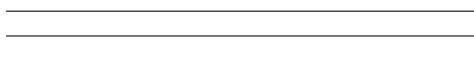


PROJEKT

WIZUALIZACJA DANYCH SENSORYCZNYCH

Wizualizacja Sensorów Line Follower'a

Michał Maćkowiak, 249464



Prowadzący:
dr inż. Bogdan Kreczmer

Katedra Cybernetyki i Robotyki
Wydziału Elektroniki
Politechniki Wrocławskiej

24 czerwca 2021

Spis treści

1	Cel projektu	1
2	Przewidywane efekty końcowe	1
3	Podcele i etapy realizacji projektu	2
4	Terminarz realizacji poszczególnych podcelów (z dokładnością do 1 tygodnia)	2
5	Efekty przeglądu materiałów związanych z projektem	4
6	Projekt interfejsu graficznego użytkownika	4
7	Komunikacja UART	5
7.1	Aplikacja do komunikacji	5
7.2	Przesyłanie informacji przez ramki	6
8	Aplikacja	8
9	Testy	10
10	Dokumentacja i archiwizacja	11
11	Podsumowanie	12

1 Cel projektu

Celem projektu jest stworzenie aplikacji za pomocą biblioteki *Qt* w języku *C++*, która będzie wizualizować dane robota mobilnego klasy *line follower*. Robot wyposażony w enkodery i czujniki odbiciowe będzie przysyłał dane, które posłużą wyświetlaniu trasy przejechanej przez robota, przedstawieniu prędkości oraz odczytu czujników odbiciowych *line follower*'a.

Trasa przejechana będzie zaznaczana punktowo na podstawie położenia robota w przestrzeni dwuwymiarowej, mając informacje z enkoderów zawartych na kołach robota. Dodatkowo enkodery pozwolą przedstawić, z jaką prędkością się porusza.

Czujniki odbiciowe służą detekcji linii, którą ten *line follower*, jak nazwa wskazuje, śledzi. Przedstawiony zostanie stan tych czujników, a zatem informacja czy czujnik wykrywa linię, czy nie.

Dodatkowe informacje o konstrukcji robota mobilnego:

- Wszystkie komponenty robota będą przylutowane lub przymocowane do zaprojektowanej płytki drukowanej.
- Zastosowany będzie mikrokontroler *STM*.
- Będzie posiadać co najmniej 5 czujników odbiciowych. Czujniki odbiciowe wykorzystują do działania diodę emitującą promieniowanie oraz fototranzystor odbierający światło odbite. W ten sposób w zależności od właściwości refleksyjnych i pochłaniających materiału oświetlanego na fototranzystorze emitowane jest napięcie proporcjonalne do ilości otrzymanego światła. Stąd też czarna linia (która pochłania światło) jest odbierana przez mikrokontroler jako sygnał niski natomiast biała przestrzeń toru (odbijająca światło) jako sygnał wysoki.

2 Przewidywane efekty końcowe

- Aplikacja będzie posiadać prosty interfejs użytkownika.
- Komunikacja z robotem, a aplikacją będzie odbywać się poprzez połączenie *bluetooth* 2.0.
- Położenie robota w przestrzeni będzie odwzorowane z dokładnością do 1 cm. Okresowo, co 30 sekund, rysowane będą punkty reprezentujące aktualne położenie robota w przestrzeni *2D*.
- Będzie wyświetlana aktualna prędkość kątowna kół robota z dokładnością do 1 rpm. Graficznie prędkość będzie przedstawiona przez słupki, które będą rosły wraz ze zwiększaniem prędkości oraz malały ze zmniejszaniem jej. Dodatkowo będą zawierać informacje o kierunku jazdy robota.
- Sygnały z czujników będą posiadać interfejs graficzny przypisany do każdego z 5 czujników. Interfejs będzie wyświetlać kolor biały lub czerwony w zależności czy wykrywana jest linia, czy też nie.

3 Podcele i etapy realizacji projektu

W tym rozdziale znajdują się wyszczególnione podcele oraz etapy tworzenia projektu. Każdy etap uznawany jest za zakończony po wykonaniu serii testów oraz ewentualnych poprawek.

