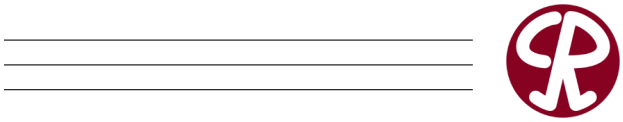
Projekt

Wizualizacja Danych Sensorycznych

Ślepy Micromouse

Mateusz Kobak, 241502



Prowadzący:
dr inż. Bogdan Kreczmer

Katedra Cybernetyki i Robotyki
Wydziału Elektroniki
Politechniki Wrocławskiej

30 czerwca 2020

Spis treści

1 Charakterystyka tematu projektu **1**

2 Podcele i etapy realizacji projektu **1**

3 Specyfikacja finalnego produktu **2**

4 Projekt graficznego interfejsu użytkownika **2**
4.1 Podział na sekcje 2

5 Interfejs komunikacyjny **3**
5.1 Opis algorytmu przesyłu danych 4

6 Wizualizacja danych **5**
6.1 Warstwa połączeniowa, lvl-0 6
6.2 Warstwa wizualizacyjna, lvl-1 6
6.3 Warstwa sterowania, lvl-2 6
6.4 Aplikacja 7

7 Warstwa sterowania i algorytmu **7**
7.1 Okno labiryntu 8
7.2 Program Robota 8
7.3 Algorytm Micromouse 11

8 Terminarz realizacji poszczególnych podcelów (z dokładnością do 1 tygodnia) **13**

1 Charakterystyka tematu projektu

Głównym celem projektu jest stworzenie robota Micromouse który będzie posiadał tylko akcelerometr i żyroskop[1] do rozpoznawania położenia i przeszkód. Posiadać on będzie pierścień zamontowany na sprężynie dająca niewielkie możliwości ruchowe, w momencie uderzenia ze ścianą pierścień odchyli się a czujnik to wykryje. Robot jest też projektem na przedmiot Roboty Mobilne. Aby komunikować się z komputerem potrzebne jest urządzenie pośredniczące które będzie również wyposażone w moduł radiowy i interfejs usb aby można było dane przesyłać do komputera. Urządzenie pośredniczące to projekt na Sterowniki Robotów, jest to joystick który posiadać będzie interfejs USB, Wifi, Bluetooth i moduł radiowy, będzie więc idealnym łącznikiem Robot - Komputer.

2 Podcele i etapy realizacji projektu

Głównym problem jest zaprogramowanie poprawnej komunikacji między aplikacją a robotem. W robocie użyto modułu RF[2] do komunikacji. Wizualizacja labiryntu jest kolejnym zadaniem, do tego potrzebny jest projekt graficzny aplikacji jak i sam program główny który będzie sterował całością.
Wyróżnione zostały dwa kamienie milowe:

- Poprawna komunikacja
- Wizualizacja labiryntu

Lista podelów:

- Przegląd literatury i zasobów Internetu związanych z tematem projektu
- Projekt układu elektronicznego (schemat ideowy)
- Projekt obudowy
- Montaż robota
- Oprogramowanie podstawowe (sterowniki)
- Implementacja komunikacji z Joystickiem i komputerem
- Algorytm właściwy robota
- Projekt graficzny aplikacji
- Komuniakcja między aplikacją a Joystickiem
- Algorytm działania aplikacji

3 Specyfikacja finalnego produktu

Celem projektu jest utworzenie aplikacji która będzie wizualizować dane sensoryczne uzyskane od robota jak i wszelkie informacje przebiegu algorytmu rozpoznającego labirynt w którym robot się znajduje i informacje o stanie baterii. Dodatkowo będzie funkcja sterowania robotem manualnie. Docelowo aplikacja będzie dostępna na komputer z Windowsem, możliwe jednak że uda się zaimplementować też na Androida, co byłoby bardzo pomocne w terenie. Projekt ten będzie wymagał zaawansowanej interpretacji danych z akcelerometru i żyroskopu, specjalnego algorytmu[3] identyfikacji labiryntu i znalezienia najszybszej ścieżki aby labirynt ukończyć. Sama aplikacja będzie złożona acz będzie jedynie ekranem na to co dzieje się wewnątrz robota, pozwoli to na dokładne zrozumienie działania jak i proste wykrywanie ewentualnych błędów.

