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:

WROCŁAW, 2017

Spis treści

Spis rysunków

Spis listingów

Rozdział 1

Wstęp

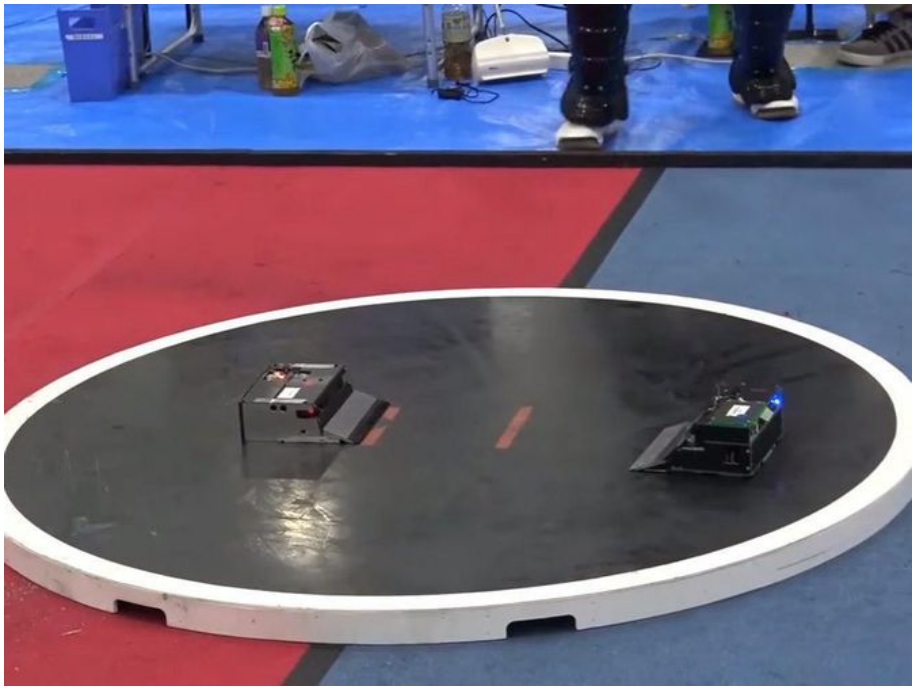
Celem niniejszej pracy jest implementacja aplikacji mobilnej służącej do sterowania robotem minisumo. W ramach pracy dyplomowej powstał samodzielnie wykonany dwukołowy robot w pełni spełniający wymagania do startu w zawodach minisumo. Dodatkowo powstała aplikacja mobilna na platformę iOS, która daje 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 sędziego,
- 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.

Rozdział 2

Wykorzystane technologie

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

2.1. C

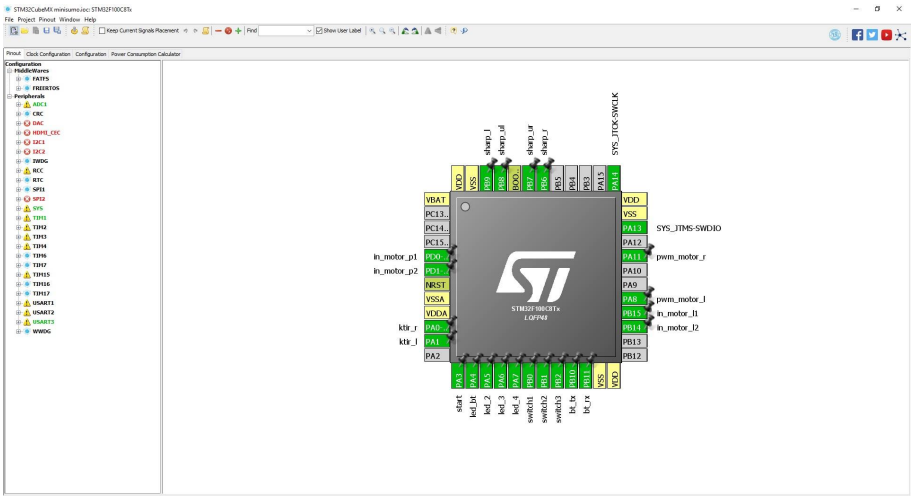
Z racji, iż sercem robota jest procesor z rodziny STM, wybór technologii został ograniczony do języka C lub C++. Wybrano język C z powodów optymalizacyjnych oraz małego stopnia skomplikowania programu.

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 bilbiotekę 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.

Na rysunku 2.1 zilustrowano konfigurację peryferiów użytego procesora w środowisku CubeMX.



