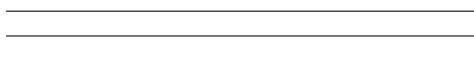


PROJEKT

WIZUALIZACJA DANYCH SENSORYCZNYCH

Wizualizacja figur trójwymiarowych

Adam Kubiak, 249480



Prowadzący:
dr inż. Bogdan Kreczmer

Katedra Cybernetyki i Robotyki
Wydziału Elektroniki
Politechniki Wrocławskiej

14 czerwca 2021

Spis treści

1	Charakterystyka tematu projektu	1
2	Specyfikacja finalnego produktu	1
3	Terminarz realizacji poszczególnych podcelów	1
3.1	Kamienie milowe	2
3.2	Wykres Gantta	2
4	Projekt graficznego interfejsu użytkownika	3
4.1	Dostępne funkcjonalności	3
4.2	Przykładowy scenariusz	4
5	Komunikacja z urządzeniem	4
5.1	Protokół komunikacji	4
6	Działanie programu	4
7	Eksperymenty	8
8	Kod i dokumentacja	10

1 Charakterystyka tematu projektu

Tematem projektu jest wizualizacja trójwymiarowej prostej figury geometrycznej w programie okienkowym napisanym z pomocą biblioteki QT. W tym celu zostanie wykorzystana płytka rozwojowa STM32F411EDISCOVERY z wbudowanym akcelerometrem i żyroskopem, dzięki której użytkownik będzie mógł manipulować położeniem przedstawionego obiektu w programie. Do komunikacji między płytką a komputerem zostanie wykorzystany interfejs UART dodatkowo programie będą wyświetlane wartości rotacji względem każdej z osi w stopniach.

2 Specyfikacja finalnego produktu

Projekt zakłada mierzenie odczytów z akcelerometru i przełożenie tych danych na orientację, wyrenderowanego obiektu 3D za pomocą biblioteki Qt3D. Użytkownik będzie mógł również manipulować danym obiektem za pomocą trzech suwaków, gdzie na jeden suwak będzie przypisana jedna z trzech osi. Dzięki interfejsowi w aplikacji będzie można również odczytać jego aktualną orientację.

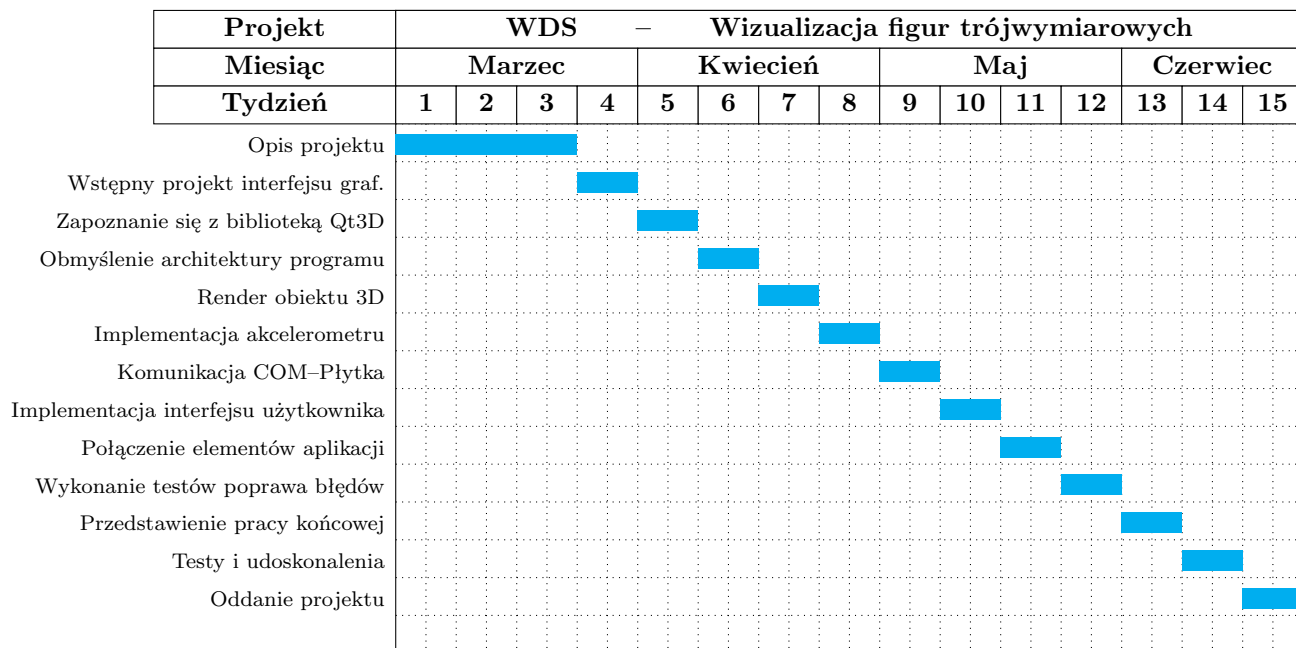
3 Terminarz realizacji poszczególnych podcelów