4 Projekt graficznego interfejsu użytkownika

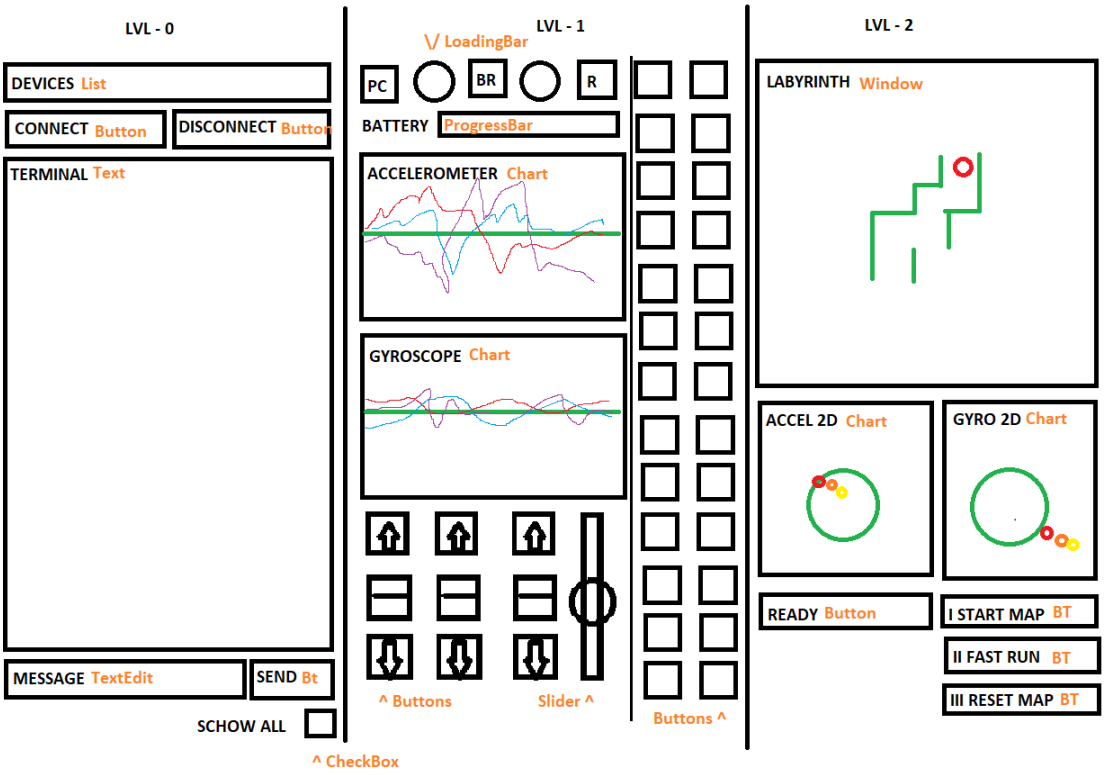
Głównym celem aplikacji jest wyświetlanie informacji uzyskanych z robota, jednak dane te można pogrupować względem istotności i szczegółowości. Projekt zakłada podział interfejsu na oddzielne 3 sekcje, niewykluczone że okaże się potrzebna sekcja czwarta. Każda sekcja odpowiedzialna jest za wyższą warstwę i poziom przetworzenia informacji, rozwiązanie takie ma pewne istotne zalety:

- Stopniowe rozwijanie aplikacji i prostota w sprawdzaniu błędów warstwy wyższej
- Możliwość użycia aplikacji w innych projektach, najniższa warstwa posiada wiele standardowych funkcjonalności
- Prostota w rozbudowie, wystarczy dodać kolejną warstwę
- Łatwość obsługi, wybiera się te warstwy na których się chce pracować, brak natłoku kontrolek itp

4.1 Podział na sekcje

Sekcje są podzielone na poziom przetworzenia informacji jakie są przesyłane pomiędzy urządzeniem a komputerem, każda wyższa warstwa używa niższej do wymiany informacji, lecz interpretuje je na wyższym poziomie. Każdą sekcję będzie można ukryć lub pokazać zależnie od potrzeb, możliwe że rozbudowa do innych projektów będzie wyglądać na dołączeniu kolejnych sekcji które się będzie wybierało i na nich pracowało. Przewidywany podział na sekcje przedstawia rysunek 1.

