Politechnika Wrocławska

Roboty mobilne
Raport końcowy

Sonarowy skaner otoczenia

Autorzy:
Dorian Janiak
Marcin Ochman

Prowadzący:
mgr inż. Mateusz Cholewiński

15 czerwca 2015

Spis treści

Spis rysunków

1 Cel projektu

Celem projektu było stworzenie robota mobilnego potrafiącego skanować otoczenie. Projekt miał nam pomóc zapoznać się z podstawowymi problemami związanymi z budową własnych robotów.

2 Krótki opis projektu

Projekt składał się z dwóch części. Pierwszą częsc miał stanowić robot mobilny, którego system jedzny miał być zbudowany z dwóch kół zamieszczonych w połowie długości stelażu. Zadaniem robota było skanowanie otoczenia przy użyciu sonarowego czujnika odległości.
Drugą część projektu miała stanowić aplikacja komputerowa, która wizualizuje otrzymywane pomiary od robota oraz powala na sterowanie robotem.

3 Doręczenie

Terminy i rodzaje dostarczanych w ramach projektu dokumentów oraz oprogramowania zostały przedstawione w Tabeli 1. Znalazły się również informacje o jawności.

Nr	Nazwa	Termin	Postać	Jawność
1	Zbudowanie podwozia robota	2	sprzęt, dokumentacja	Grupa
2	Finalizacja pracy sprzętowej nad robotem	3	sprzęt, dokumentacja	Grupa
3	Oprogramowanie robota	4	oprogramowanie, dokumentacja	Grupa
4	Algorytm budowania mapy	13	oprogramowanie, dokumentacja	Grupa
5	Ukończenie prac nad aplikacją	7	oprogramowanie, dokumentacja	Grupa
6	Połączenie działania robota i aplikacji	7,8	oprogramowanie, sprzęt, dokumentacja	Publiczne
7	Raport końcowy	9	dokumentacja	Publiczna

Tablica 1: Tabela doręczeń

4 Budżet

Projekt ma charakter sprzętowy, a więc wymaga on kupna elementów. Zestawienie kosztów zostało umieszczone w tabeli 3. Wszelkie koszta zostaną pokryte z własnych funduszy.

Element	**Cena całkowita(zł)**
Zestaw STM32 NUCLEO	55
Moduł Bluetooth HC-05	37
Zestaw silników i kół z enkoderami Dagu RS034	110
Silnik krokowy ze sterownikiem ULN2003	20
Kulka podporowa	10
Płytka PCB uniwersalna	8
Mostek H L298N	11
Pozostałe elementy zabezpieczające i zasilające	30
Nieprzewidziane koszty	30
Ogólny koszt:	311

Tablica 2: Budżet zadania

5 Zarządzanie projektem i jego monitorowanie

Ze względu na to, że projekt jest realizowany tylko przez 2 osoby, lider nie został wyznaczony. Tak mały zespół nie powinien mieć problemów komunikacyjnych. Realizować projekty będziemy częściowo zdalnie, jednak

planujemy również regularnie się spotykać i wspólnie pracować nad projektem. Ustalenia będą powstawały w trakcie spotkań oraz w trakcie rozmów realizowanych na portalu facebook.com . Mimo, że jest to serwis społeczny, już od dłuższego czasu okazuje się, że nadaje się do komunikacji grup projektowych. Nasz projekt nie wymaga tworzenia grup użytkowników o różnych uprawnieniach, przydzielania zadań i tworzenia ich zestawień, a więc ten portal oraz umówione spotkania powinny wystarczyć.
Współdzielić kod programów będziemy poprzez portal **github.com** . Nie zakładamy tajności projektu, a więc może być on udostępniony publicznie.
Postępy monitorować będziemy na bieżąco poprzez porównanie aktualnych osiągnięć z harmonogramem. Kamienie milowe pozwolą nam ocenę ewentualnych opóźnień. Opóźnienia wymuszą na nas przyspieszenie prac lub ograniczenie założeń projektu.

6 Przydział zadań

Ponieważ grupa składa się tylko z dwóch osób projekt w dużej mierze będzie realizowany wspólnie. Można jednak wyróżnić część zadań, które będą realizowane osobno. Podział prezentuje poniższa tabela (Tabela 4). Nakreśla ona wybór lidera danego zadania. Będzie on przede wszystkim odpowiedzialny za kontrolę czasu wykonania zadania.

Nr	Nazwa zadania	Lider
Z1	Zebranie wszystkich potrzebnych elementów do budowy robota	M. Ochman
Z2	Budowa podwozia robota	M. Ochman
Z3	Instalacja układów elektronicznych, elektrycznych oraz czujników	M.Ochman
Z4	Oprogramowanie robota - komunikacja oraz sterowanie	M.Ochman
Z5	Stworzenie programu komputerowego wizualizujący dane symulacyjne ładowane z pliku	D. Janiak
Z6	Dodanie modułu, obsługującego komunikację poprzez Bluetooth	D.Janiak
Z7	Dodanie możliwości sterowania robotem z poziomu programu komputerowego	D.Janiak
Z8	Stosowanie poprawek	D.Janiak

Tablica 3: Podział zadań w grupie

W skład grupy wchodzą osoby:

Marcin Ochman - student AiR. Wybrał specjalność Robotyka, ponieważ interesuje się zarówno komputerami, elektroniką oraz nowinkami technicznymi. Jego głównym atutem jest umiejętność programowania w różnych językach takich jak C, C++, C#, Python, SQL. Swoje doświadczenie zdobywa zarówno na uczelni oraz pracując w firmie Vulcan.

Dorian Janiak - Pisze od kilku lat programy w językach C++/C. Podejmował się również pracy z takimi językami jak Python, Matlab czy QML. Obecnie pracuje jako programista C++. Jego głównym zainteresowaniem jest grafika 3D (używał OpenGL i GLSL, zna podstawy RayTracingu, posługuje się programem Blender 3D). Ukończył kurs języka niemieckiego na poziomie B2.

7 Funkcjonalność

Poniżej zostały opisane części funkcjonalności oddzielnie dla robota oraz aplikacji komputerowej.

7.1 Funkcjonalność robota

Robot jest w stanie wykonywać poniższe czynności:

- Obrót wału silnika krokowego celem kalibracji pozycji startowej skanowania.
- Wykonanie skanu otoczenia w zakresie 180 stopni obrotu.
- Komunikacja poprzez Bluetooth z komputerem PC.
- Symulowana odpowiedź na żądanie zmiany położenia robota.

Nie udało się natomiast stworzyć systemu jezdnego robota. Przez co poniżej przedstawiamy listę nieskończonych funkcji:

- Przemieszczenie się robota w zadaną pozycję.
- Oszacowanie rzeczywiście przebytej trasy.

7.2 Funkcjonalność aplikacji komputerowej

W przypadku aplikacji komputerowej stworzone zostały wszystkie funkcje wyszczególnie na etapie założeń. Ostatecznie udało się również utworzyć dodatkowe funkcje takie jak: wyświetlanie logów dot. stanu aplikacji, ładowanie mapy otoczenia z plików symulacyjnych. Poniżej wypunktowane zostały wszystkie stworzone funkcje programu:

- Możliwość sterowanie widokiem 3D.
- Rysowanie mapy otoczenia oraz robota w widoku 3D.
- Wyświetlanie logów dotyczących stanu aplikacji.
- Możliwość konfiguracji połączenia szeregowego.
- Komunikacja poprzez port szeregowy.
- Ładowanie mapy z pliku symulacyjnego.
- Dostęp do funkcji kalibracji pozycji startowej skanu otoczenia.
- Możliwość sterowania pozycją robota.
- Żądanie wykonania oraz obsługa skanowania otoczenia.

8 Przebieg sterowania

Sterowanie robotem odbywa się poprzez interfejs Bluetooth. W robocie zamontowany został moduł Bluetooth HC-05. Po pierwszym połączeniu się z komputerem interfejs domyślnie jest widziany jako port szeregowy COM. Tak więc komunikacja może odbywać się również z poziomu terminala.

8.1 Protokół komunikacyjny

Na potrzeby projektu stworzyliśmy własny protokół komunikacji, składający się z kilku krótkich komend przesyłanych poprzez Bluetooth.

- Potwierdzenie powodzenia operacji.

  ```
  W0;0
  ```

- Informacja o rozpoczęciu działania robota.

  ```
  W1;0
  ```

- Żądanie wykonania skanowania wysyłane do robota.

  ```
  W2;2;ile_pomiarów;vKąt
  ```

  - ile_pomiarów - całkowita liczba dodatnia, ilość pomiarów jakie mają zostać wykonane w zakresie 180 stopni obrotu.
  - vKąt - całkowita liczba dodatnia, szybkość obrotu czujnika wyrażona w stopniach na sekundę.