- 22 marca 2020 – Opis projektu
- 29 marca 2020 – Wstępny projekt interfejsu graficznego aplikacji
- 12 marca 2020 – Zapoznanie się z środowiskiem Qt Creator i biblioteką Qt3D do wizualizacji obiektu 3D
- 19 kwietnia 2020 – Obmyślenie funkcjonalności i architektury programu
- 26 kwietnia 2020 – Praca nad klasami odpowiedzialnymi za wyświetlanie i manipulowanie obiektem 3D
- 4 maja 2020 – Implementacja akcelerometru z żyroskopem na płytce rozwojowej
- 10 maja 2020 – Implementacja komunikacji między płytką rozwojową a aplikacją
- 17 maja 2020 – Praca nad interfejsem użytkownika, gdzie będzie pokazana orientacja obiektu w stopniach, suwaki do zmiany orientacji obiektu z poziomu interfejsu
- 24 maja 2020 – Połączenie wszystkich elementów aplikacji i przedstawienie ich w jednym oknie
- 31 maja 2020 – Wykonanie testów, wykrycie oraz poprawa znalezionych błędów
- 7 czerwca 2020 – Przedstawienie i opisanie pracy końcowej w sprawozdaniu
- 14 czerwca 2020 – Oddanie projektu

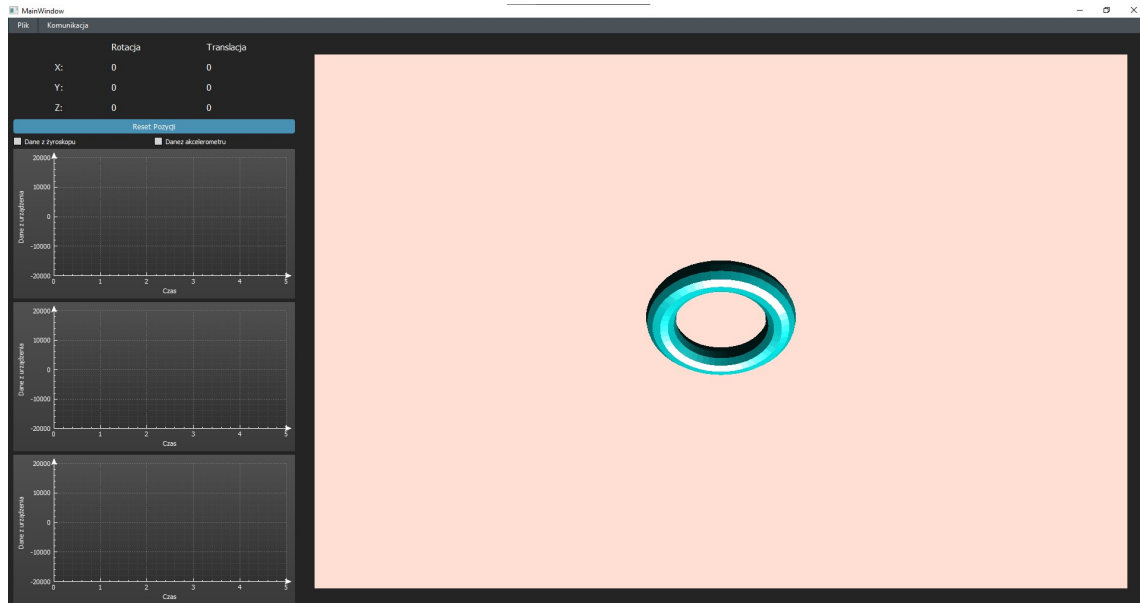
3.1 Kamienie milowe

- Wizualizacja obiektu 3D
- Komunikacja między płytką rozwojową a aplikacją
- Połączenie utworzonych elementów aplikacji
- Oddanie programu i sprawozdania