- Sekcja 0 – Zawiera ona najbardziej podstawową funkcjonalność, jest to w istocie monitor portu szeregowego, służy do monitorowania przepływu informacji i kontroli poprawności działania całej aplikacji. Zawiera ona interfejs wyboru portu szeregowego i możliwości odbierania i wysyłania danych w postaci ciągu znaków.
- Sekcja 1 – Odpowiedzialna jest za obsługę połączenia między komputerem, mostkiem(joystickiem) a robotem, dodatkowo pokazane jest napięcie bateryjne i w postaci wykresów dane z żyroskopu i akcelerometru. Zawierać będzie też grupę guzików do sterowania robotem i wiele innych guzików odpowiadających za różne



Rysunek 1: Interfejs aplikacji

funkcje robota, nie będą one ściśle zdefiniowane gdyż służyć będą głównie podczas rozbudowy algorytmu samego robota. Warstwa ta nadal ma funkcjonalność pozwalającą na użycie aplikacji w innych projektach.

- Sekcja 2 – Zadaniem tej sekcji będzie natychmiastowe połączenie się z robotem i kontrola algorytmu przeszukiwania labiryntu, większość interfejsu będzie zajmować okno w którym rysowany będzie labirynt identyfikowany przez robota, oraz ponownie wykresy do wyświetlania danych z akcelerometru i żyroskopu, lecz dane będą odpowiednio przetworzone i bardziej czytelne dla użytkownika, bezpośrednio będą one pokazywały jak działa algorytm. W tym momencie jednak ciężko przewidzieć jakie dokładnie w tej warstwie pojawią się kontrolki, gdyż nie ma jeszcze obrazu całego algorytmu i problemy z nim związane nie są jeszcze jasne.

5 Interfejs komunikacyjny

Robot micromouse posiada moduł radiowy NRF24L01 do komunikacji bezprzewodowej. Do połączenia z komputerem wymagane jest urządzenie pośredniczące, użyty zostanie wielofunkcyjny Joystick który jest realizowany jako projekt na sterowniki robotów. Joystick ten posiada tak samo moduł NRF24L01 i wejście USB przez które urządzenie będzie pracować jako Communication Device Class, czyli przez port szeregowy będzie realizowana komunikacja z komputerem. Moduł NRF ma zaimplementowany system CRC, dodatkowo jest opcja aby potwierdzane było odebranie paczek danych poprzez odesłanie danych zwrotnych lub zwykłego sygnału ACK. Dane przesyłane przez USB

Rysunek 2: Schemat połączeniowy

mają bardzo małe prawdopodobieństwo na błąd i nie ma potrzeby w tym miejscu a ich kontrolę. Moduł NRF pracuje jednocześnie tylko w jednym trybie, odbiornika lub nadajnika. Robot w tym przypadku jest nadajnikiem aby szybko wysyłać dane z czujników do joysticka, potem joystick odsyła te dane do komputera. Aby móc wysyłać dane z komputera i joysticka do robota, zastosowany został tryb potwierdzania odebrania danych z odesłaniem danych zwrotnych zamiast samego potwierdzania sygnałem ACK. Schemat połączeniowy pokazany jest na rysunku 2.

5.1 Opis algorytmu przesyłu danych

Potrzebne jest aby odbierać dane z akcelerometru i żyroskopu, w tym celu został zaprojektowany specjalny algorytm przesyłania danych. Paczka danych jest zdefiniowana jako tablica pojedynczych bajtów, sam interfejs USB jak i komunikacja radiowa wspiera wysyłanie bajta po bajcie. Pierwszy bajt mówi o funkcji całej paczki, czyli mówi jakie to będą dane i ile bajtów cała paczka będzie zawierać. Aplikacja jak i robot musi znać definicje każdej funkcji, w przypadku nie rozpoznania funkcji, aplikacja informuje o otrzymaniu nieznanej funkcji, funkcje rozpoznane nie pokazują się w oknie portu szeregowego gdyż służą do interpretacji przez funkcje wyższych warstw, np. rysowanie wykresów. Znajomość długości całej paczki danych ma sens ponieważ zdarza się niezwykle rzadko że nie dociera cała paczka danych, w przypadku dotarcia niekompletnej funkcji aplikacja sygnalizuje tą sytuacje. Komunikacja taka zazwyczaj jest realizowana znakowo i nie dało by się wtedy całkowicie rozróżnić paczki danych z funkcją od zwykłych danych, na przykład wysyłanych do urządzenia w celach kontrolnych. Trzeba wtedy uznać jakieś znaki są zakazane i używane wyłącznie do odpowiednich funkcji, dodatkowo taka komunikacja jest znacznie wolniejsza i wymagała by niepotrzebnej konwersji wartości na ciąg znaków i zmiennej długości paczki co jest problemem ponieważ nadajnik i odbiornik definiują przed rozpoczęciem komunikacji jakiej długości są paczki, dodatkowo przy szybkim odczytywaniu danych w czasie rzeczywistym nie można sobie pozwolić na takie rozwiązanie. Długość paczki wysyłanej i odbieranej została zdefiniowana jako długość najdłuższej funkcji i będzie dostępna możliwość zmiany zdefiniowanej długości ale w sytuacjach gdy najdłuższa paczka przesyłana w jakimś momencie będzie miała inny rozmiar a potrzeba będzie by zoptymalizować przesyłanie jej. Taka implementacja jest wymuszona przez sam układ NRF i w momencie wysłania paczki krótszej niż zdefiniowana długość, wysyła się i tak maksymalna ilość bajtów. W momencie gdy paczka danych nie zawiera funkcji to wtedy są po prostu znaki jeden za drugim i zazwyczaj jest używana do ustawiania jakiś parametrów czy załączania odpowiednich trybów w robocie lub w joysticku. Tablica ASCii przewiduje wartości od 0 do 127, bajt odpowiedzialny za funkcje przyjmuje więc wartość między 128 a 255. Kolejne bajty i ich zastosowanie jest zdefiniowane przez definicje funkcji i odpowiednio te dane są wtedy przetwarzane. Opis podstawowych funkcji:

