami (robotem a aplikacją mobilną) została zrealizowana przy użyciu łączności bezprzewodowej Bluetooth.

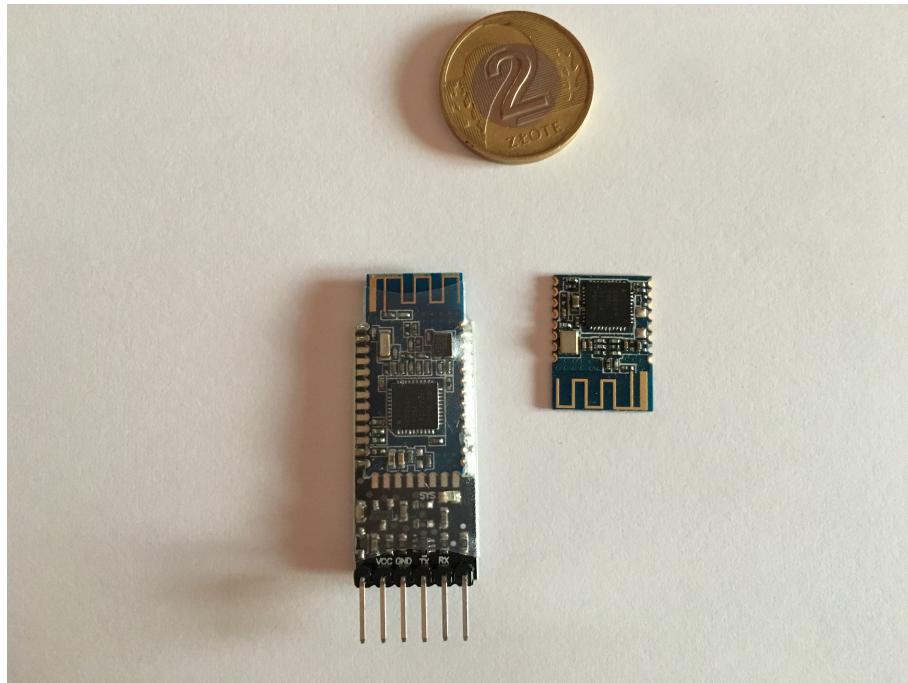


Rys. 3.1: Schemat komunikacji między urządzeniami.

Na rysunku 3.1 przedstawiono schemat komunikacji między robotem a aplikacją mobilną. Aplikacja mobilna komunikuje się radiowo z modułem Bluetooth, który następnie za pomocą przewodów podłączonych do linii TX oraz RX procesora przesyła wyslaną wiadomość.

3.1. Moduł bluetooth

Z racji, iż docelowym urządzeniem komunikującym się z robotem był telefon iPhone, który obsługuje wyłącznie technologię BLE (Bluetooth Low Energy), wybór docelowego modułu komunikacyjnego został mocno ograniczony. Z początku zdecydowano się na moduł Bluetooth HM-11 ze względu na bardzo małe rozmiary. Niestety okazało się, iż owy moduł posiada nietypowy raster, przez co niemożliwym było przylutowanie jakiegokolwiek złącza. Jedynym rozwiązaniem byłoby przylutowanie modułu do płytki elektronicznej robota, aczkolwiek projekt płytki nie przewidywał dodatkowych elementów. W związku z czym zdecydowano się na starszy model HM-10, który dostarczał taką samą funkcjonalność, lecz posiadał gotowe złącze goldpin, które znaczco ułatwiało podłączenie do wyżej wspomnianej płytki. Jedyną wadą zastosowanego modułu był rozmiar.



Rys. 3.2: Moduły Bluetooth.

Na rysunku 3.2 po lewej stronie widoczny jest docelowo użyty moduł HM-10, natomiast po prawej wcześniej wspomniany HM-11. U góry ilustracji znajduje się moneta dla odwzorowania rozmiarów omawianych modułów.

3.2. Realizacja komunikacji

Komunikacja między urządzeniami bazuje na wysyłaniu ciągu dziesięciu znaków, a dokładniej nieujemnych liczb całkowitych, które następnie są parsowane oraz interpretowane w odpowiedni sposób. Domyślnie wiadomość składa się z dziesięciu zer, a następnie wypełniana jest odpowiednimi wartościami. Poniżej przedstawiono przykładowe formaty wiadomości przesyłane między urządzeniami.