3.2 Wykres Gantta



4 Projekt graficznego interfejsu użytkownika



Rysunek 1: Interfejs użytkownika

4.1 Dostępne funkcjonalności

- **Pasek narzędzi** – W pasku narzędzi będzie można wyszukać urządzenia podłączone do portów COM, następnie połączyć się z płytką wybierając odpowiedni port. Możliwy jest również wybór pliku z modelem 3D, którym chcemy manipulować.
- **Pasek statusu** – Na pasku statusu, wyświetlany zostaje komunikat tekstowy z nazwą portu COM wybranego przez użytkownika, który będzie wykorzystywany do komunikacji z płytką rozwojową.
- **Tabela z danymi** – Tabela w której będzie można zobaczyć orientację oraz przesunięcie obiektu 3D względem jego położenia początkowego
- **Reset pozycji** – zrestartowanie położenia obiektu 3D do jego początkowego położenia.
- **Przyciski do wyboru wyświetlanych danych na wykresach** – Zaznaczając odpowiedni checkbox użytkownik może wybrać z którego czujnika będą wyświetlane dane na wykresach. Możliwe jest zaznaczenie obu checkbox'ów na raz co spowoduje jednoczesne wyświetlanie danych.
- **Widget z wizualizacją 3D** – Obserwacja wizualizacji obiektu 3D manipulowanego za pomocą akcelerometru z żyroskopem lub myszki i klawiatury.
- **Wykresy danych** – Wykresy danych odczytywanych z akcelerometru i żyroskopu

4.2 Przykładowy scenariusz

Uruchamiamy aplikację. Najpierw musimy skonfigurować połączenie z płytą STM, na pasku narzędzi klikając odpowiednią sekcję po przeszukaniu portów możemy podłączyć się do płytki. Aplikacja poinformuje nas o prawidłowym nawiązaniu połączenia z płytą rozwojową. Użytkownik będzie mógł wybrać w dowolnym momencie tryb sterowania wizualizacją obiektu, ma on do wyboru dwie opcje: sterowanie akcelerometrem z żyroskopem lub myszką podłączoną do komputera. Gdy znajdzie taka potrzeba użytkownik będzie mógł zresetować pozycję do pozycji startowej. Możliwy jest także wybór jakie odebrane dane mają być wyświetlane na wykresach.

5 Komunikacja z urządzeniem

Płytką rozwojową STM32F411EDISCOVERY ma wbudowany akcelerometr **LSM303DLHC** i żyroskop **L3GD20** co sprawia, że użytkownik jest zwolniony z planowania okablowania układu. Płyka jednak nie jest wyposażona w konwerter USB-UART, ponieważ użycie tego typu komunikacji zakłada projekt, konieczne okazało się dołączenie do płytki takiego konwertera. W projekcie został wykorzystany **konwerter USB-UART PL2303**

5.1 Protokół komunikacji

Ramka protokołu komunikacyjnego zaczyna się od dużej litery alfabetu, dzięki której program będzie mógł odróżnić skąd odbierane są dane:

- **A** – Akcelerometr
- **G** – Żyroskop

Z powodu indendycznego formatu odbieranych danych z obu urządzeń ramki dla nich różnią się tylko wymienionym wyżej identyfikatorem.

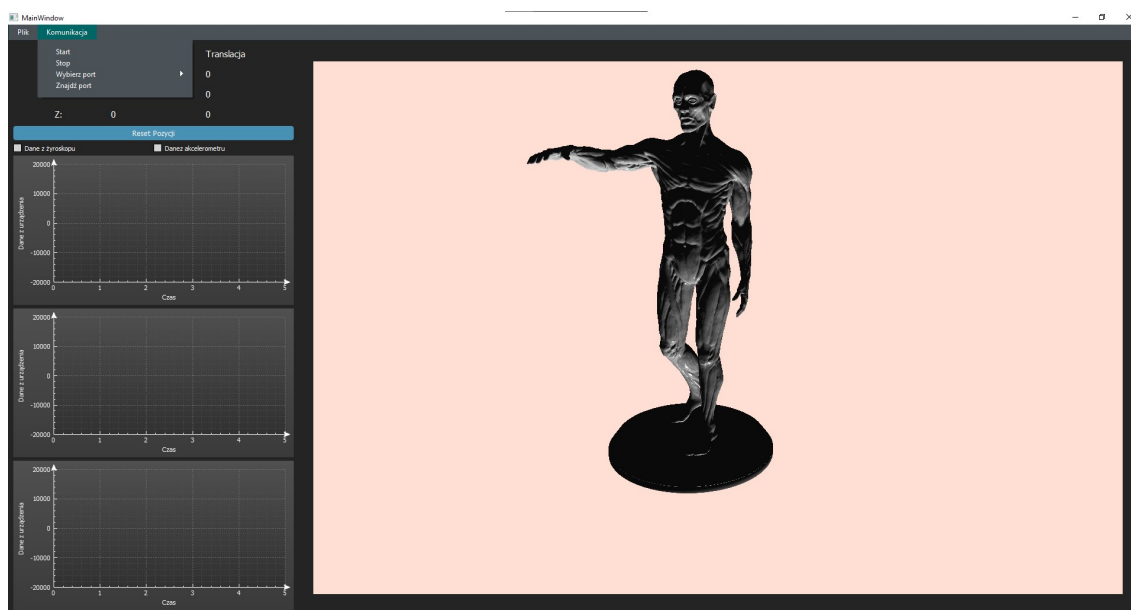
Forma ramki to: $* \langle int8_t \rangle \langle int8_t \rangle \langle int8_t \rangle CRC$