Funkcja FUNC_ACCEL_GYRO_DATA

Posiada wartość 128, jej celem jest wysłanie z robota do komputera danych z żyroskopu i akcelerometru. Dane z czujnika są długości 16 bitów i mogą przyjmować wartości ujemne, dodatkowo akcelerometr jak i żyroskop są trójosiowe, więc daje to 6 wartości 16 bitowych. Wartości te trzeba rozdzielić na bajty które można dopiero wysłać dalej. Paczka danych jest więc złożona z 13 bajtów, gdzie kolejno pierwszy jest numerem funkcji, kolejne odpowiednio osiami x, y, z akcelerometru i tak samo dla żyroskopu. Zamiana 16 bitowej wartości odbywa się poprzez przepisanie młodszych 8 bitów do pierwszej ramki a do następnej przypisane są bity starsze. Konstrukcje paczki danych pokazuje kod:

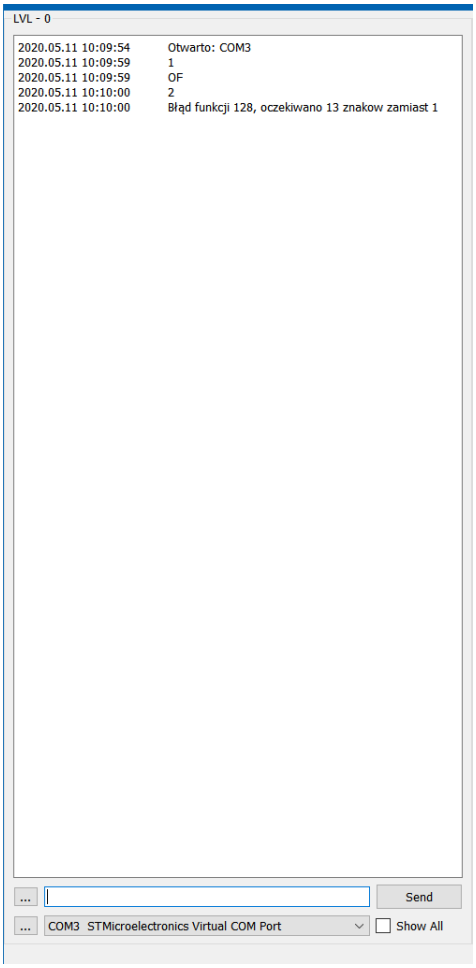
```
Msg[0]  = FUNC_ACCEL_GYRO_DATA;
Msg[1]  = AccelData.x;
Msg[2]  = AccelData.x >> 8;
Msg[3]  = AccelData.y;
Msg[4]  = AccelData.y >> 8;
Msg[5]  = AccelData.z;
Msg[6]  = AccelData.z >> 8;
Msg[7]  = GyroData.x;
Msg[8]  = GyroData.x >> 8;
Msg[9]  = GyroData.y;
Msg[10] = GyroData.y >> 8;
Msg[11] = GyroData.z;
Msg[12] = GyroData.z >> 8;
```