Sensory

Komunikacja między urządzeniami rozpoczyna się od wysłania wiadomości przez aplikację mobilną w której pierwsza cyfra wiadomości ma wartość równą 4. Następnie co określony czas robot wysyła wiadomość w której zawarty jest aktualny stan czujników. Komunikacja kończy się w momencie, gdy aplikacja mobilna wyśle wiadomość w której pierwsza cyfra jest różna od 4.

4 A B C D XX YY 0

Rys. 3.3: Schemat wiadomości zawierającej informacje odnośnie stanu sensorów.

Na rysunku 3.3 przedstawiono strukturę wiadomości wysłanej przez robota do aplikacji mobilnej. Pierwsza cyfra (równa 4) sygnalizuje aplikacji, iż przesyłana wiadomość zawiera informacje na temat stanu czujników. Cyfry *A*, *B*, *C*, *D* sygnalizują stan cyfrowych czujników wykrywających przeciwnika w sposób binarny (0 – nie wykryto przeciwnika, 1 – wykryto przeciwnika). Cyfry *XX* oraz *YY* informują o stanie analogowych czujników wykrywających koniec ringu (0 – wykryto czarny kolor, 100 – wykryto biały kolor).

Automatyczne Sterowanie

Komunikacja bazuje na jednorazowym wysłaniu wiadomości do robota z poziomu aplikacji mobilnej. Treść wiadomości określa czy robot powinien wstrzymać start do momentu otrzymania komunikatu od sędziego, rodzaj algorytmu walki oraz maksymalną moc silników.

1 A B XXX 0000

Rys. 3.4: Schemat wiadomości w której zawarta jest konfiguracja robota.

Rysunek 3.4 obrazuje schemat wiadomości wysłanej z aplikacji mobilnej do robota. Pierwsza cyfra (równa 1) informuje, iż robot przechodzi w stan automatycznej jazdy. Cyfra *A* określa numer algorytmu walki (na chwilę obecną występują trzy algorytmy, numerowane 1,2 oraz 3). Kolejna cyfra *B* dostarcza informację odnośnie rodzaju startu (0 – ruch robota rozpoczyna się w momencie wysłania wiadomości, 1 – robot oczekuje na sygnał sędziego). Ostatnia wartość *XXX* określa procentowo maksymalną moc jaką mogą osiągnąć obydwa silniki (przyjmuje wartości 000 – 100, gdzie 100 oznacza maksymalną moc, a 0 minimalną).

Sterowanie zdalne - akcelerometr

Aplikacja mobilna jednorazowo wysyła komunikat, którego pierwsza cyfra oznacza wejście w trym zdalnego sterowania. Następnie aplikacja rozpoczyna nasłuchiwanie wiadomości zawierających kierunki ruchu oraz moce poszczególnych silników.

6 A B XXX YYY 0

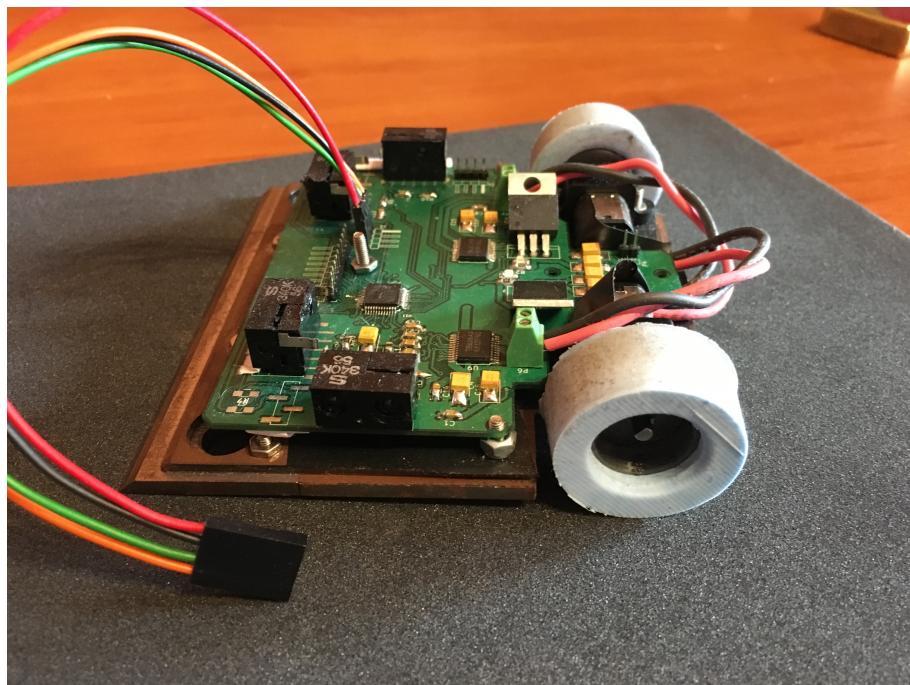
Rys. 3.5: Schemat wiadomości zdalnego sterowania przy użyciu akcelerometru.