Gdzie $*$ jest identyfikatorem nadającego dane urządzenia. Po nim znajdują się trzy wartości odpowiadające trzem osiom XYZ. Dane te mieszczą się w zakresie wartości od -16132 do 16132, są to wartości całkowite. Na końcu znajduje się suma kontrolna. Po stronie mikrokontrolera ramka jest formatowana do stringa i wysyłana przez komunikację UART za pomocą funkcji dostarczonej przez bibliotkę HAL dla mikrokontrolerów STM. W aplikacji Qt, za pomocą operacji na odebranych stringu, jest on dzielony na poszczególne elementy, które ma w sobie ramka. Jest to możliwe dzięki oddzieleniu poszczególnych elementów ramki spacjami. Przed przypisaniem odebranych danych do odpowiednich zmiennych w programie, następuje sprawdzenie ich poprawności za pomocą sumy kontrolnej CRC.

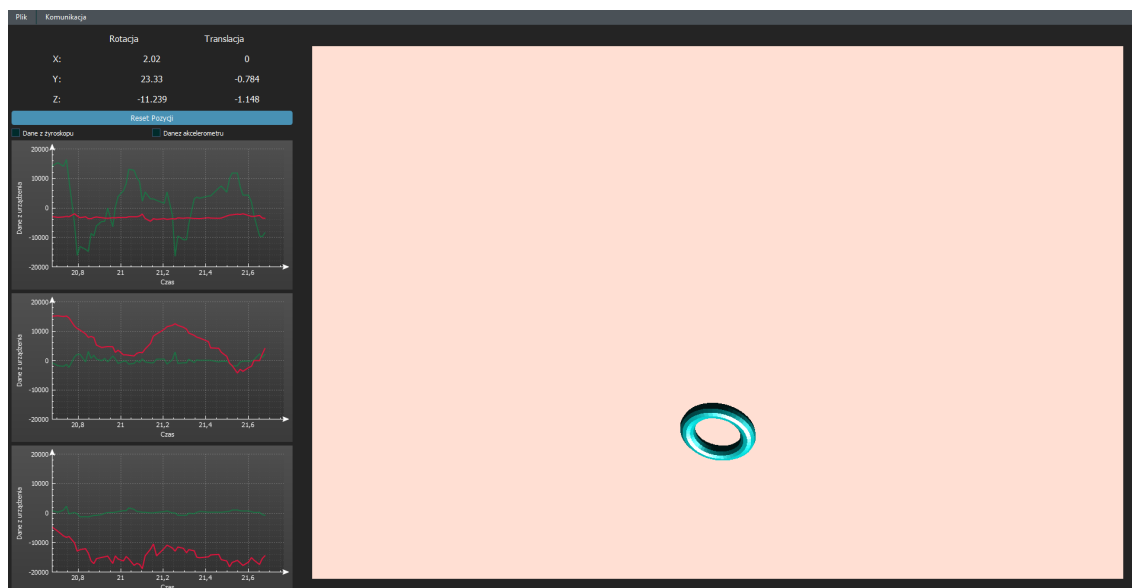
6 Działanie programu

Dane odbierane są przez klasę Device, która ma pole klasy QSerialPort. Dzięki omawianemu polu i wbudowanym w nim mechanizmom możemy wydajnie prowadzić wymianę danych pomiędzy płytą deweloperską, a programem. Dzięki intuicyjnemu menu na pasku urządzeń [2] użytkownik może wybrać za pomocą którego portu ma dojść do komunikacji z płytą deweloperską. Po wybraniu portu użytkownik jest o tym informowany