Funkcja FUNC_JOYSTICK_DATA

Funkcja ta odpowiada za wysyłanie informacji o stanach przycisków i gałek w joysticku do robota lub komputera. Robot otrzymać tą paczke może tylko jako informacje zwrotne wcześniej nadanych danych. Pierwszy bajt jest funkcją a następnie wartość odpowiadająca pochyleniu gałki lewej, dwa bajty na współrzędną x i dwa na y. Bajt piąty jest bardziej złożony, poszczególny bit jest odpowiedzialny za oddzielny przycisk po stronie lewej joysticka. Kolejne bajty są analogicznie dla strony prawej. łącznie paczka zawiera 11 bajtów. Aktualnie funkcja jest używana do manualnego sterowania robotem z joysticka.

6 Wizualizacja danych

W trakcie realizacji projektu okazało się że aplikacja ma większy potencjał niż sama wizualizacja danych, zostanie ona użyta do samego sterowania robotem i zaimplementowany w niej będzie cały algorytm działania. Wynika to z faktu większej prostoty i przejrzystości programowania w QT designer, dodatkowo okazało się że połączenie między robotem a komputerem jest na tyle stabilne i szybkie (koło 200 pakietów na sekundę) że sterowanie zewnętrzne będzie możliwe. Z powodu tej zmiany warstwa wizualizacji (lvl-1) nie będzie posiadała takiej dużej ilości kontrolek jak przewidywano, zamiast tego poziom 2 będzie odpowiedzialny za uruchamianie i kontrole algorytmu.



Rysunek 3: Warstwa połączeniowa

6.1 Warstwa połączeniowa, lvl-0

Sekcja ta odpowiedzialna jest za wybór i połączenie się z urządzeniem, posiada też terminal i okno podglądu płynących informacji , co najważniejsze ma zaimplementowany protokół i rozpoznaje informacje przychodzące które są do warstw wyższych i tam je przekazuje. Interfejs przedstawia rysunek 3.

6.2 Warstwa wizualizacyjna, lvl-1

Warstwa ta w większości zajmowana jest przez dwa wykresy, górny przedstawia pomiar z akcelerometru a dolny z żyroskopu, oba wykresy można przełączyć w tryb całkowania i dostosować współczynnik zapominania, dodatkowo posiada kilka kontrolek służących do testów nowych funkcji w robocie, wskaźnik pokazujący prędkość przesyłu w pakietach na sekundę i możliwość zatrzymania wykresu. Warstwę wizualizacyjna przedstawia rysunek 4.

6.3 Warstwa sterowania, lvl-2

robot który będzie sterowany jest projektem na przedmiot Roboty Mobilne, aktualnie do ukończenia projektu zostało utworzenie algorytmu sterowania, dzielić będzie się na dwie części:

Rysunek 4: Warstwa wizualizacyjna

- Wykrywanie ścian i położenia względem labiryntu
- Algorytm micromouse do przeszukiwania labiryntu i dotarcia do końca labiryntu

Obie te części zostaną zaprojektowane najpierw w tej aplikacji, najprawdopodobniej pierwsza część zostanie przeniesiona do programu samego robota bezpośrednio aby zwiększyć jej skuteczność, reszta sterowania zostanie w aplikacji. Największym wyzwaniem w tym miejscu będzie właśnie poprawna identyfikacja ścian i położenia robota względem nich.

6.4 Aplikacja

Komunikacja z robotem w dwie strony została w pełni zaprogramowana, wystarcza w pełni aby utworzyć algorytm sterujący. Urządzenie pośredniczące w komunikacji jest realizowane na projekt z Sterowników robotów i jest w pełni funkcjonalne. Okno aplikacji przedstawia rysunek 5.

7 Warstwa sterowania i algorytmu

W aplikacji została rozbudowana ostatnie warstwa w której kontrolować można algorytm sterowania robotem. Na obrazku 5 zaznaczono kolejno :

- 1. - Okno podglądu pozycji robota i całej siatki labiryntu wraz z scianami.
- 2. - Okno do podglądu każdego ruchu i działania algorytmu.
- 3. - Ustawienia dla elementarnych ruchów robota.
- 4. - Kontroli sterowania algorytmem i generowania labiryntu. Opcja

- 5. - Okno podglądu przechylenia pierścienia robota, służy do korekcji obrotu podczas zderzenie.