Ilustracja 3.5 przedstawia budowę wiadomości wysłanej do aplikacji mobilnej przez robota. Pierwsza cyfra (równa 6) wskazuje, iż robot jest w trybie zdalnego sterowania za pomocą akcelerometru. Kolejne cyfry *A* oraz *B* oznaczają kierunki odpowiednio lewego oraz prawego silnika (1 – obrót w tył, 2 – obrót w przód). *XXX* oraz *YYY* podobnie jak w przypadku trybu automatycznej jazdy określają procentowe moce silników, lewego oraz prawego (przyjmuje wartości 000 – 100, gdzie 100 oznacza maksymalną moc, a 0 minimalną).

Rozdział 4

Robot minisumo

W niniejszym rozdziale opisany został proces tworzenia robota minisumo. Wyróżniono w nim trzy główne sekcje: *Konstrukcja*, *Elektronika* oraz *Oprogramowanie*.



Rys. 4.1: Wykonany robot minisumo.

Zdjęcie 4.1 przedstawia omawianego robota minisumo. W celu ukazania elektroniki ściągnięto nadwozie.

4.1. Konstrukcja

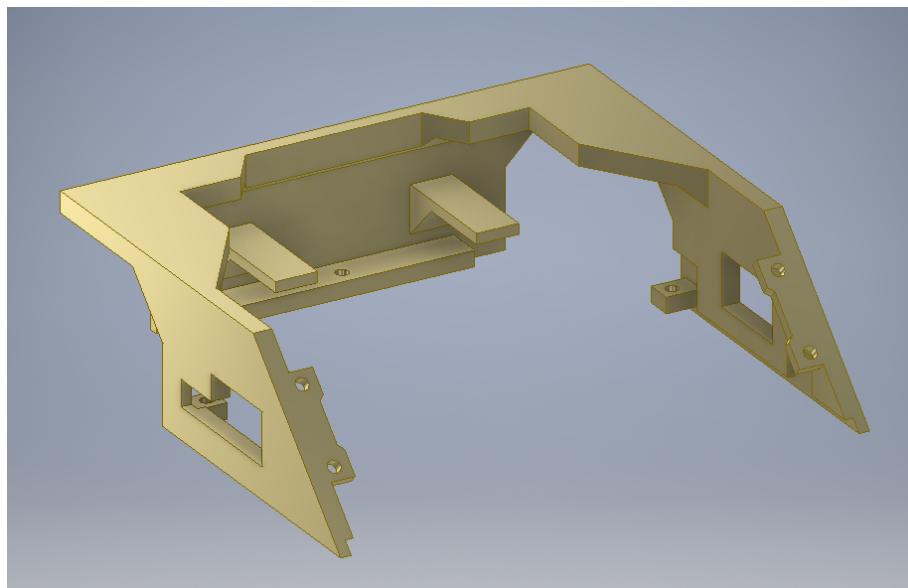
4.1.1. Założenia

Głównym założeniem konstrukcyjnym było zminimalizowanie szansy podbicia robota przez przeciwnika. W związku z czym napęd robota składa się z dwóch kół umiejscowionych w tylnej

części konstrukcji. Przednia część robota zakończona jest pługiem, który bezpośrednio dotyka podłoża. Dodatkowo starano się, aby środek ciężkości całej konstrukcji znajdował się jak najbliżej podłoża.