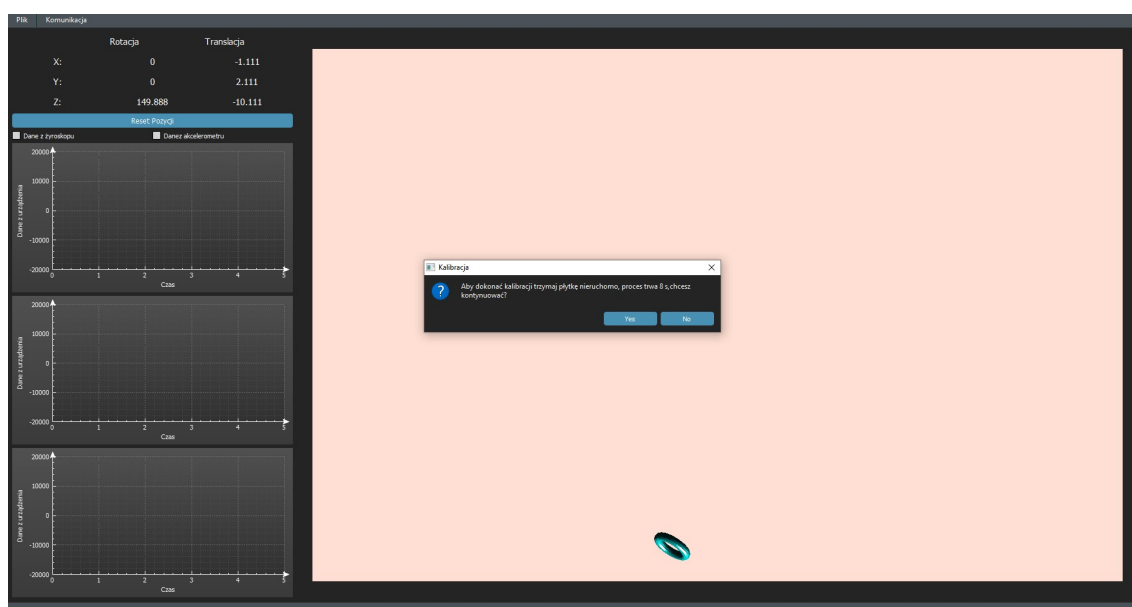
dzięki wyświetlanemu komunikatowi na pasku statusu[6]. Bezpieczeństwo przesyłu danych gwarantuje zastosowanie mechanizmu sumy kontrolnej, która jest wyliczana na płycie deweloperskiej przed wysyłką pakietu, a następnie dołączana jest do pakietu danych. Przy zastosowaniu tego samego algorytmu w kodzie aplikacji sprawdzamy czy wysłana wartość CRC zgadza się z tą wysłaną wcześniej z płytki. Klasa Device jest również odpowiedzialna za przetworzenie dostarczonych danych na użyteczne wartości, które możemy wykorzystać do manipulacji wczytanego przez użytkownika dowolnego modelu 3D[2] oraz kalibrację żyroskopu i akcelerometru. Użytkownik ma możliwość podjęcia decyzji czy chce dokonać kalibracji czujników. Podjęcie decyzji jest wykonywane przy pomocy okna dialogowego[4], gdzie użytkownik może kliknąć tak lub nie. Przy zdecydowaniu się na wykonanie kalibracji wyświetla się komunikat informujący o tym, że trwa proces kalibracji[5]. Głównym elementem aplikacji jest klasa ObjectScene, to w niej renderowany jest obiekt, który można zobaczyć w przeznaczonym do tego widżecie. Sloty tej klasy są odpowiedzialne za odbieranie emitowanych sygnałów z przetworzonymi wcześniej wartościami rotacji i translacji wizualizowanego aktualnie modelu 3D. Surowe dane otrzymywane z żyroskopu i akcelerometru możemy obserwować na wykresach[3] wygenerowanych za pomocą biblioteki QCustomPlot, które w czasie rzeczywistym prezentują nam w jaki sposób ruch płytki na której umieszczone są sensory wpływa na odbierane przez aplikację dane. Minimalistyczny i przejrzysty wygląd aplikacji zapewnia napisany skrypt w pliku ".qss".