Lista podcelów:

1. Zapoznanie się z dokumentacją rozwiązań sprzętowych zastosowanych w projekcie. Pozyskanie dodatkowej wiedzy śledząc artykuły oraz inne zasoby Internetu
2. Projekt interfejsu graficznego
3. Postęp bez gotowego robota:
 - (a) Wykonanie pierwszej wersji aplikacji posiadającej pojedynczy widżet
 - (b) Zaimplementowanie komunikacji przez *UART*, by otrzymywać dane z mikrokontrolera
 - (c) Ręczne symulowanie poruszania silników oraz odczytywanie „surowego” stanu enkoderów dzięki podłączeniu silniki z enkoderami do zewnętrznej płytki deweloperskiej *STM32F429I-DISCO1*
 - (d) Symulacyjne pobieranie danych z czujnika/ów odbiciowych
 - (e) Przetworzenie danych, by otrzymać prędkość kątową kół oraz współrzędne przestrzeni *2D* znając prędkość i kierunek obrotu kół
 - (f) Stworzenie graficznej reprezentacji danych w aplikacji - wyświetlanie trasy przejechanej, aktualnej prędkości oraz stanu czujników odbiciowych
4. Zmontowanie robota (zlutowanie i podłączenie) oraz wstępne testy poprawności jego działania. W tym podłączenie czujników
5. Pełna integracja robota mobilnego z aplikacją. Testy oraz zmiany konstrukcyjne/programowe.
6. Ostateczne sprawdzenie poprawności działania aplikacji przy współpracy z robotem. Skończenie dokumentacji projektu.

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

Poniżej znajduje się harmonogram realizacji projektu. Flagą **KM** zaznaczone są kamienie milowe postępu wykonywania projektu. Data przy, każdym z zadań oznacza datę ukończenia podcelu.

- 22 marca 2021 – zakończenie przeglądu materiałów związanych z projektem
- 29 marca 2021 – zaprojektowanie interfejsu graficznego
- **KM 5 kwietnia 2021** – wykonanie pierwszej wersji aplikacji posiadającej pojedynczy widżet
- 12 kwietnia 2021 – zaimplementowanie komunikacji przez *UART*

- Poniżej na rysunku 1 znajduje się diagram Gantta przedstawiający harmonogram pracy.



5 Efekty przeglądu materiałów związanych z projektem

Zapoznanie się z biblioteką oraz obsługą *Qt* w języku *C++*. [1] [2] [3] [4] [5] [6]

Zapoznanie się z programowaniem mikrokontrolerów STM. [7] [8] [9]

Pozyskanie wiedzy na temat portu szeregowego *UART*, jego implementacji i działania. [10] [11]

6 Projekt interfejsu graficznego użytkownika

Aplikacja będzie posiadać interfejs graficzny umożliwiający przedstawienie w czytelny sposób danych. Projekt interfejsu został przedstawiony na rysunku 2. Funkcjonalności interfejsu przedstawione są poniżej:

- Górny pasek narzędzi:
 - Przycisk "START" - uruchamia on proces łączenia się z robotem, po połączeniu rozpoczyna komunikację oraz odbieranie danych.
 - Przycisk "VIEW" - pozwala on wybrać, które okna są widoczne w aplikacji.
 - Przycisk "EXIT" - odpowiada za wyjście z aplikacji.
- W głównej części aplikacji będą wyświetlane okna odpowiedzialne za wizualizację różnych danych.
 - Okno "Wheels speed" - przy pomocy oddzielnych słupków dla każdego koła wyświetlana jest aktualna prędkość robota. Wzrost słupka odpowiada zwiększeniu prędkości, natomiast spadek słupka odpowiada zmniejszeniu prędkości. Pod słupkami znajduje się wartość liczbowa prędkości obracania się kół. Na środku słupka znajduje się poziom zerowy prędkości. Prędkość oznaczona pod osią zera oznacza prędkość kątową koła skierowaną w przeciwnym kierunku do kierunku jazdy.