4.1.2. Nadwozie

Konstrukcja nadwozia została zaprojektowana przy użyciu środowiska *Autodesk Inventor 2017*. Wybór został podektywany łatwością obsługi narzędzi oraz darmową licencją studencką. Obudowa została wykonana w technologii druku 3D, ze względu na małą masę, koszt produkcji, oraz możliwość łatwego dostosowania do potrzeb projektu (otwory na śruby oraz czujniki przeciwnika, przestrzeń na koła).

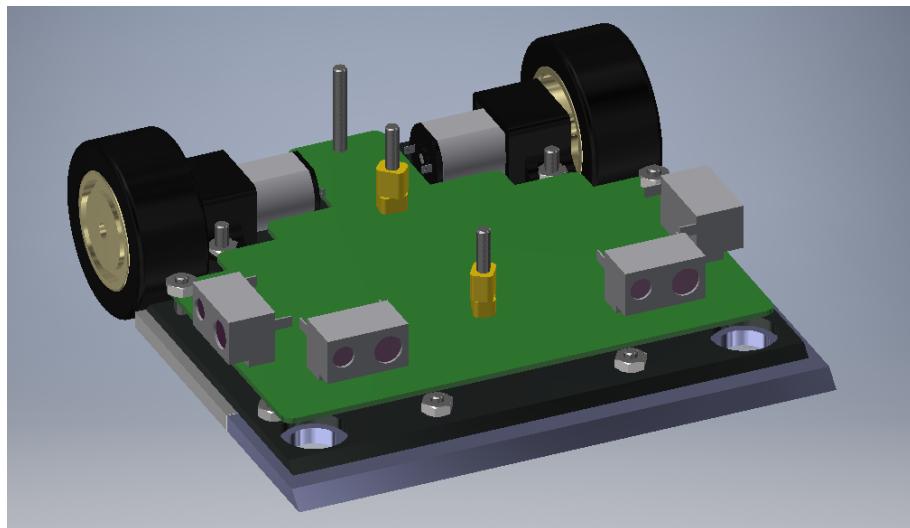


Rys. 4.2: Projekt nadwozia w środowisku Autodesk Inventor 2017.

Na ilustracji 4.2 znajduje się projekt nadwozia robota. Ze względu na niewielką wytrzymałość plastikowego wydruku przywiązano dużą wagę do szerokości obudowy. Starano się, aby była możliwie jak najszersza, co skutkowało wzrostem trwałości całej konstrukcji. Dodatkowo front robota został wykonany ze stali, ponieważ w przypadku podważenia przeciwnika jest obszarem najbardziej podatnym na zniszczenia.

4.1.3. Podwozie

Całość podwozia wykonana została ze stali ze względu na wytrzymałość oraz dużą masę, która przyczyniła się do znacznego obniżenia środka ciężkości całej konstrukcji. Podobnie jak w przypadku nadwozia projekt został stworzony w środowisku *Autodesk Inventor 2017*. Konstrukcja składa się z trzech płyt, z czego jedna zawiera ostrze od strugarki, wykonane z węglika spiekaneego. Materiał ten cechuje się wysoką wytrzymałością oraz większą odpornością na kruszenie w porównaniu do zwykłej stali.