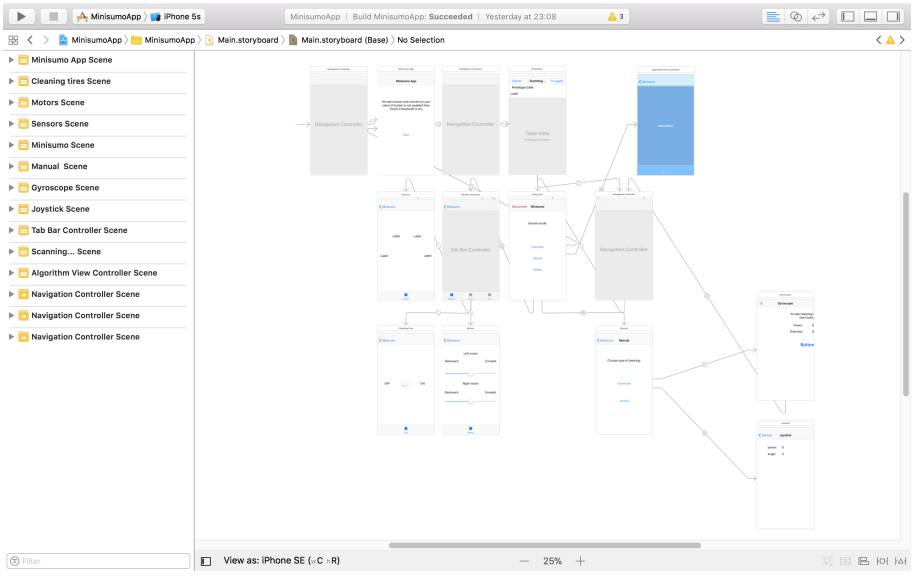
Rys. 2.1: Konfiguracja peryferiów użytego procesora STM32F100C8T6B - LQFP48.

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 użytym 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.

Na rysunku 2.2 zilustrowano 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, zarządzanie tekstem i wyświetlaniem, obsługę aplikacji oraz rozszerzeń aplikacji.

`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 podklasą *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 *CoreGrapics* zostało wykorzystane do stworzenia wirtualnego dżojstiku służącego zdalnemu sterowaniu robotem minisumo.

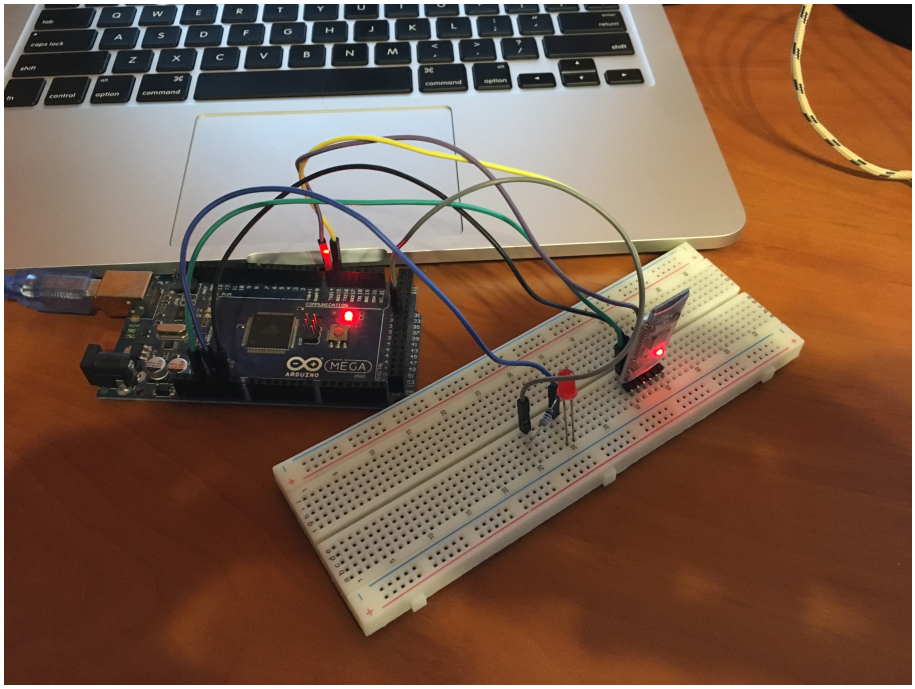
2.2.4. CoreMotion

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 żyroskopu.

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. Aczkolwiek 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 (*baud rate*. Do tego celu użyto mikrokontrolera *Arduino Mega 2560* oraz płytki 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.

Na rysunku 2.3 zilustrowano układ Arduino Mega 2560 wraz z płytką stykową oraz podłączonym modułem bluetooth. Dodatkowo układ wyposażono w diodę mającą na celu sygnalizację poprawności wgrywania komend.

Rozdział 3

Komunikacja

3.1. Moduł bluetooth

3.2. Logika

Rozdział 4

Robot minisumo

4.1. Konstrukcja

4.1.1. Nadwozie

4.1.2. Podwozie

4.1.3. Napęd

4.2. Elektronika

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).