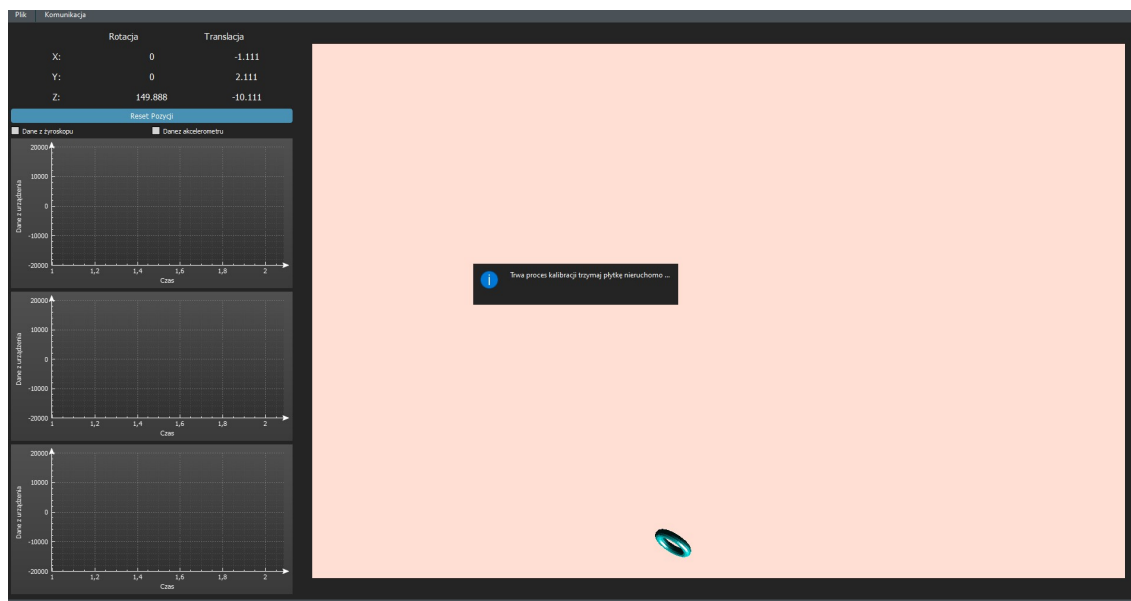
Rysunek 2: Wczytanie skomplikowanego modelu 3D i prezentacja menu komunikacji



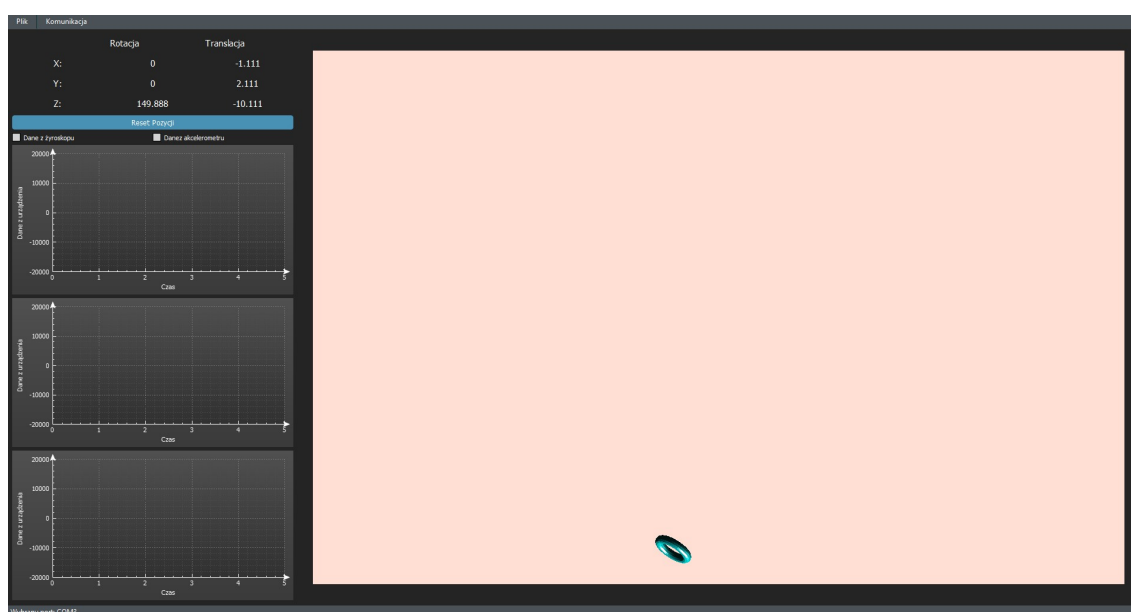
Rysunek 3: Prezentacja funkcjonujących wykresów