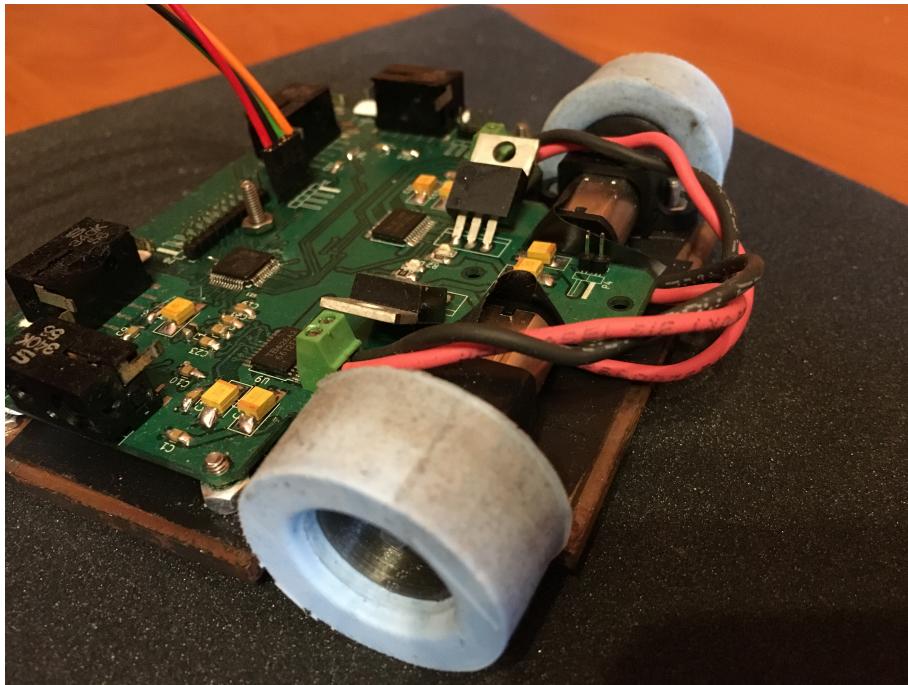


Rys. 4.3: Projekt podwozia w środowisku Autodesk Inventor 2017.

Ilustracja 4.3 ukazuje wizualizację podwozia wraz z silnikami oraz kołami. Projekt wyeksportowano do formatu 2D, a następnie odpowiednie kształty wraz z otworami zostały wykonane przy użyciu techniki wypalania laserem.

4.1.4. Napęd

Jak wspomniano wcześniej, napęd robota składa się z dwóch kół, które sterowane są niezależnie. Ruch robota wzorowany jest na zasadzie działania czołgu. W przypadku skrętu, do jednego z silników dostarczany jest większy prąd, co skutkuje wzrostem prędkości kątowej napędzanego koła. Z powodu różnicy prędkości robot zaczyna skręcać. Zdecydowano się na takie rozwiązanie ze względu na dozwoloną masę (większa ilość kół oraz silników znacząco zwiększyłaby masę całej konstrukcji) oraz manewrowość – dzięki zastosowaniu omawianego rozwiązania możliwy jest obrót robota w miejscu (w przypadku, gdy koła kręcą się w przeciwnie strony), dzięki czemu pojazd jest szybki oraz zwinny.



Rys. 4.4: Napęd robota minisumo.

Na zdjęciu 4.4 przedstawiono napęd robota na który składają się dwa silniki oraz dwie felgi z oponami. Jako jednostkę napędową użyto silników *Pololu HPCB* ze względu na rozmiary, dopuszczalne napięcie zasilania oraz korzystny stosunek ceny do jakości. Felgi zostały wykonane przy pomocy wydruku 3D, a opony były gotowym komponentem wykonanym z poliuretanu.

4.2. Elektronika

Blabla . . .

4.2.1. Założenia

4.2.2. Źródło zasilania

4.2.3. Procesor

4.2.4. Sensoryka

4.2.5. Sterownik silników

4.2.6. Schemat płytki z interfejsem

4.2.7. Schemat płytki głównej

4.3. Oprogramowanie

4.3.1. Transmisja danych

4.3.2. Obsługa przychodzących wiadomości

4.3.3. Algorytmy walki

Rozdział 5

Aplikacja mobilna

5.1. Platforma

5.2. Kompatybilność

5.3. Wzorzec MVC

5.4. Komunikacja

5.5. Struktura aplikacji

5.5.1. Widok główny

5.5.2. Widok sterowania automatycznego

5.5.3. Widok sterowania zdalnego

5.5.4. Widok diagnostyki

Rozdział 6

Implementacja

6.1. Kompilacja projektu

Rozdział 7

Podsumowanie

7.1. Zrealizowane założenia

7.2. Dalszy rozwój projektu

7.3. Uwagi

Literatura

- [1] *Robot klasy sumo* <https://en.wikipedia.org/wiki/Robot-sumo> (dostęp 09.11.2017).
- [2] *Zawody minisumo* <https://pl.wikipedia.org/wiki/Minisumo> (dostęp 09.11.2017).
- [3] *Kurs HAL* <https://forbot.pl/blog/kurs-stm32-f4-1-czas-poznac-hal-spis-tresci-kursu-id14114> (dostęp 09.11.2017).
- [4] *Dokumentacja Apple* https://developer.apple.com/library/content/documentation/Swift/Conceptual/Swift_Programming_Language/index.html (dostęp 09.11.2017).
- [5] *Arduino* <https://www.arduino.cc/> (dostęp 09.11.2017).