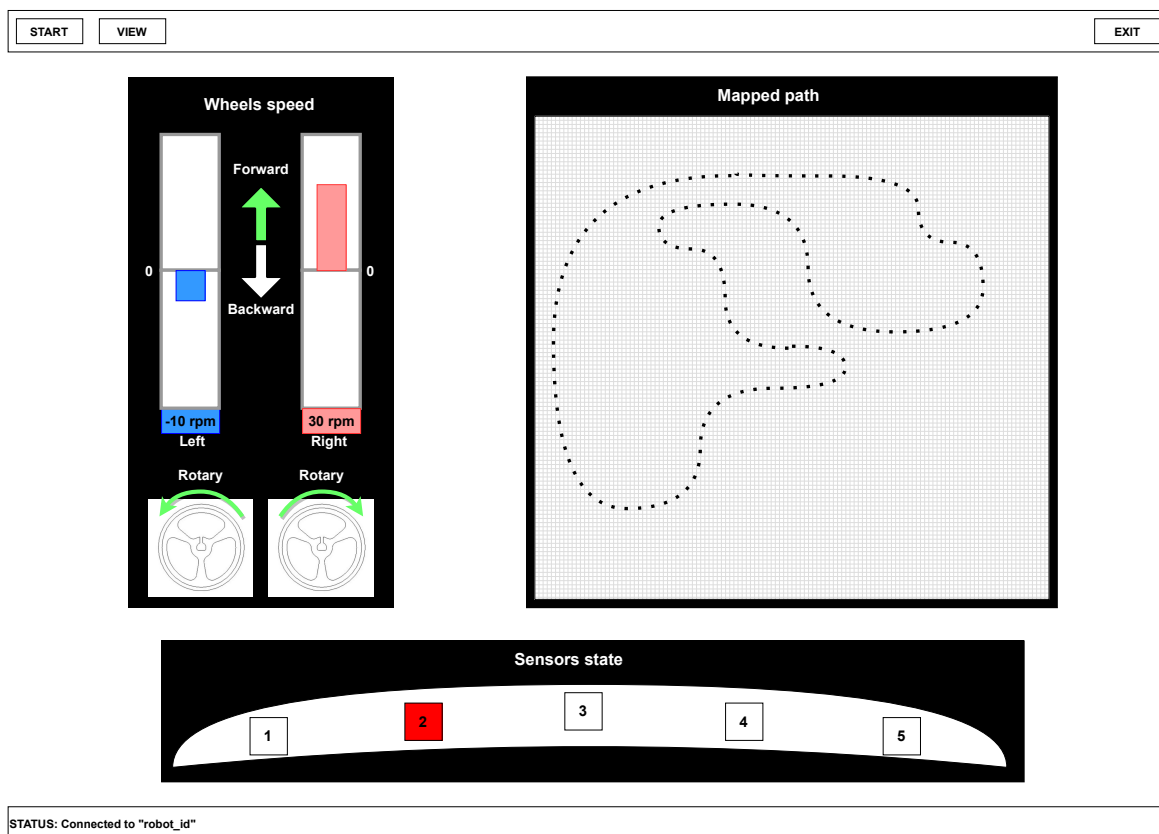
Dodatkowo to okno posiada informacje na temat kierunku jazdy robota mobilnego. Między słupkami znajdować będzie się wertykalna dwukierunkowa strzałka. Zaświecenie (zmiana koloru na zielony) górnej części oznacza jazdę w przód (*forward*), a dolnej jazdę w tył (*backward*).

Na dole okna przedstawione są schematycznie koła robota mobilnego. Nad nimi znajdują się zakrzywione strzałki. Kierunek grotu strzałki oznacza kierunek obrotu koła. Grot skierowany w prawo, odpowiada obrót koła w przód; skierowany w lewo, obrót koła w tył.

Brak świecących strzałek oznacza nie poruszanie się robota.
 - Okno "Mapped path" przedstawiać będzie kwadratowy układ współrzędnych z tikiem co 1 cm. Na nim nanoszone będą punkty pomiaru położenia robota w przestrzeni startując od środka układu. Nałożone punkty będą obrazować trasę przejechaną przez robota.
 - Okno "Sensors state" wizualizować będzie stan 5 czujników odbiciowych robota. Kolor czerwony oznaczać będzie wykrywanie linii, natomiast biały to

stan domyślny, niewykrywania linii. Czujniki reprezentowane są przez ponumerowane kwadraty naniesione na schematyczną platformę robota z czujnikami.

- Na samym dole aplikacji znajduje się informacja o statusie połączenia ("STATUS"). Wyświetlać będzie czy połączenie zostało nawiązane i jeżeli tak to z jakim urządzeniem, w przeciwnym wypadku wyświetli informacje o braku połączenia.



Rysunek 2: Projekt interfejsu graficznego

7 Komunikacja UART

Sprawdzanie komunikacji uart odbieranie oraz wysyłanie informacji testowano na płytce deweloperskiej *STM32F429I-DISCO1*. W końcowej wersji aplikacja będzie łączyć się przez podłączenie *Bluetooth* z robotem.

Została stworzona aplikacja, która umożliwia komunikację.