Rysunek 4: Prezentacja komunikatu z pytaniem do użytkownika



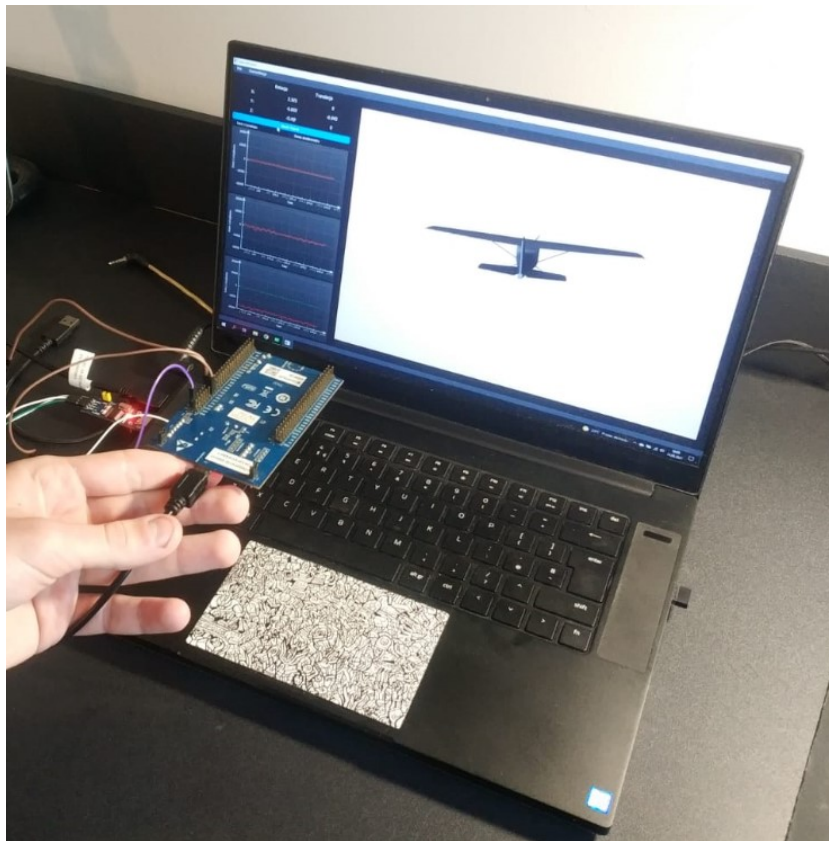
Rysunek 5: Prezentacja komunikatu o trwającym procesie kalibracji



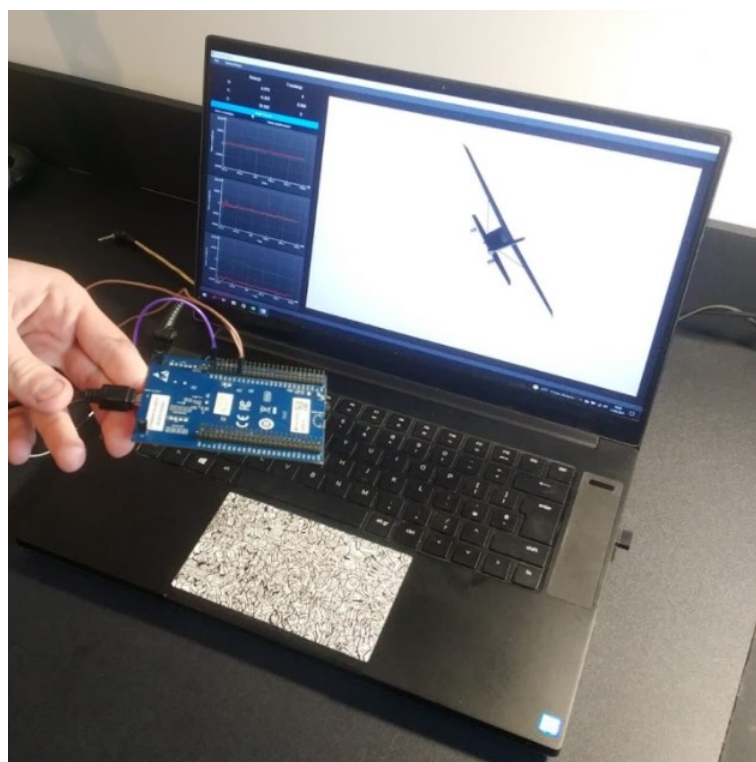
Rysunek 6: Prezentacja komunikatu o wybranym porcie COM na pasku statusu