- Rezultat wykonania skanowania, który robot wysyła w odpowiedzi na wiadomość W2.

  ```
  W3;ile_pomiarów*2;[kąt_akustyczny;pomiar_w_cm]
  ```

  - ile_pomiarów - całkowita liczba dodatnia, ilość pomiarów jakie zostały wykonane w ramach jednego pomiaru.
  - kąt_akustyczny - całkowita liczba dodatnia, kąt wyrażony w stopniach

  - pomiar_w_cm - całkowita liczba dodatnia, wynik pomiaru wyrażony w cm.

- Żądanie obrotu wału silnika krokowego bez wykonania pomiarów. Odpowiedzią na pozytywne wykonanie operacji jest wiadomość W0.

  `W4;3;kierunek;kąt;vKąt`

  - kierunek - 1 gdy zgodnie zasadą prawej dłoni, w przeciwnym wypadku 0
  - kąt - całkowita liczba, kąt obrotu wyrażony w stopniach
  - vKąt - całkowita liczba dodatnia, szybkość obrotu czujnika wyrażona w stopniach na sekundę.

- Żądanie jazdy robota, a konkretniej przemieszczenia do konkretnego punktu. Najpierw robot powinien obrócić się o kąt zadany w parametrze obrót_przed_ruchem, następnie przejechać po linii prostej odległość podaną w parametrze dystans_cm, a na koniec jeszcze obrócić się o kąt podany w parametrze obrót_po_ruchu.

  `W5;4;vSzybkość;dystans_cm;obrót_przed_ruchem;obrót_po_ruchu`

  - vSzybkość - całkowita liczba dodatnia, szybkość wyrażona w cm na sekundę
  - dystans_cm - całkowita liczba dodatnia, dystans jaki ma robot przejechać po linii prostej wyrażony w cm
  - obrót_przed_ruchem - całkowita liczba, kąt wyrażony w stopniach
  - obrót_po_ruchu - całkowita liczba, kąt wyrażony w stopniach

- Rezultat wykonania przemieszczenia robota, który jest odpowiedzią na wiadomość W5.

  `W6;3;dystans_cm;obrót_przed_ruchem;obrót_po_ruchu`

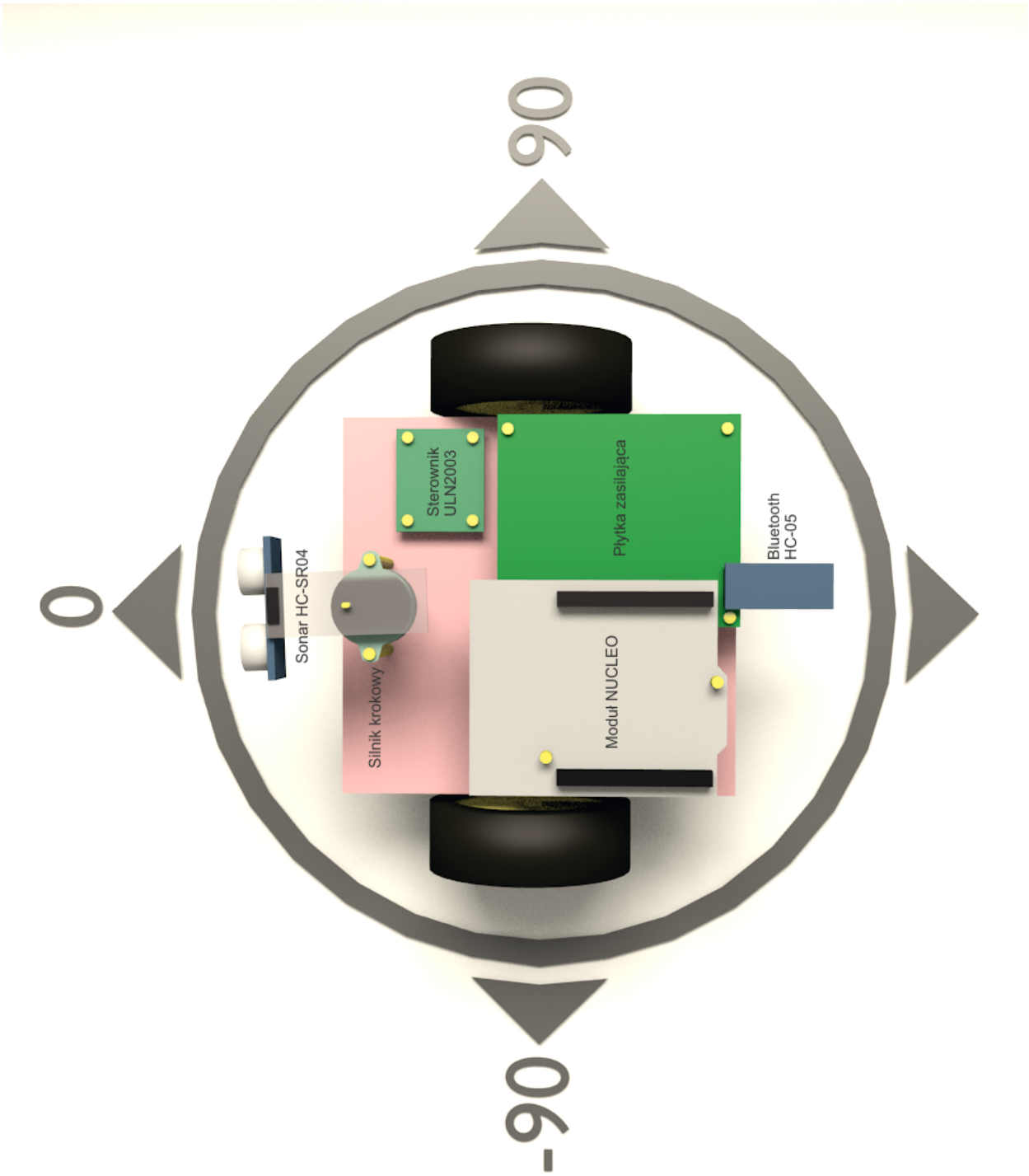
  - dystans_cm - całkowita liczba dodatnia, dystans jaki ma robot przejechać po linii prostej wyrażony w cm
  - obrót_przed_ruchem - całkowita liczba, kąt wyrażony w stopniach
  - obrót_po_ruchu - całkowita liczba, kąt wyrażony w stopniach

- Niepowodzenie operacji.