7.1 Okno labiryntu

Szczegółowego opisu wymaga wizualizacja labiryntu. Generować można labirynt minimalnie o wymiarach 3x3 a maksymalnie 16x16, gdzie górne ograniczenie to standardowy rozmiar labiryntu według regulaminu zawodów Micromouse. Po wygenerowaniu trzeba zaznaczyć punkt startowy i końcowy, lewym klawiszem myszy możemy dodać pojedynczy punkt startowy a lewym klawiszem można dodawać wiele punktów końcowych. Zaimplementowany jest algorytm wyszukiwania ścieżki poprzez numerowanie komórek labiryntu, numerki można podglądać przełącznikiem Show Text. Dodatkowo jest możliwość zaznaczenia Manual aby po włączeniu programu sterującego samemu zaznaczać pola celu do których robot będzie podążać. Kolorem zielonym obrysowywanie jest pole startowe, pomarańczowym kole końcowe, czerwonym pole celu a fioletowym aktualną ścieżkę. Pozycja i orientacja robota względem labiryntu jest pokazana za pomocą strzałki w czerwonym okręgu. Dodatków kółkiem myszy możemy skalować labirynt w oknie. Okno labirytnu przedstawia rysunek 6.

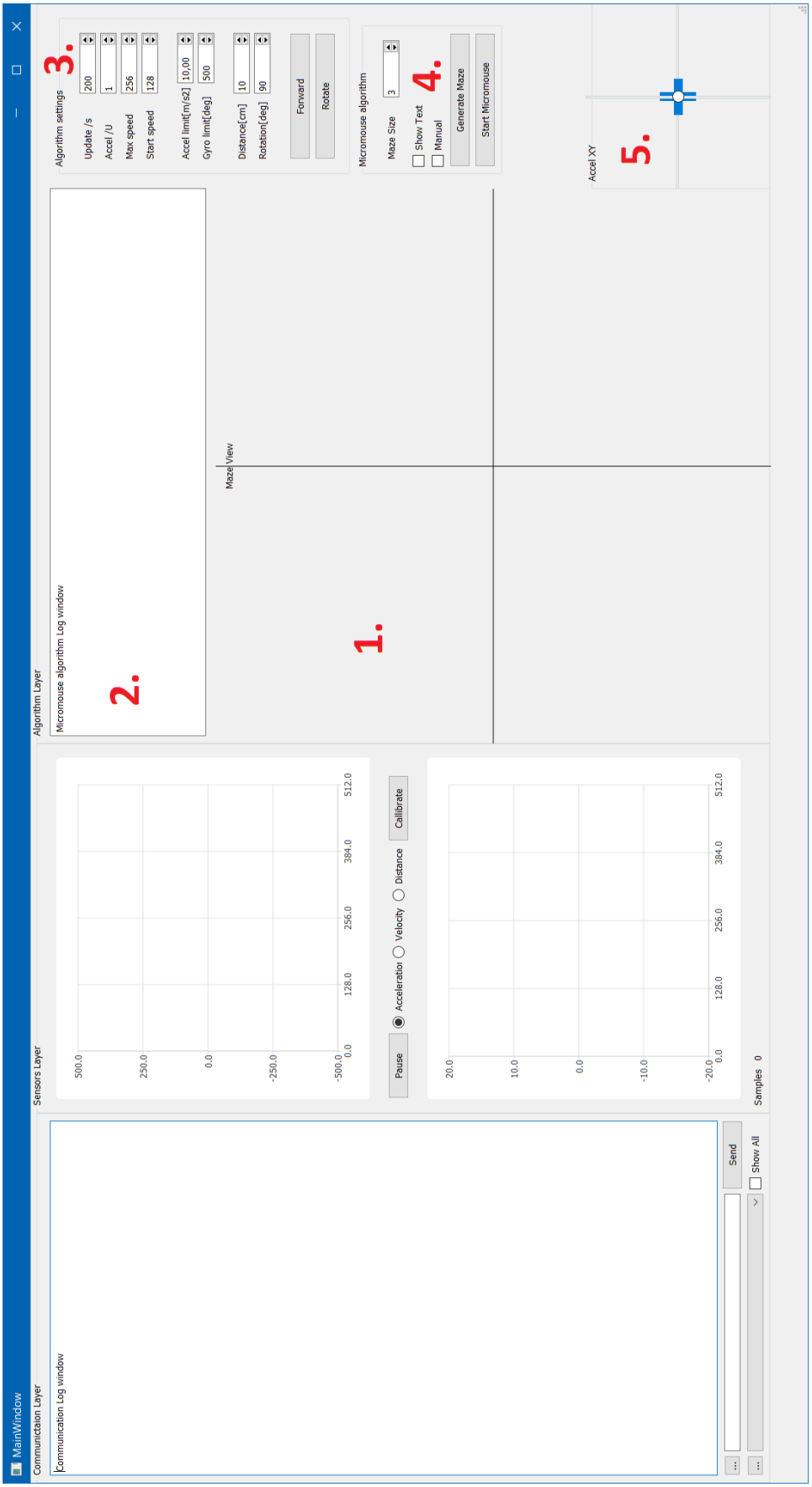
7.2 Program Robota

Robot posiada zaimplementowane elementarne ruchy które wykorzystuje algorytm sterowania, przeniesione zostały one do robota we względu na dokładność jaką sam robot otrzymuje bez komunikacji z komputerem. Komunikacja zwłaszcza radiowa jednak powoduje opóźnienia na tyle duże że ruchy sterowane z komputera okazały się niewystarczająco precyzyjne. Podczas wykonywania przez robota ruchów nie jest możliwe odczytywanie z niego danych z powodu pełnej koncentracji robota na dokładności i braku implementacji nie blokującej komunikacji.

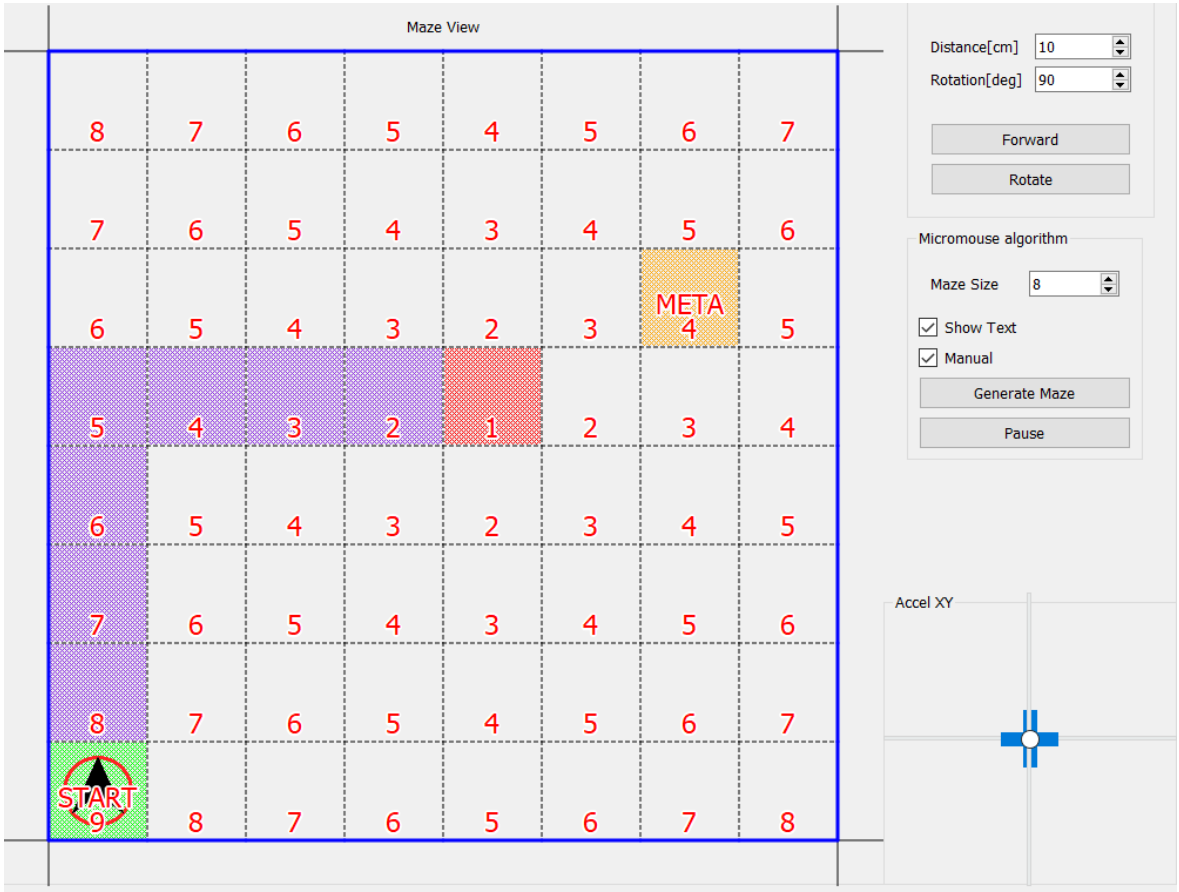
- Jazda do przodu - podawana jest dokładna wartość wyrażona w centymetrach o ile robot ma się przemieścić, dodatkowo warunek stopu który przekłada się na odczyty z akcelerometru mówiące o zderzeniu, przekazywane są też podstawowe informacje o prędkościach i przyspieszeniu. Dodatkowo regulator PD pilnuje aby robot nie zakręcał podczas przemieszczania się. funkcja zwraca przebyty dystans.