7 Eksperymenty

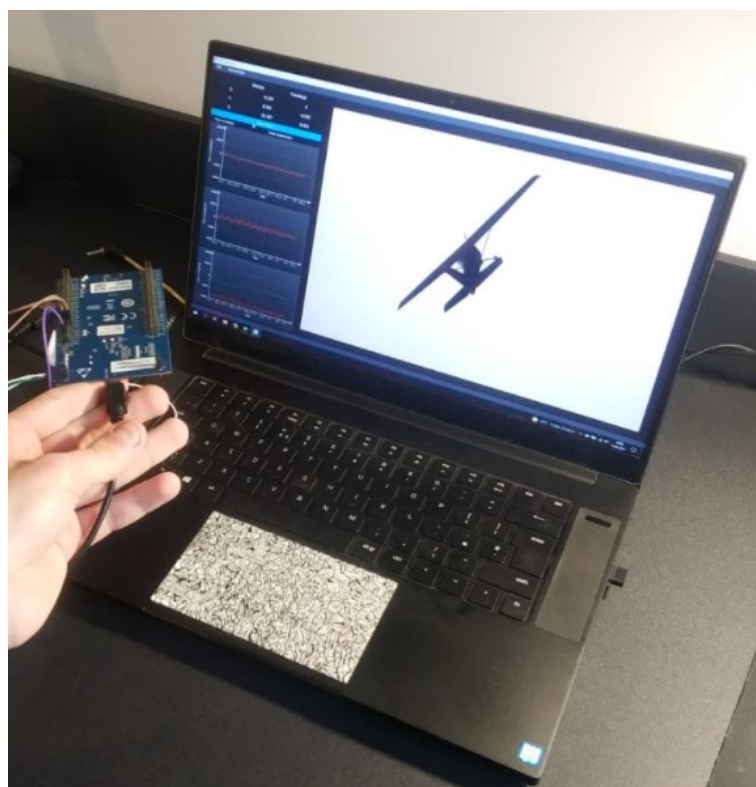
Przeprowadzone zostały obserwacje na różnych modelach 3D, pierwszym spostrzeżeniem podczas wykonywania testów było to, że modele różnią się ustawionymi osiami w swojej geometrii, przez co nie zawsze oś 0X na czujniku będzie oznaczała oś 0X modelu 3D. Na prezentowanych zdjęciach testy zostały przeprowadzone na modelu samolotu[7]. Zauważyć można, że mimo obracania płytki wokół osi 0Z[8][9], model wykonuje rotację wokół osi 0Y. Jednak nie wpływa to w jakikolwiek sposób w wygodę korzystania z programu, jest to spowodowane tym, że po wykryciu przez użytkownika jak ułożone są osie modelu 3D, można zresetować pozycję i ustawić kamerę jak tylko użytkownik tego zapragnie. Czułość wykrycia przesunięcia płytki została zmniejszona przez co lekkie ruchy płytką nie powodują widocznych efektów, zastosowane zostało takie rozwiązanie po to aby użytkownik podczas obracania płytką nie otrzymywał nie porządkanych przesunięć obiektu. Wykonanie testów na różnych modelach 3D daje satysfakcjonujące wyniki dla rotacji i translacji.



Rysunek 7: Początkowa pozycja płytki



Rysunek 8: Obrót w prawo wokół osi 0Z czujników



Rysunek 9: Obrót w lewo wokół osi 0Z czujników

8 Kod i dokumentacja

Kod aplikacji można znaleźć w repozytorium pod linkiem:

<https://github.com/AdamKubiak/WDS_{Qt3D}>

Dokumentacja kodu programu została wygenerowana za pomocą Doxygen i jest ona dołączona do repozytorium z linku powyżej.

Literatura

- [1] B. Kreczmer. *Materiały z z wykładów z kursu Wizualizacja Danych Sensorycznych*.
<https://kcir.pwr.edu.pl/kreczmer/wds/>.
- [2] M. Patyk. *Kurs Qt komunikacja z Arduino przez UART*
<https://forbot.pl/blog/kurs-qt-2-komunikacja-z-arduino-przez-uart-id356017>.
- [3] Marek Galewski. *Aplikacje i ćwiczenia w języku C z biblioteką HAL*.
Wydawnictwo BTC, Legionowo, Lipiec 2019.
- [4] ST. *LM303DLHC datasheet*.
<https://www.st.com/resource/en/datasheet/lsm303dlhc.pdf>
- [5] ST. *L3GD20 datasheet*.
<https://www.st.com/en/mems-and-sensors/l3gd20.html#documentation>