7.1 Aplikacja do komunikacji

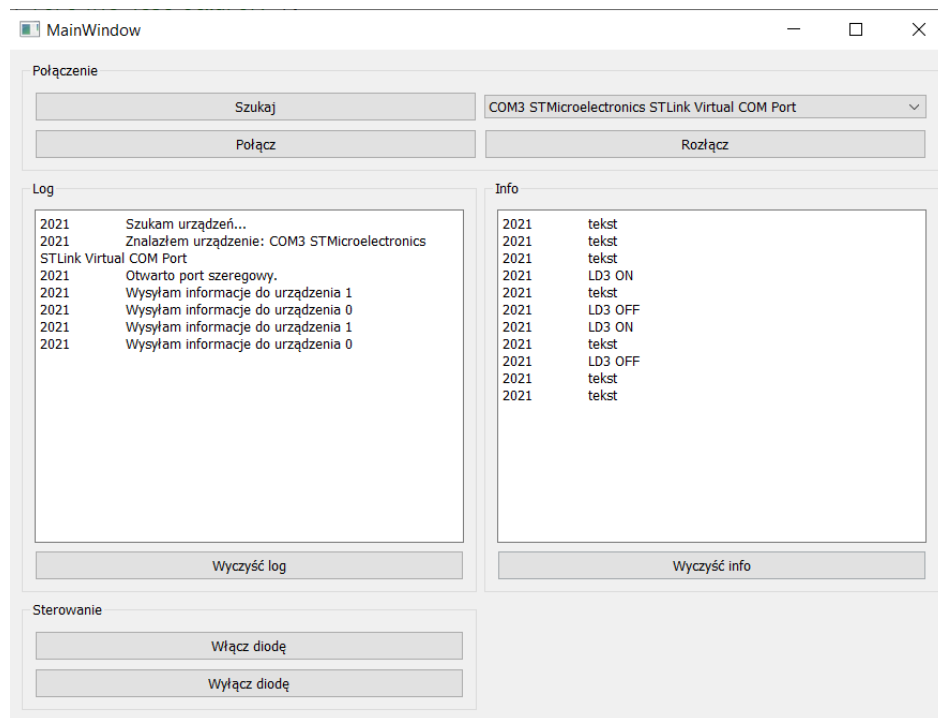
Owa aplikacja posiada wymienione funkcjonalności:

- Przycisk **Szukaj** - wyszukiwanie portów *COM*, do których jest podłączona jest płytka,

- **Lista** urządzeń wyszukanych - wyświetla urządzenia podłączonych wyszukanych po naciśnięciu przycisku **Szukaj**,
- Przycisk **Połącz** - podłącza się do wybranego na liście urządzenia,
- Przycisk **Rozłącz** - rozłącza połączenie z urządzeniem,
- **Log** - pole tekstowe, które służy do wyświetlania dziennik komunikacji (np. informacje o połączeniu, rozłączeniu, błędach),
- **Info** - pole tekstowe, które służy do wyświetlania tekstów odebranych z płytki (np. stany czujników),
- Przycisk **Włącz diodę** - wciśnięcie wysyła sygnał "1" do płytki, który odbiera go i wykonuje akcje zaświecenia diody na płytce,
- Przycisk **Wyłącz diodę** - wciśnięcie wysyła sygnał "0" do płytki, który odbiera go i wykonuje akcje wyłączenia diody.
- Przyciski **Wyczyść ...** - naciśnięcie ich powoduje wyczyszczenie odpowiedniego pola tekstowego.

Przyciski do sterowania diodą w późniejszym etapie może nie być wykorzystane. Aktualnie służyło tylko by sprawdzić wysyłanie i odbieranie informacji przez płytkę.

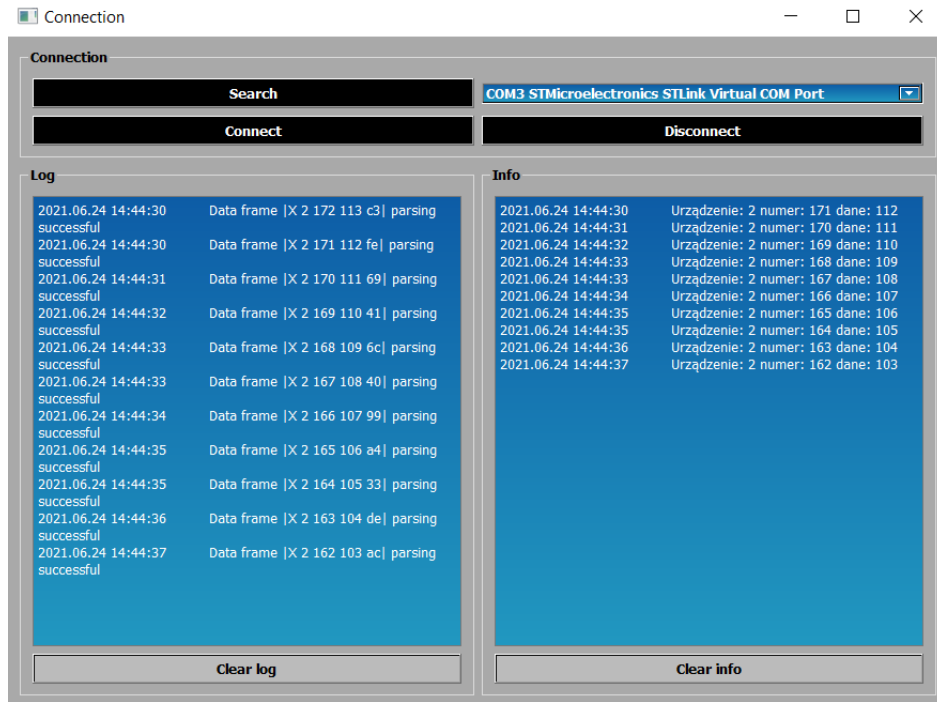
Okno aplikacji można zobaczyć na rysunku 3 poniżej.