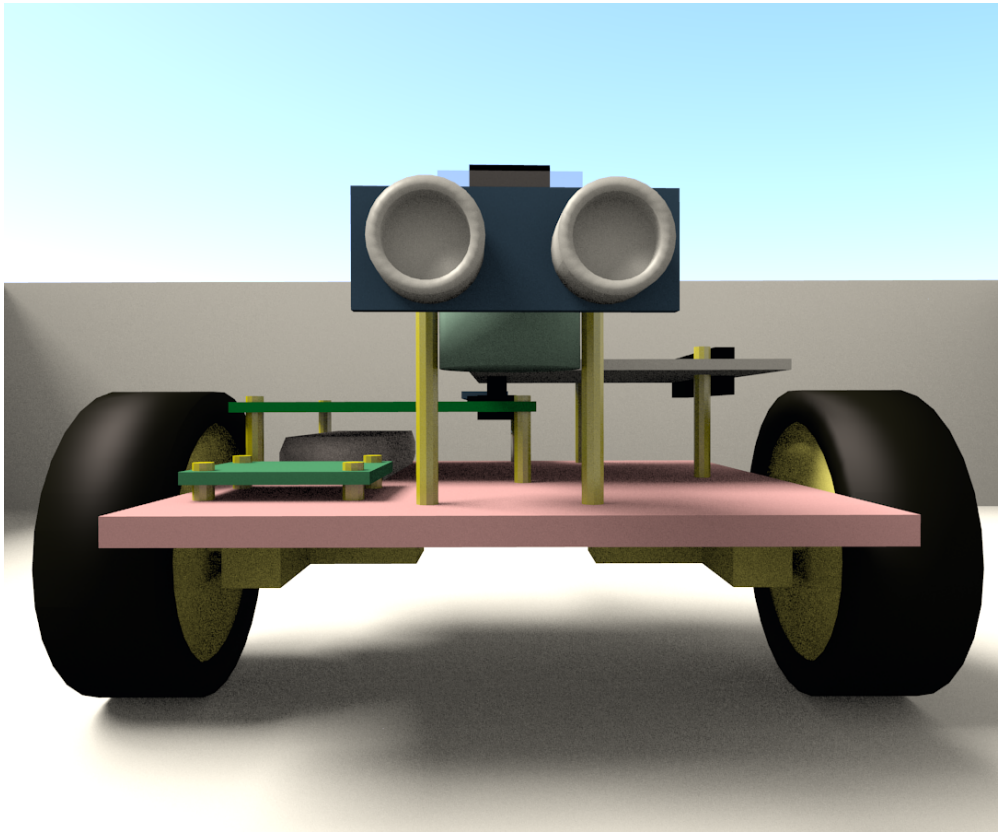
  `W99;0`

8.2 Kolejne etapy sterowania

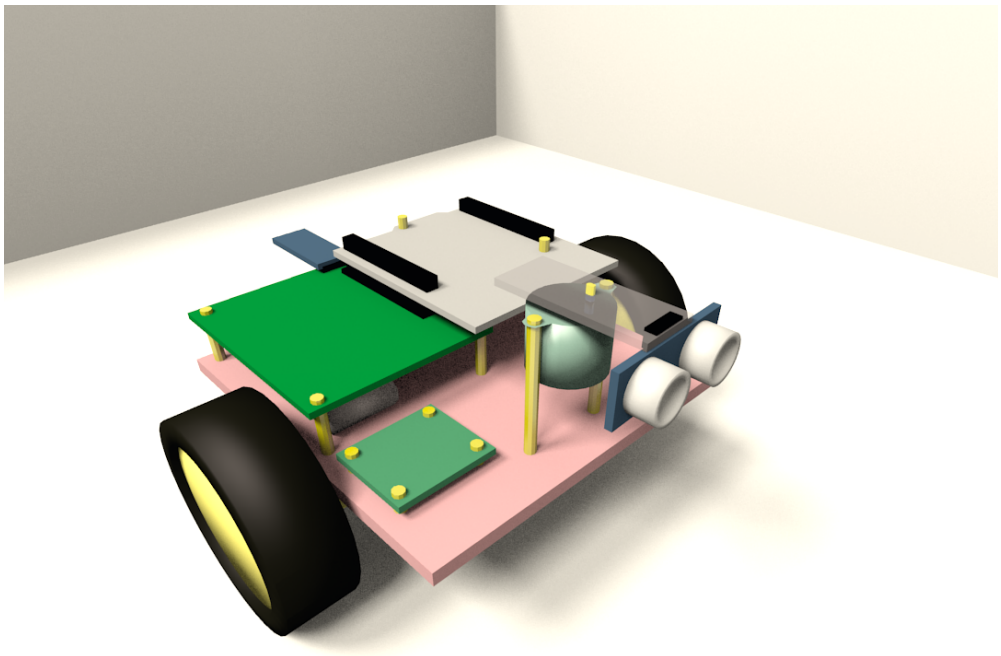
Aby połączyć się z robotem należy najpierw skonfigurować połączenie. Domyślnie jest to 9600 bodów, 8 bitów danych, 1 bit stopu, brak parzystości oraz brak kontroli przepływu. Po wybraniu takiej konfiguracji można połączyć się z robotem. Aby zobaczyć odpowiedź robota bez konieczności wysyłania komend sterujących można nacisnąć czarny przycisk na robocie (Reset). Wtedy robot wyśle wiadomość typu W1.
Po połączeniu się z robotem należy upewnić się czy oś akustyczna znajduje się na pozycji -90 stopni. Jeśli nie to należy obrócić wał silnika krokowego w taki sposób, aby oś akustyczna znajdowała się w odpowiedniej pozycji. Kalibracja odbywa się przy użyciu wiadomości W4.
Kiedy poprzednie etapy już zostały wykonane można wykonać skanowanie w zakresie 180 stopni. Skanowanie odbędzie się po przesłaniu komendy W2. Po skończonym skanowaniu robot wyśle odpowiedź W3 z zestawem pomiarów dla kolejnych konfiguracji osi akustycznej. Odpowiedź zostanie przesłana dopiero po skończeniu, a nie w trakcie.
Aby zbudować mapę otoczenia można robota przemieścić w inne położenie i wykonać skan otoczenia z innego punktu. Z poziomu aplikacji należy najpierw przy użyciu pilota przesunąć w widoku 3D żółty obiekt reprezentujący robota, a następnie nacisnąć przycisk *Jedź robocie, jedź*. W przypadku terminala należy wysłać wiadomość W5. Robot w tej sytuacji zwróci wiadomość W6.
Ponieważ nasz robot ostatecznie nie jest w stanie jeździć, należy go przemieścić ręcznie do zadanej wcześniej pozycji.



Rysunek 1: Model 3D robota z zaznaczonym kątem akustycznym



Rysunek 2: Model 3D robota



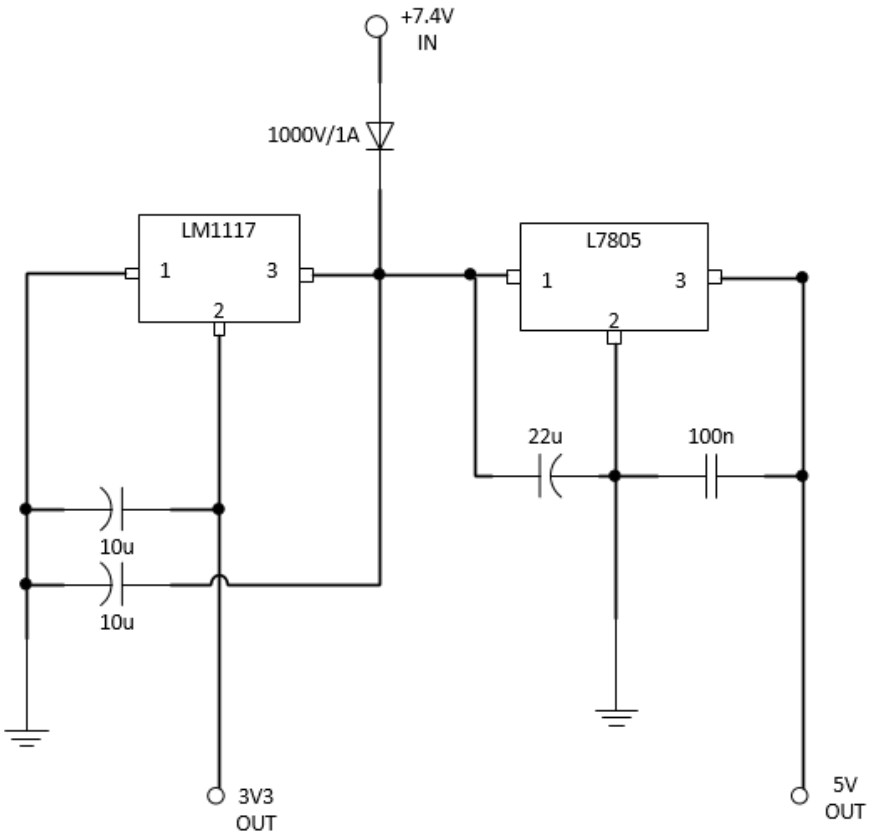
Rysunek 3: Model 3D robota

9 Budowa robota

Robot nie został do końca zbudowany. Ostatecznie nie posiada planowanego wcześniej systemu jezdnego. Na rysunku 1 został zamieszczony obrazek prezentujący model 3D planowej konstrukcji. Jak widać na renderach zamierzaliśmy zamieścić pod płytką zasilającą akumulator li-pol.

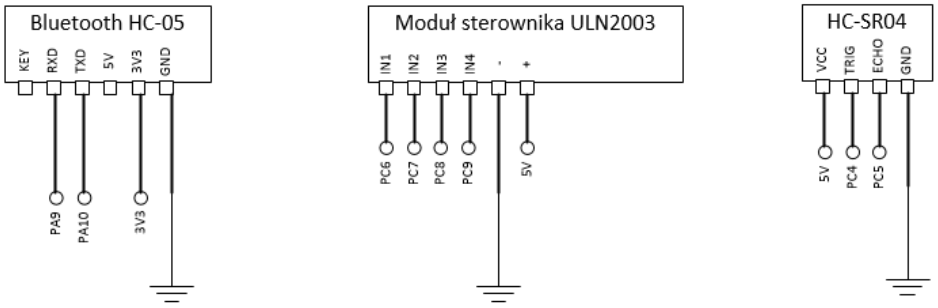
9.1 Połączenia elektryczne

Do zasilenia płytki zastosowaliśmy akumulator li-pol o napięciu 7.4V. Napięcie akumulatora zostało ustabilizowane na dwóch poziomach - 3.3V oraz 5V, ponieważ takie jest wymagane do zasilania poszczególnych modułów. Aby otrzymać takie poziomy zastosowaliśmy stabilizatory napięcia LDO LM1117t (dla 3V3) oraz L7805BV (dla 5V). W celu zabezpieczenia układu zastosowana została dioda prostownicza (zabezpieczenie przed odwrotną polaryzacją) oraz kondensatory (redukcja zakłóceń).
Moduł NUCLEO został zasilony napięciem 5V (pin E5V).
Rysunek 4 przedstawia połączenia sekcji zasilania.