- Obrót o zadany kąt - nie zawiera warunku stopu, prędkość maksymalna, minimalna i przyspieszenie też są przekazywane. Funkcja zwraca wykonany obrót.

Dodatkowo jest możliwość używania tych funkcji z okna ustawień algorytmu w celu testowania jakości ustawień.



Rysunek 5: Aplikacja



Rysunek 6: Okno labiryntu

7.3 Algorytm Micromouse

Algorytm ma za zadanie wykrywać ściany w labiryncie i zapamiętywać ich ustawienie w pamięci, po odkryciu całego labiryntu wyznaczana będzie ścieżka optymalna z punktu startowego do centrum labiryntu, czyli mety. Następnie robot będzie musiał przebyć wyznaczoną ścieżkę jak najszybciej [3]. Algorytm został zaimplementowany w aplikacji. Takie rozwiązanie bardzo usprawniło tworzenie algorytmu. Istotne jest to że wymiana danych mimo swej prędkości nie jest wstanie zapewnić dokładności w takich operacjach jak przemieszczanie się czy obracanie, liczenie podwójnej całki z tak otrzymywanych danych byłoby bezużyteczne, robot więc posiada funkcje do przemieszczania się i obrotu samodzielnie, zapewnia to dokładność.
Jazda o zadane przemieszczenie odbywa się poprzez dostarczenie też informacje o wartości progowej przyspieszenia które oznaczałoby zderzenie się z przeszkodą, niezależnie od przyczyny zakończenia ruchu, robot wysyła wartość przemieszczenia jaką udało mu się wyliczyć na podstawie odczytów z akcelerometru. Podczas obrotu nie podaje się warunku stopu. Uzyskana dokładność obrotu jest bardzo zadowalająca, lecz przemieszczanie się jest obarczone takim błędem że nie możliwe jest zadanie robotowi przemieszczenia większego niż 18 cm (rozmiar jednej komórki labiryntu) bez błędu na tyle dużego który zakłóci działanie algorytmu. Problem ten rozwiązuje sposób przemieszczania się w którym robot zawsze przemieszcza się najdalej jak potrafi, algorytm dostając informację zwrotną dopasowywać ją do labiryntu, jednak takie rozwiązanie powoduje następny problem, robot będzie mógł wpaść w pułapki z których nie będzie dało się w ten sposób wyjechać, rozwiązaniem jest połączenie tych dwóch taktyk tak by algorytm sam decydował o sposobie przemieszczania się. Wykonanego robota prezentuje rysunek 7.

Rysunek 7: Wykonany robot

8 Terminarz realizacji poszczególnych podcelów (z dokładnością do 1 tygodnia)

- 9 marca 2020 – zakończenie przeglądu materiałów związanych z danym tematem
- 16 marca 2020 – schemat układu elektronicznego
- 16 marca 2020 – Projekt obudowy
- 23 marca 2020 – Montaż robota
- 30 marca 2020 – Oprogramowanie podstawowe (sterowniki)
- 6 kwietnia 2020 – Implementacja komunikacji z Joystickiem i komputerem
- 18 maja 2020 – Algorytm właściwy robota
- 23 marca 2020 – Projekt graficzny aplikacji
- 6 kwietnia 2020 –Komuniakcja między aplikacją a Joystickiem
- 4 maja 2020 – Algorytm działania aplikacji

Literatura

[1] InvenSense. MPU-6050 Six-Axis (Gyro + Accelerometer) MEMS MotionTracking Devices. Sierpień 2013.

[2] Nordic Semiconductor. nRF24L01+ Single Chip 2.4Ghz Transciever. Preliminary Product Specification v1.0. Marzec 2008.

[3] Robert PIOTROWSKI. Robot typu Micromouse – wykonanie, sterowanie i optymalizacja. Grudzień 2014.

Rysunek 8: Diagram Gantta