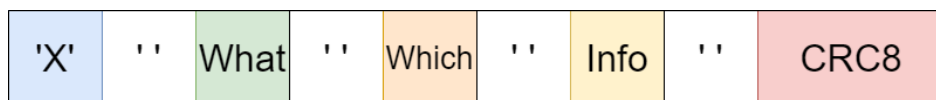
Rysunek 3: Okno aplikacji

7.2 Przesyłanie informacji przez ramki

Komunikacja procesora z aplikacją o stanie czujników oraz silników będzie odbywać się za pomocą ramek. Zdefiniowana ramka przedstawiona jest na rysunku poniżej.



Rysunek 5: Komunikacja przy pomocy ramek



Rysunek 4: Ramka komunikacji

Separatorem informacji w ramce jest *spacja*. Ramka zaczyna się od znaku 'X'. Następnie po separatorze znajduje się komunikat *What* odpowiada on typowi czujnika, z którego dane pochodzą (enkodery czy czujniki transoptorowy). Po następnej spacji (*Which*) znajduje się numer wcześniej wybranego sensora (np. prawy enkoder to 0, a lewy 1). Kolejna spacja oddziela informacje o stanie czujnika lub informacje, którą chce wysłać (np. szybkość). Rozmiar *Info* nie jest stały. Na koniec po ostatniej spacji znajduje się CRC8, czyli hexadecymalna suma kontrolna o rozmiarze dwóch znaków.

Suma kontrolna ma na celu sprawdzenie czy dane zostały poprawnie przesłane przez połączenie kablowe z komputerem.

Wyjątkiem w wysyłaniu danych jest przesyłanie położenia. By to zrobić potrzebne są dwie pozycje na współrzędną x oraz współrzędną y . Z uwagi, że jest to jedno urządzenie, potrzebne dane wysyłane są w dwóch miejscach ramki na pozycji *Which* oraz domyślnej pozycji *Info*.

Dostępne warianty przesyłu dla pozycji *What*:

- 1 – enkodery do przesyłania szybkości,
- 2 – enkodery do przesyłania położenia,
- 3 – sensory do przesyłania stanu czujników.

Wynik działania aplikacji z sumą kontrolną znajduje się na rysunku 5.

Zauważyć można, że aplikacja zmieniła się od pierwszej implementacji. Zlikwidowano przyciski do włączania diod na płycie oraz zmieniono kolory okna.

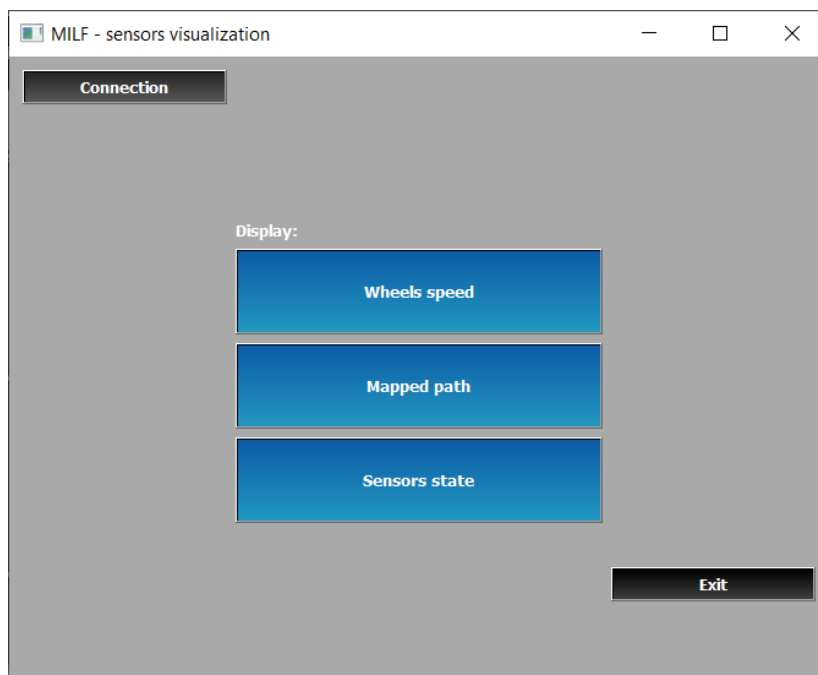
W polu tekstowym *Log* znajdują logi z aplikacji w tym informacja o tym czy ramka została poprawnie sparsowana, jeżeli tak to dane przesłane wczytywane są do wykonywania działań. W przeciwnym wypadku wyświetlana jest informacja i dane są ignorowane.