Rysunek 4: Schemat podłączenia zasilania

Rysunek 5 przedstawia połączenia pinów modułu NUCLEO z pozostałymi modułami.



Rysunek 5: Schemat połączeń modułów

9.2 Kosztorys

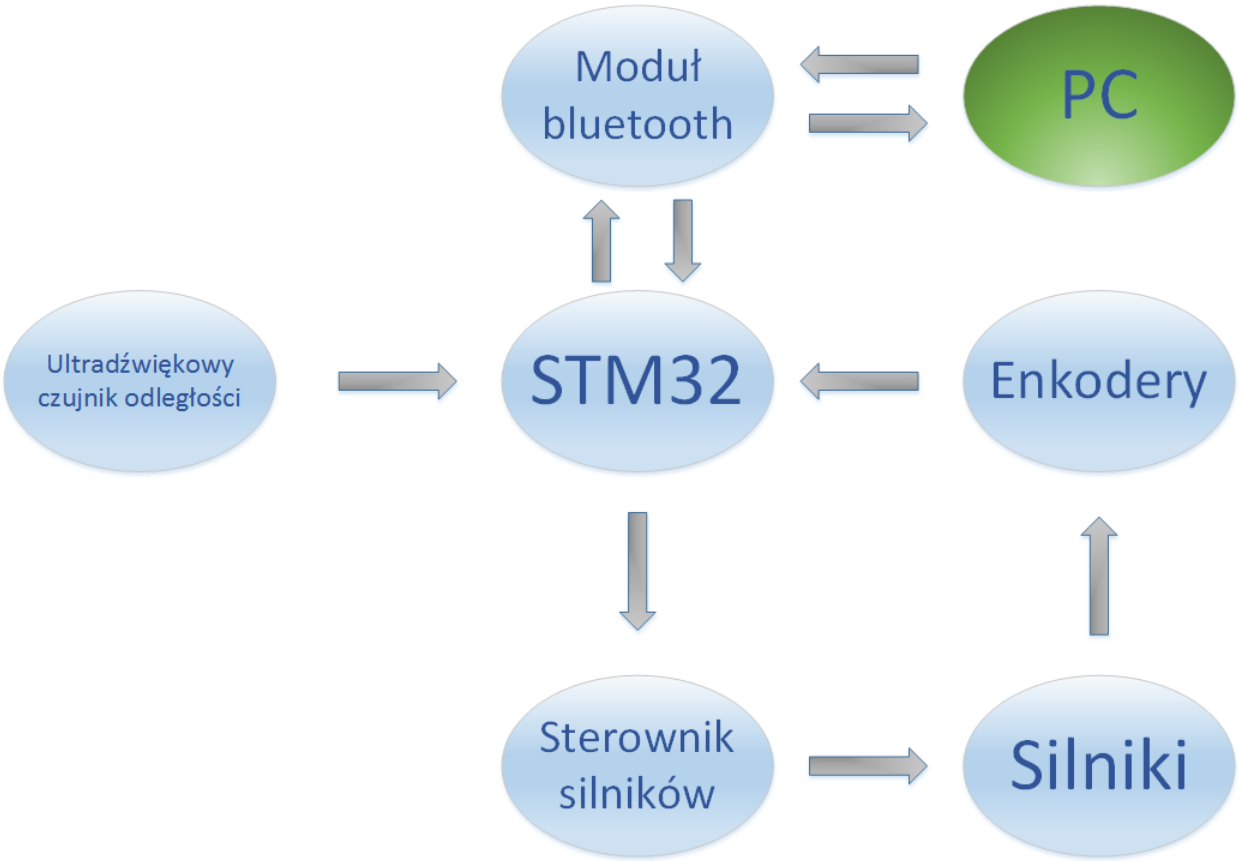
Na etapie założeń oszacowaliśmy, że będziemy potrzebowali 311 zł do realizacji projektu. Jak widać na poniższym zestawieniu udało się to zrealizować niższym kosztem. Koszty wzrosłyby gdybyśmy rozpoczęli montaż stelażu. Wtedy należałoby dokupić jeszcze pleksę, słupki montażowe, uchwyty do silników oraz zestawy śrubek.

Element	**Cena całkowita(zł)**	**Zakup poza założeniami**
Zestaw STM32 NUCLEO	49.56	
Moduł Bluetooth HC-05	31.37	
Zestaw silników i kół z enkoderami Dagu RS034	92.65	
Silnik krokowy ze sterownikiem ULN2003	14.79	
Kulka podporowa	9.15	
Płytka PCB uniwersalna	3.83	
Mostek H L298N	9.28	
Pozostałe elementy zabezpieczające i zasilające	4.61	
Mostek H TB6612	11.99	TAK
Przewody żyłowe	9.99	TAK
Ogólny koszt:	237.22	

Tablica 4: Budżet zadania

9.3 Oprogramowanie robota

Na rysunku 6 została przedstawiona komunikacja pomiędzy poszczególnymi modułami robota(zaznaczone na niebiesko). Diagram ten definiuje w jaki sposób wygląda sterowanie robotem oraz komunikacja pomiędzy komputerem osobistym. Widać, że za całe sterowanie jest odpowiedzialny *STM32*. To on powinien decydować jak mają poruszać się silniki, jak powinny być przetworzone informacje z enkoderów oraz czujnika ultradźwiękowego. Ostatecznie to on jest odpowiedzialny za wysyłanie oraz odbieranie informacji i przetwarzanie komend przychodzących z komputera.



Rysunek 6: Diagram przedstawiający moduły robota i PC oraz komunikację pomiędzy nimi

Oprogramowanie robota zostało napisane przy użyciu środowiska Keil μVision 5. W projekcie wyorzystaliśmy bibliotekę standard peripheral library

9.4 Ostateczny efekt robota

Ostatecznie połączone zostały ze sobą sekcja zasilania, moduł NUCLEO, silnik krokowy sterowany przy użyciu sterownika ULN2003, moduł Bluetooth HC-05 oraz czujnik ultradźwiękowy HC-SR04. W tej sytuacji robot nie może się sam przemieszczać, ale za to jest w stanie wykonywać skanowanie otoczenia co było tematem naszego projektu.
Rysunek 7 pokazuje rzeczywiście połączone moduły.

10 Wygląd aplikacji komputerowej

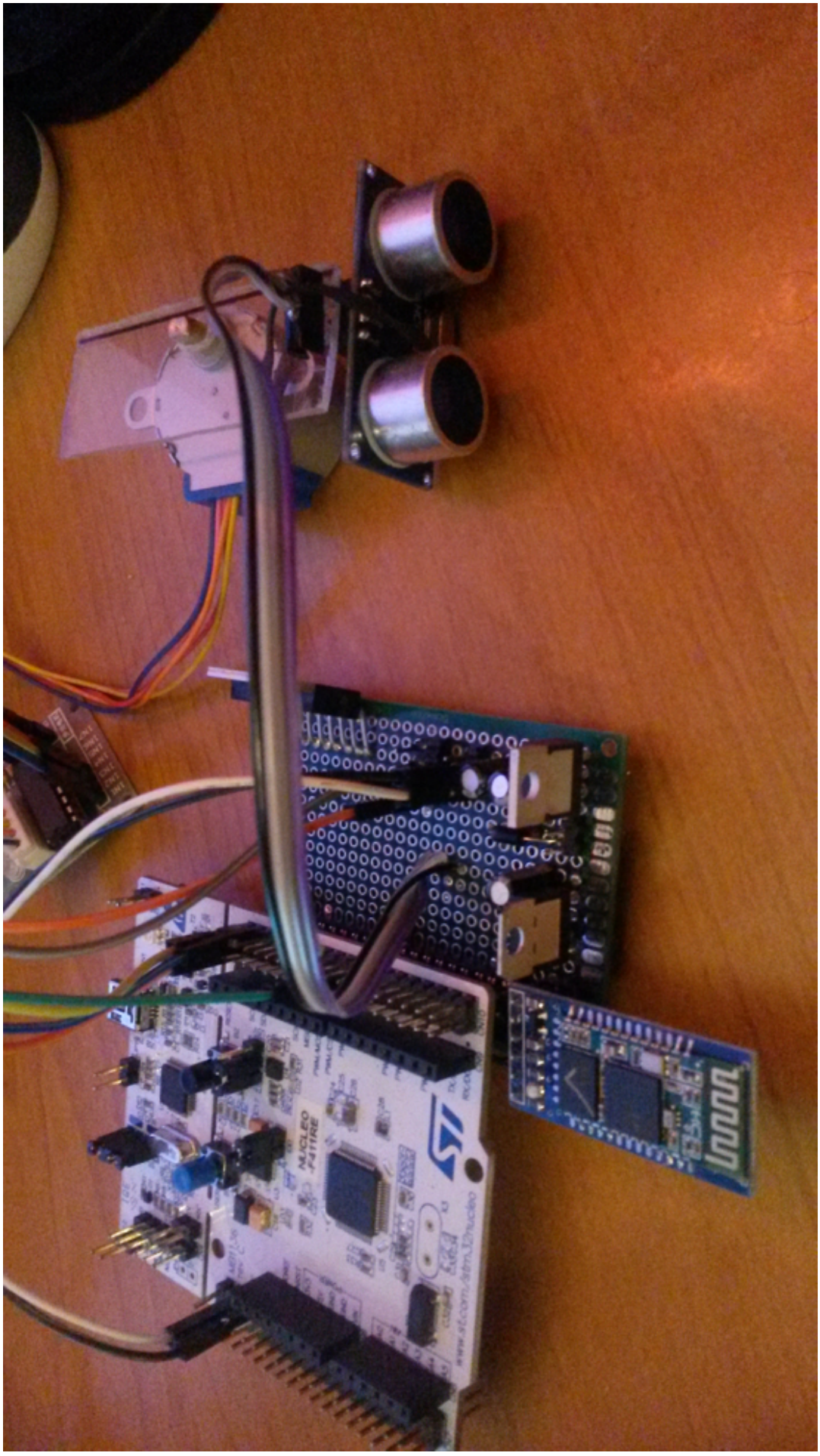
Główna aplikacja komputerowa składa się z widżetów oraz dodatkowych okien wyświetlanych w przypadku reakcji na błędy lub wywołanie akcji z poziomu paska menu.

10.1 Widżety dokowane

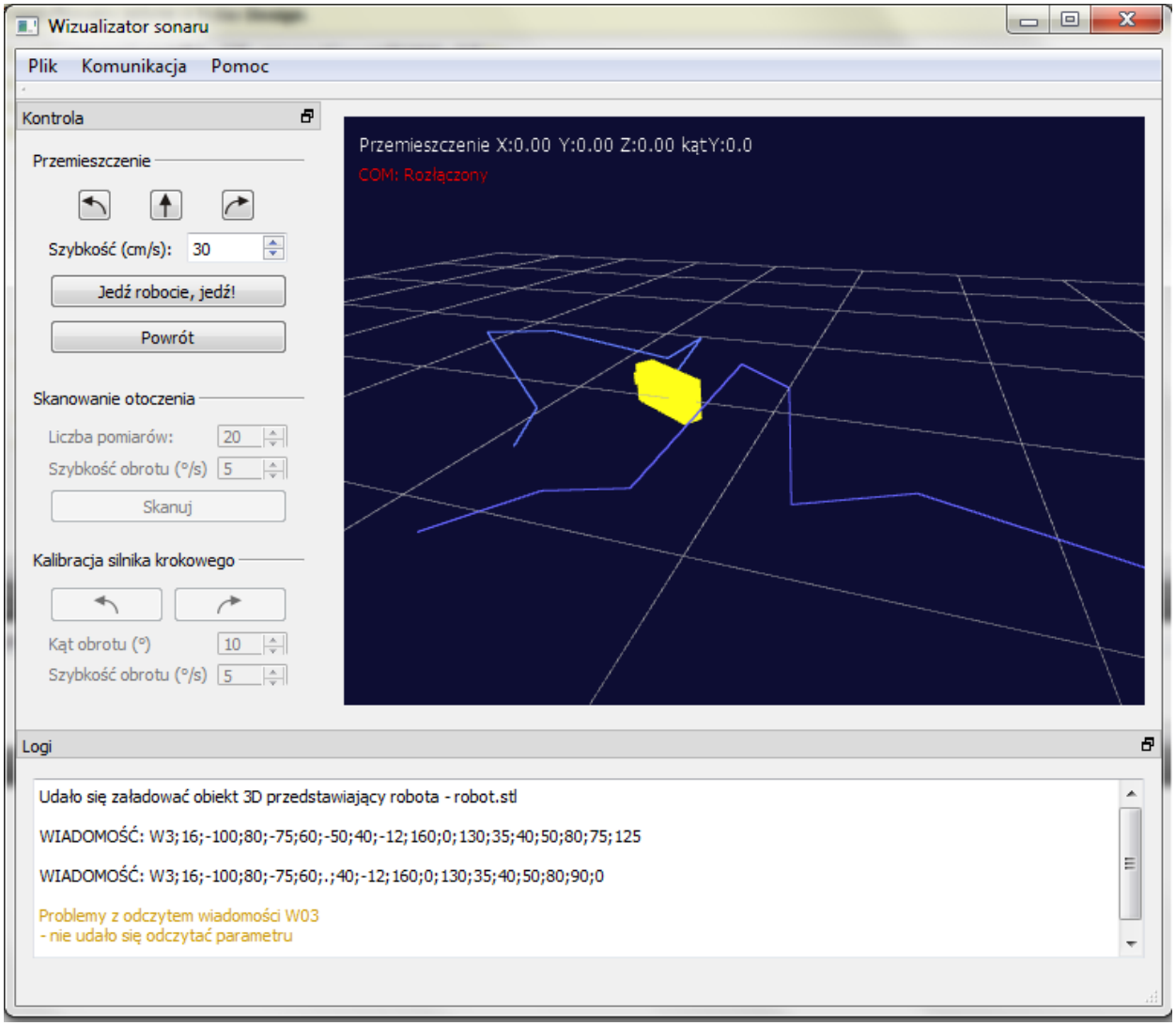
Aplikacja składa się z głównego okna widocznego na rysunku 8. W oknie znajdują się trzy widżety: Kontrola, Logi oraz widok 3D. Oba pierwsze widżety są dokowane i mogą zostać odłączone. Pozwala to na wygodniejsze operowanie lub nawet rozkładanie widoku aplikacji na kilka monitorów (rysunek 9). Widżet **Logi** zawiera jedynie pole tekstowe, którego edytować nie można i w którym pojawiają się wszelkie informacje na temat statusu operacji oraz aplikacji. Zostały wyróżnione różnego typu komunikaty:

- kolor czarny - zwykła informacja
- kolor pomarańczowy - ostrzeżenie
- kolor czerwony - błąd (ale nie krytyczny)

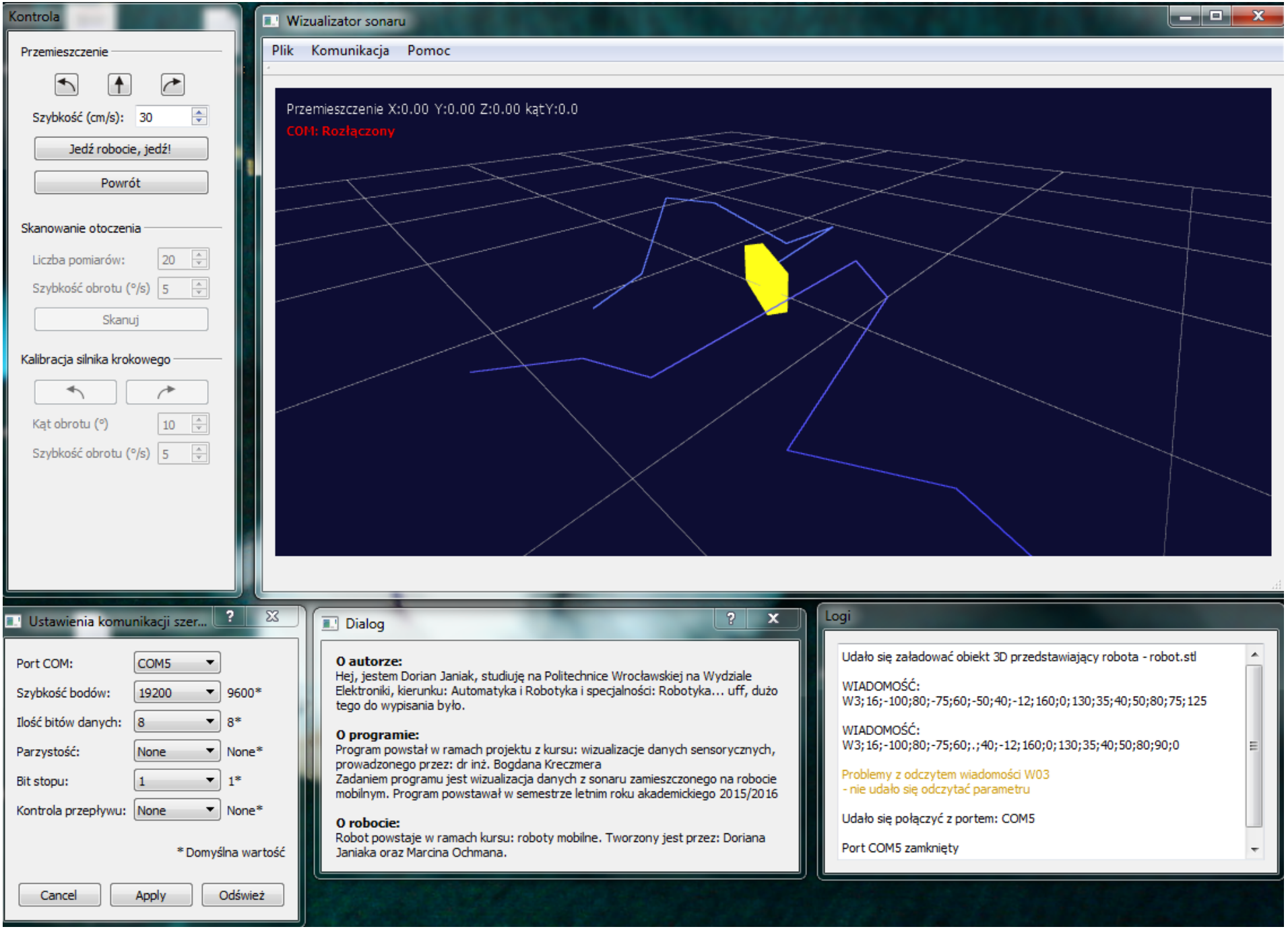
Widżet **Kontrola** składa się z grup:



Rysunek 7: Ostateczny wygląd robota



Rysunek 8: Wygląd aplikacji



Rysunek 9: Otwarte kilka okien oraz widżetów - otwieranie na kilku monitorach

- Przemieszczenie - gdzie udostępnione zostały trzy przyciski do sterowania pozycją robota w jakiej chcielibyśmy, aby ostatecznie się znalazł. Aby nakazać robotowi przemieszczenie się do zadanej pozycji należy nacisnąć przycisk *Jedź robocie, jedź!* (Choć należałoby się zastanowić czy przycisk nie powinien nazywać się *Robocie, niech ktoś cię przesunie!* w kontekście niezamontowanych kół do robota). W przeciwnym wypadku robot się nie znajdzie w zadanej pozycji i przy najbliższym skanowaniu otoczenia lub przy naciśnięciu przycisku *Powrót* robot powróci do ostatniej zapamiętanej pozycji (takiej, do której miał dojechać). Można zadać również szybkość jazdy robota. Robot nie musi jej obsłużyć.
- Skanowanie otoczenia - grupa pozwala na nakazanie robotowi wykonania skanu otoczenia. Skanowanie odbywa się w zakresie 90 do -90 stopni. Można zadać ile pomiarów ma się znaleźć w ramach jednego skanowania i z jaką szybkością ma się wykonywać skanowanie (obrót silnika krokowego). Aby grupa była dostępna aplikacja musi połączyć się z robotem.
- Kalibracja silnika krokowego - grupa pozwala na obrócenie silnika krokowego (sterowanie kątem akustycznym czujnika odległościowego) bez wykonywania pomiaru i odczytu mapy otoczenia. Aby grupa była dostępna aplikacja musi połączyć się z robotem.

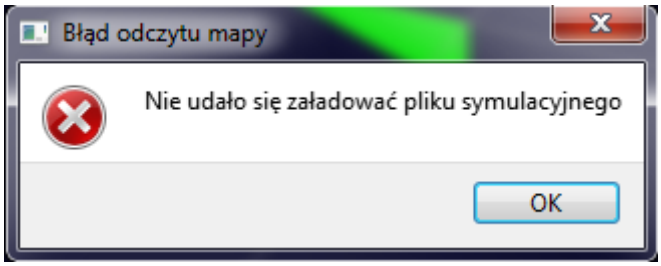
10.2 Widok 3D

Główną część okna stanowi **widok 3D**, w którym przy użyciu myszy komputerowej można sterować kątem kamery (LPM + ruch), jej przybliżeniem (rolka) oraz przemieszczeniem jej centralnego punktu (PPM + ruch). W oknie tym rysowana jest mapa 3D. Widok jest renderowany przy użyciu klas **QOpenGLWidget** (od wersji Qt 5.4 wyparła QGLWidget) oraz **QOpenGLFunctions**, obsługujących OpenGL. W oknie pojawiają się takie obiekty jak:

- wyniki skanowania - widoczne na screenie (linie w odcieniach niebieskiego). Każdy dodatkowy pomiar jest podnoszony względem poprzedniego (wzdłuż osi Y) oraz jest rozjaśniany jego kolor. Pozwala to odróżnić kolejne pomiary od siebie. Jest to w szczególności przydatne gdy okaże się, że w trakcie jednego skanowania zostanie zarejestrowany pomiar bardziej odległy niż 3m - wtedy program odrzuca taki pomiar i siatka zostaje rozerwana w tym miejscu. Wtedy mimo kilku osobnych podsiatek można zauważyć, że pochodzą one z tego samego pomiaru, ponieważ są na jednakowej wysokości oraz o jednakowym kolorze.
- robot - robot jest symbolizowany przez żółty obiekt. W praktyce można jednak obiekt ten zmienić poprzez podmianę pliku **objects/robot.stl**.
- siatka - domyślnie jedna kratka odpowiada kwadratowi o boku 1 metru.

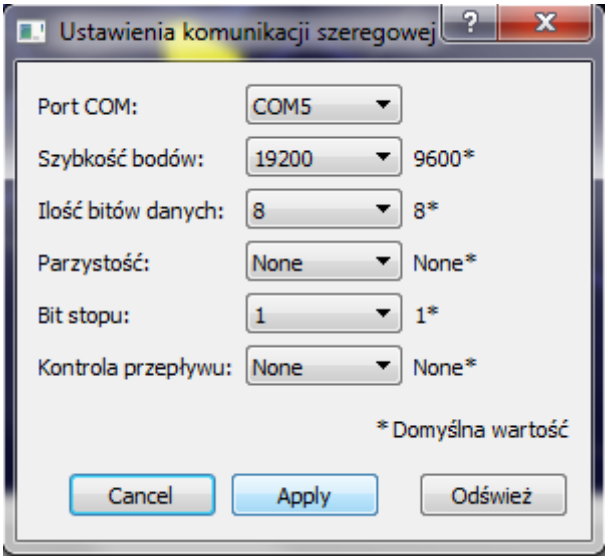
10.3 Dodatkowe okna

Poważniejsze błędy, wymagające uwagi użytkownika są raportowane okienkiem błędu.



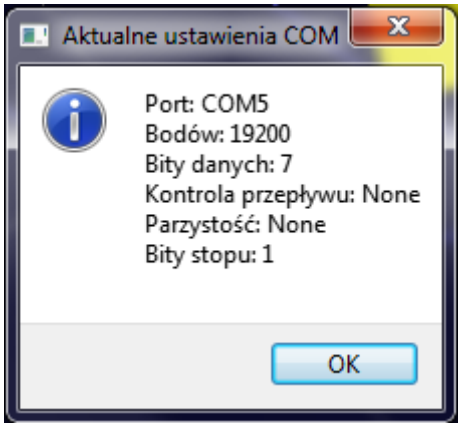
Rysunek 10: Komunikat błędu

Po wybraniu opcji *Konfiguracja* z menu *Komunikacja* wyświetla poniżej przedstawione okno konfiguracji połączenia szeregowego.