8 Aplikacja

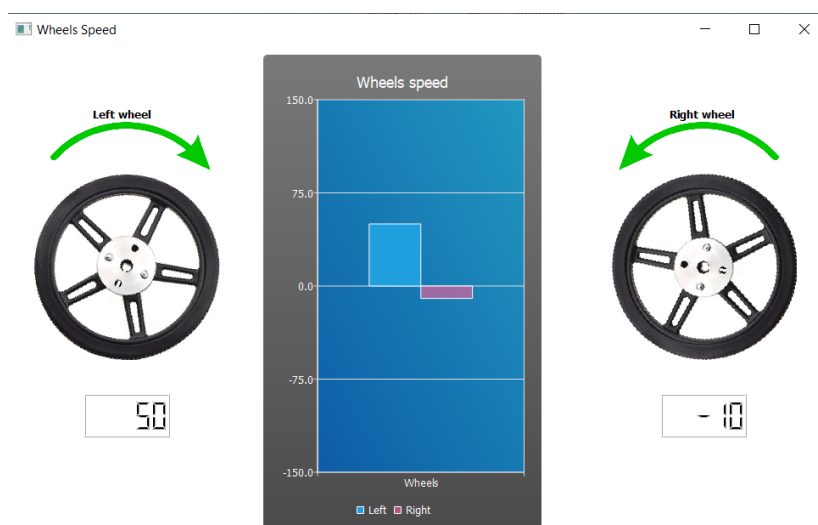
Wykonana została aplikacja, która pozwoli na wizualizację pomiarów. Główne okno, rysunek 6, zawiera przyciski, umożliwiające włączanie kolejnych okien aplikacji. Znajdują się tam następujące przyciski:

- *Connection* – naciśnięcie przycisku spowoduje otworenie okna do łączenia się z urządzeniem poprzez uart. Okno przedstawione jest w rozdziale 7 (rysunek 5),
- *Wheels speed* – naciśnięcie otwiera okno przedstawione na rysunku 7, które zawiera wykres słupkowy szybkości kół robota. Dodatkowo będzie zawierać informacje o kierunku kręceniu się kół. Rysunki kół obracają się w zależności od aktualnej szybkości kół faktycznego robota,
- *Mapped path* – naciśnięcie otwiera okno, które przedstawia na liniowym wykresie śledzoną trasę przejechaną przez robota. Wygląd okna znajduje się na rysunku 8,
- *Sensors state* – naciśnięcie otwiera okno, przedstawione na rysunku 9, które przedstawia stan czujników odbiciowych robota,
- *Exit* – ostatni przycisk znajdujący się na oknie głównym, umożliwia opuszczenie całej aplikacji zamykając wszystkie otwarte okna.

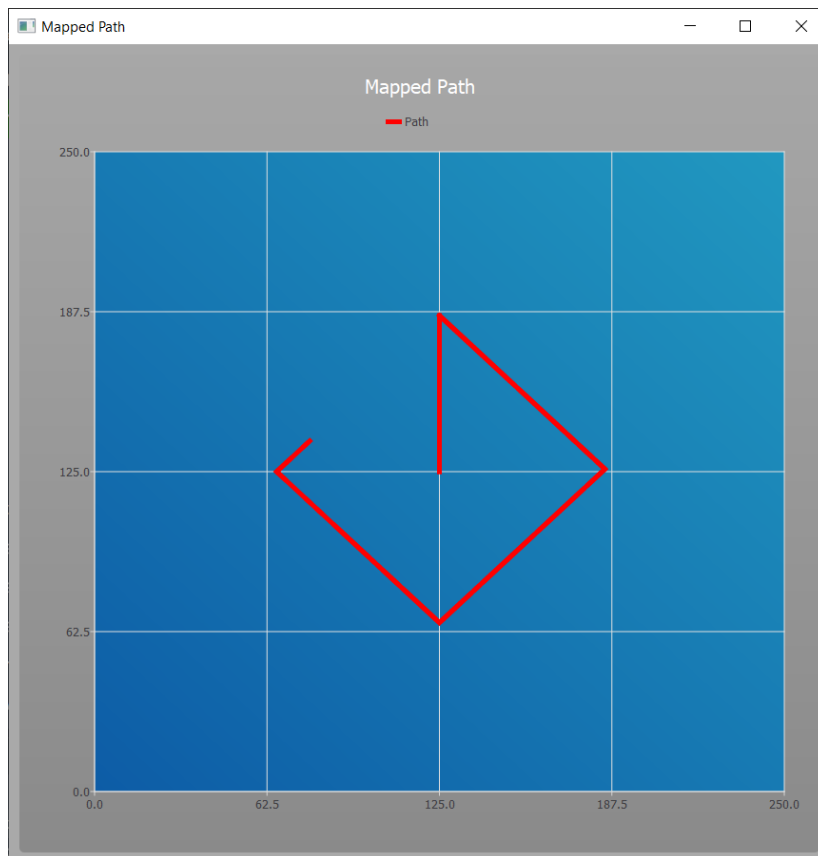
Biblioteka *Qt* pozwala na pełną personalizację swoich aplikacji. Z początku czyta białoszara aplikacja, która nie zachęcała kolorami zmieniała się nie do poznania. Końcowy efekt wizualny jest naprawdę satysfakcjonujący. Zwłaszcza przyciągająca jest możliwość gradientowego kolorowania elementów.