Rysunek 11: Konfiguracja połączenia szeregowego

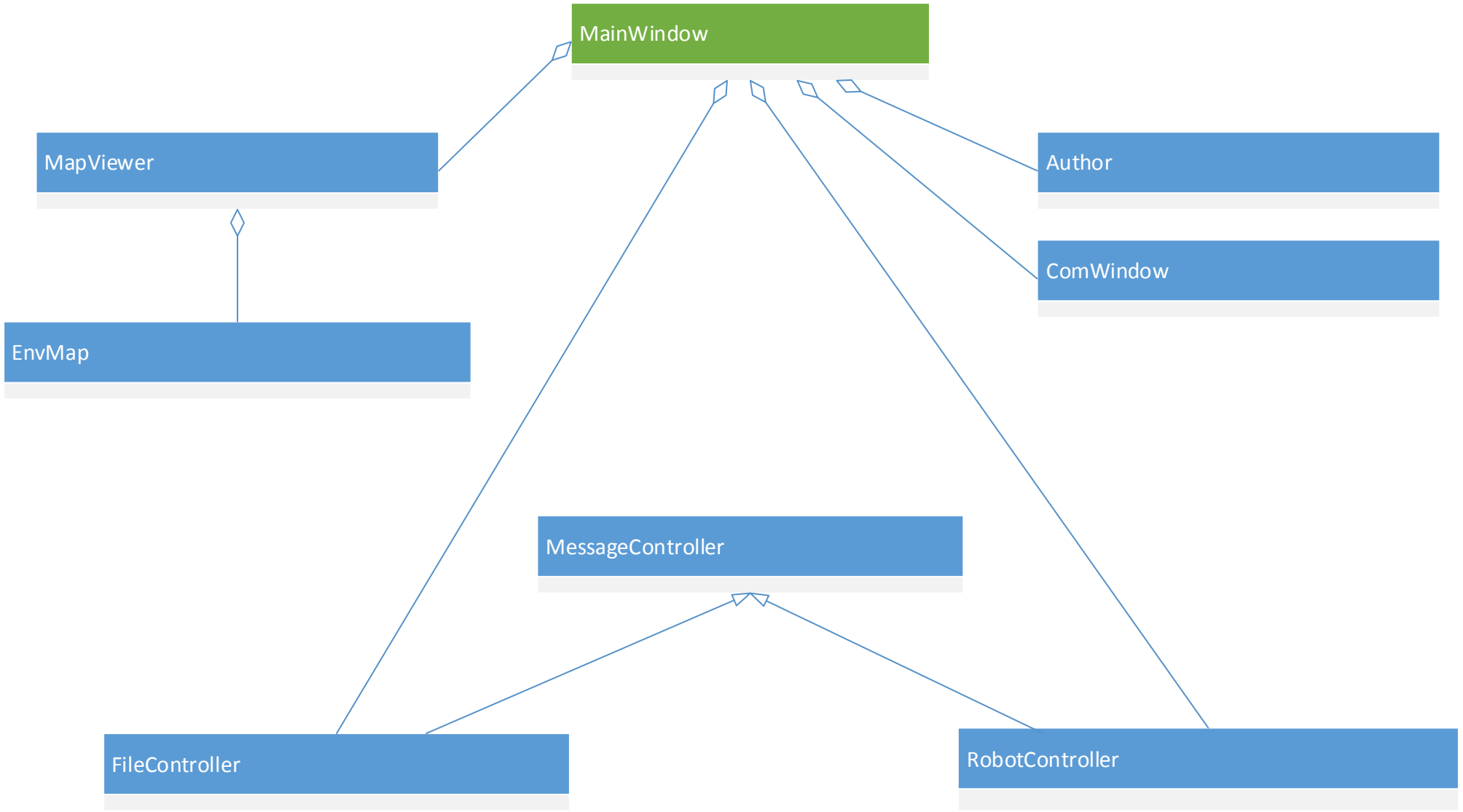
Po wybraniu opcji *Wyświetl informacje* z menu *Komunikacja* wyświetla się poniższe okno z podsumowaniem aktualnie ustawionych parametrów komunikacji poprzez port szeregowy.

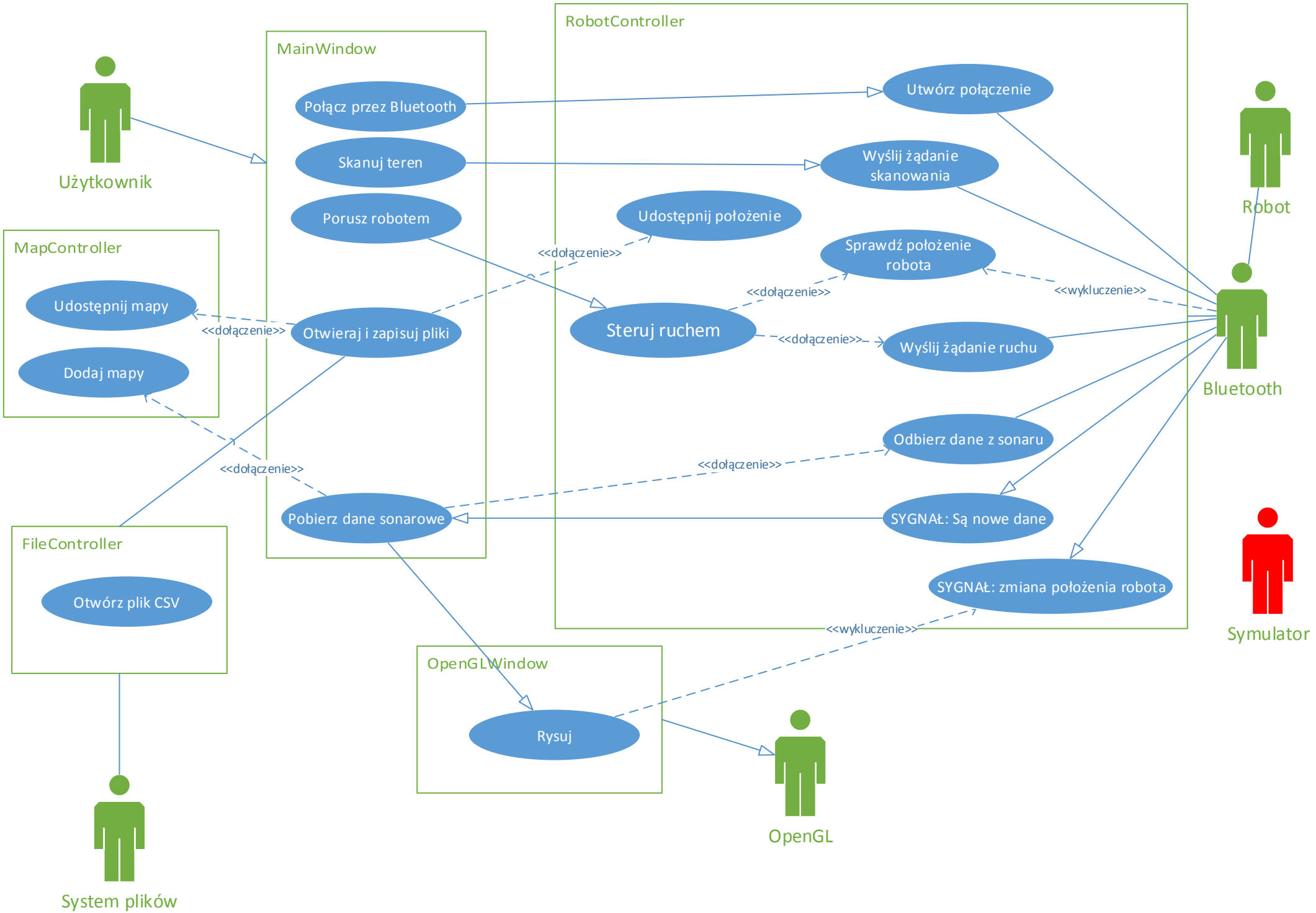


Rysunek 12: Aktualnie ustawiona konfiguracja połączenia szeregowego

10.4 Diagramy

Poniżej zostały zaprezentowane diagramy klas oraz przypadków użycia dla aplikacji komputerowej.


MainWindow
MapViewer
Author
ComWindow
EnvMap
MessageController
FileController
RobotController


RobotController
MainWindow
Użytkownik
Połącz przez Bluetooth
Skanuj teren
Porusz robotem
Utwórz połączenie
Wyślij żądanie skanowania
Udostępnij położenie
Sprawdź położenie robota
Steruj ruchem
Wyślij żądanie ruchu
Odbierz dane z sonaru
SYGNAŁ: Są nowe dane
SYGNAŁ: zmiana położenia robota
Robot
Bluetooth
Symulator
MapController
Udostępnij mapy
Dodaj mapy
Otwieraj i zapisuj pliki
<<dołączenie>>
<<wykluczenie>>
FileController
Otwórz plik CSV
Pobierz dane sonarowe
OpenGLWindow
Rysuj
OpenGL
System plików

11 Test robota

W ramach testu przeprowadzone zostało skanowanie mieszkania. Przeskanowane zostały 3 pokoje. Rysunek 13 przedstawia wynik skanowania. Bez znajomości geometrii pomieszczeń trudno wywnioskować jak wygląda pomieszczenie na podstawie samego skanu sonaru. Rysunek 14 przedstawia ten sam wynik skanowania z naniesionymi dodatkowo informacjami nt. geometrii pomieszczeń. Jak widać wynik okazuje się całkiem zbieżny z rzeczywistą geometrią. Należy również zauważyć, że przesuwanie robota mogło nanieść spore błędy na wynik skanu.

12 Podsumowanie

Niestety, nie udało nam się w całości zrealizować projektu opisanego w pierwszym dokumencie z założeniami projektowymi. Nie zrealizowaliśmy funkcjonalności poruszania się robota. Przyczyną był źle ułożony harmonogram, który zakładał, że zrobimy znacznie więcej niż byliśmy w stanie wykonać. Praca nad innymi, równie ambitnymi projektami w tym semestrze spowodowała, że prace nad robotem posuwały się niewystarczająco szybko, co spowodowało, że nie byliśmy w stanie zbudować robota na czas. Jesteśmy jednak zadowoleni z wykonanej pracy, ponieważ robot daje możliwość skanowania otoczenia i jego wizualizacji w trybie 3D.

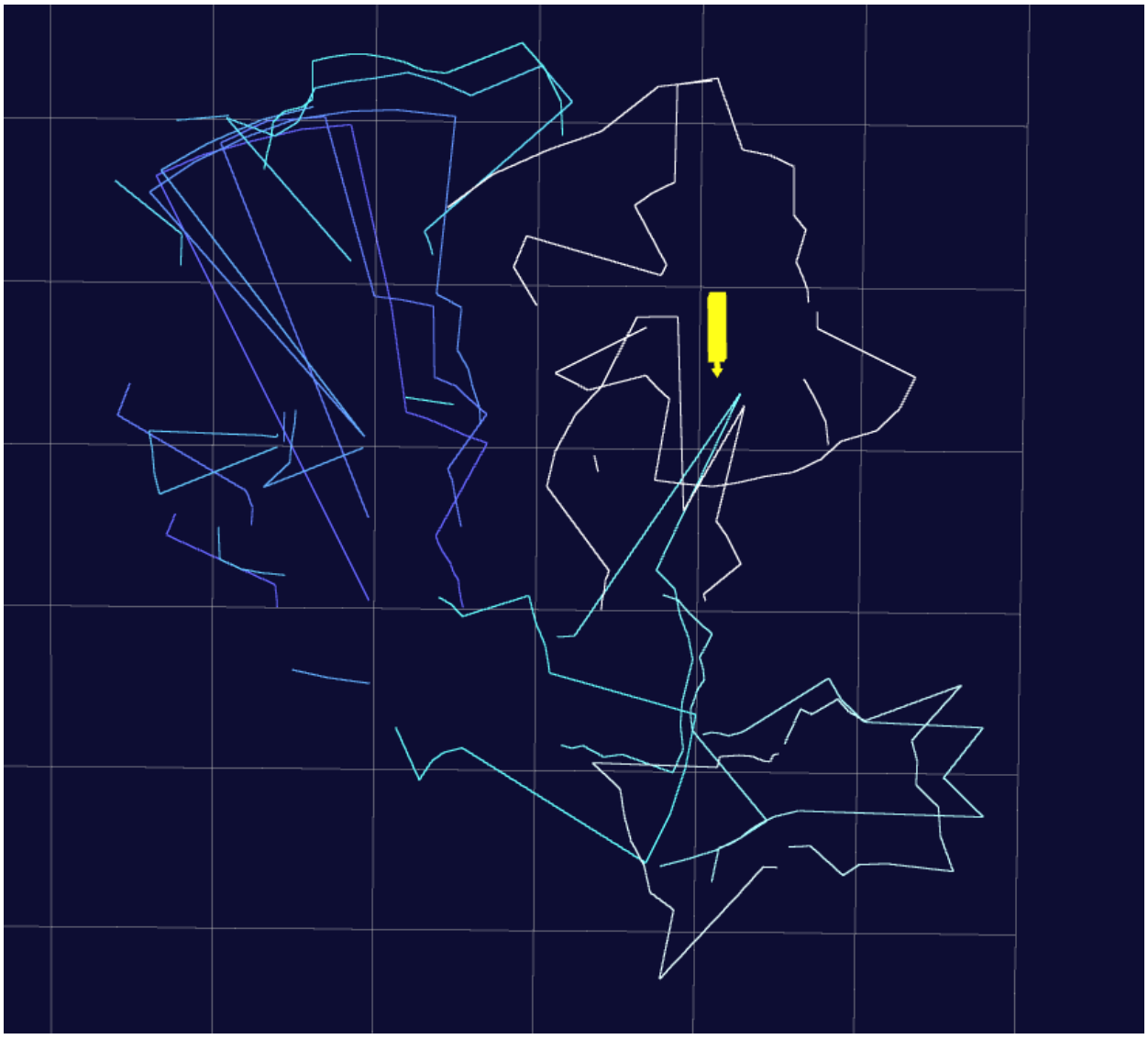
13 Materiały źródłowe

Naszym głównym źródłem informacji do stworzenia aplikacji były strony dokumentacji technicznej bibliotek - *Qt5* oraz *OpenGL*. Strony główne zostały zamieszczone poniżej:

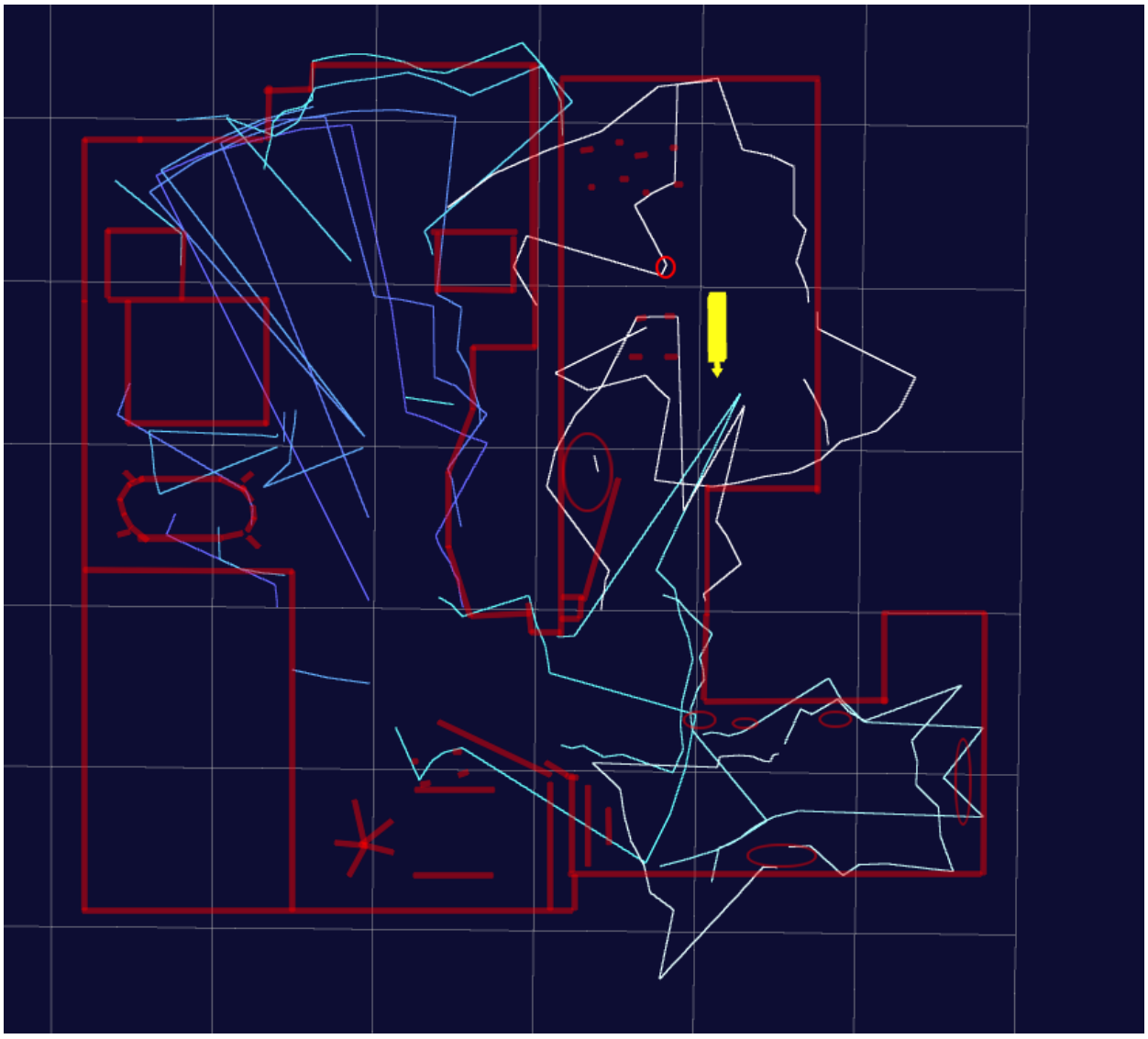
- http://doc.qt.io/qt-5/reference-overview.html
- https://www.opengl.org/documentation/

Natomiast do obsługi modułów robota zosały wykorzystane informacje zawarte pod poniższymi adresami:

- http://www.mlodedrwale.pl/2013/07/05/tani-modul-bluetooth-cz1/
- http://www.python.rk.edu.pl/w/p/sterowanie-silnikami-krokowymi-i-serwomechanizmami-za-pomoca-pymcu/
- http://www.bajdi.com/adding-encoders-to-those-cheap-yellow-motors/



Rysunek 13: Wynik skanowania



Rysunek 14: Wynik skanowania z naniesioną geometrią pomieszczeń