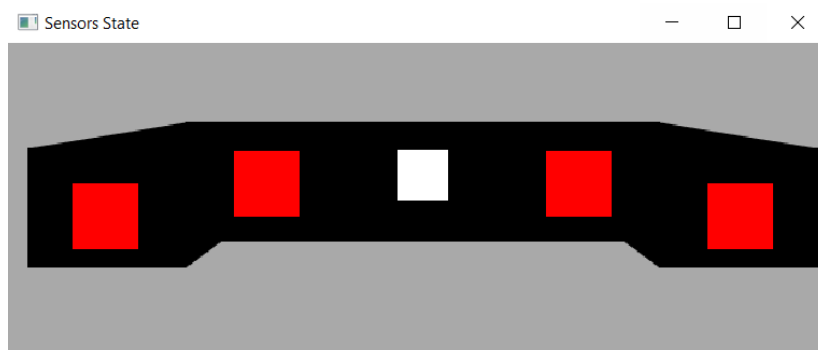
Rysunek 6: Główne okno aplikacji



Rysunek 7: Okno wizualizacji szybkości kół oraz kierunku ich obrotu



Rysunek 8: Okno z wykresem liniowym przejechanej trasy

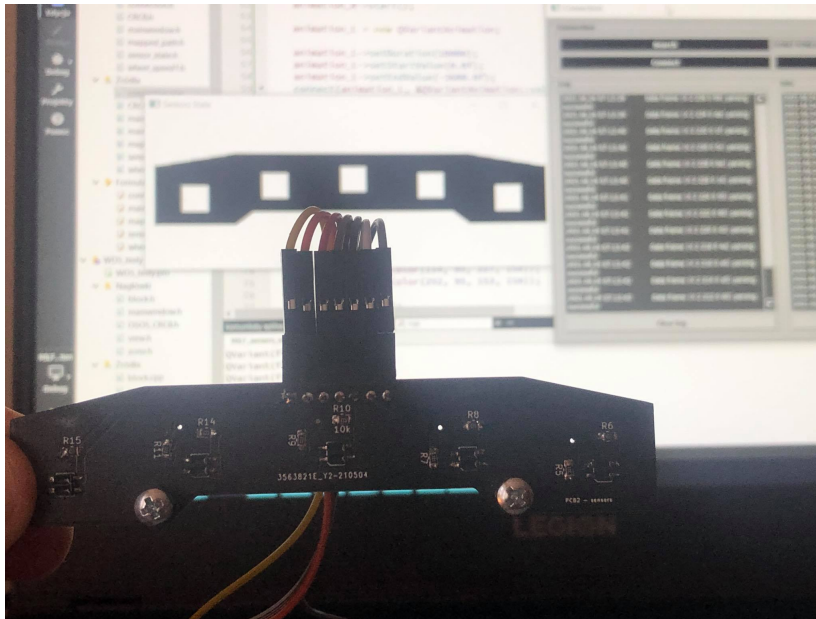


Rysunek 9: Okno wizualizacji stanu każdego czujnika

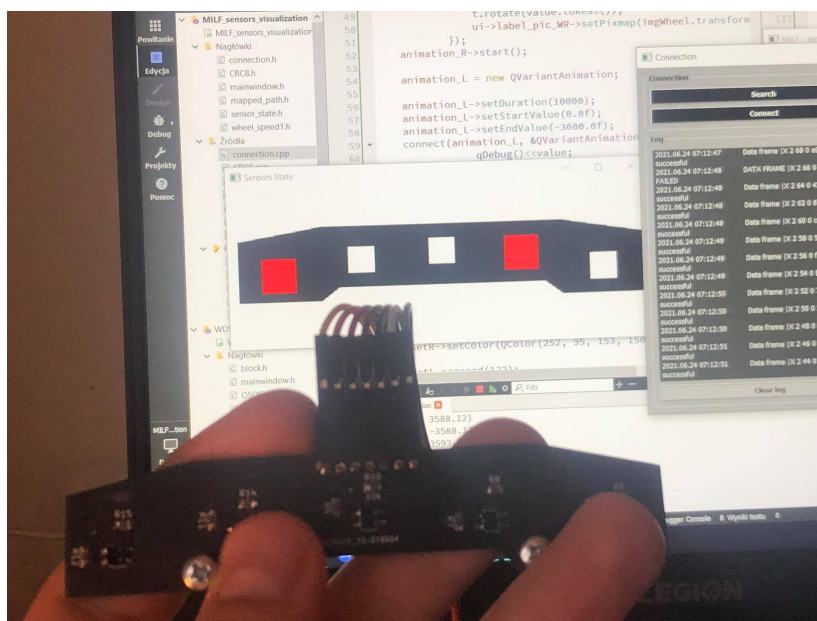
9 Testy

Testowanie aplikacji przebiegało na symulowaniu danych z enkoderów, czyli położenia oraz prędkości i innych informacji o kołach z uwagi na brak sprawnego robota.

Dane z czujników odbiciowych nie były symulowane, gdyż znajdowały się one na osobnej płytce PCB, tzw. wyspie czujników. Sensory te, gdy otrzymywały sygnał zwrotny, czyli powierzchnia jest biała, wtedy czujnik pozostawał w stanie niskim. Jeżeli światło powrotne nie wróciło lub było za słabe to wtedy sensor wysyłał sygnał wysoki. Efekt działania sensorów w aplikacji przedstawiony jest na rysunkach 10 oraz 11.



Rysunek 10: Test sensorów w stanie niskim



Rysunek 11: Test sensorów w stanie wysokim

10 Dokumentacja i archiwizacja

Dokumentacja wygenerowana jest przy pomocy oprogramowania *Doxygen* i graficznego interfejsu *Doxywizard* na podstawie plików z kodem projektu. Głównie w plikach nagłówkowych znajdują się komentarze z opisem funkcji, parametrów, metod oraz zmiennych. Aktualna wersja dokumentacji znajduje się pod adresem strony na serwerze panamint.

Aktualny kod źródłowy projektu oraz dokumentacja wraz z wszystkimi dokumentami w formie pików znajduje się na moim repozytorium na platformie github.

11 Podsumowanie

Biblioteka oraz środowisko programistyczne *Qt* zachęca prostym wyglądem i dużą ilością przykładów. Jednak początki z tym środowiskiem nie są takie łatwe. Parę godzin czytania dokumentacji oraz zapoznawania się z oprogramowaniem udało się wyselekcjonować potrzebne i obiecujące klasy oraz metody potrzebne do wykonania projektu.

Pojawiły się małe problemy, które znacząco spowolniły pracę nad aplikacją oraz odebrały motywację. Przeglądanie zasobów internetu nie pozwoliły skutecznie poradzić sobie z napotkanym problemem. Po paru ciężkich dniach w końcu udało się rozwiązać problem i szybko ruszyć z rozwojem aplikacji.

Późniejsze problemy były możliwie łatwe do rozwiązania, albo wymagały trochę więcej czasu.

Dzięki temu projektowi można było zapoznać się z możliwościami bardzo rozbudowanej biblioteki *Qt*, która posiada znacznie więcej funkcjonalności, niż udało się użyć w projekcie. Stanowczo biblioteka ta będzie służyć do wykonywania innych projektów, prywatnych bądź uczelnianych.

Zaskoczeniem dla mnie była prostota używania wykresów oraz grafik. Bez żadnego problemu, posiłkując się przykładami, można poznać potrzebne metody.

Dodatkowo projekt ten pozwolił zapoznać się bliżej z elektroniką, projektowaniem płytek PCB, lutowaniem ich, programowaniem mikrokontrolerów oraz obsługi różnego rodzaju czujników.

Pozwolił też przypomnieć sobie i utrwalić programowanie większych projektów w języku *C++* oraz zgłębić inne rzeczy takie jak projektowanie własnego protokołu komunikacyjnego.

Literatura

- [1] Mateusz Patyk. Kurs qt. *forbot.pl*.
- [2] Jasmin Blanchette and Mark Summerfield. *C++ GUI Programming with Qt 4*. <http://www.qtrac.eu/C++-GUI-Programming-with-Qt-4-1st-ed.zip>, 2006.
- [3] Dan Munteanu. Robust qt c++ gui programming 2d graphics app tutorial. *udemy.com*, 2020.
- [4] Bryan Cairns. Qt 5 core for beginners with c++. *udemy.com*, 2020.
- [5] Bryan Cairns. Qt 5 core intermediate with c++. *udemy.com*, 2020.
- [6] Bryan Cairns. Qt 5 core advanced with c++. *udemy.com*, 2020.
- [7] Marek Galewski. *STM32. Aplikacje i ćwiczenia w języku C z biblioteką HAL*. BTC, 2019.
- [8] C. Noviello. *Mastering the STM32 Microcontroller*. LeanPub, May 2016.
- [9] Bartek Kurosz. Kurs stm32 f4. *forbot.pl*.
- [10] Kamami. Stm32cube w przykładach (usart). *stm32.com*.
- [11] Bartek Kurosz. Komunikacja przez uart. *forbot.pl*.