Wizualizacja danych sonaru

Dorian Janiak

11.06.2015

1 Krótki opis

Projekt zakładał stworzenie aplikacji komputerowej, która będzie wizualizowała otrzymywane dane z sonaru ultradźwiękowego. Aplikacja ma próbować łączyć dane w taki sposób, aby móc z nich stworzyć zarys otoczenia. Drugą część projektu stanowi symulacja, której zadaniem jest z odpowiednio przygotowanych obiektów geometrycznych wygenerować dane potrzebne do wykonania wizualizacji (ma być symulacją sonaru). Dane mają zostać przesłane przy pomocy wybranego protokołu komunikacyjnego.

2 Cel

Celem projektu było zapoznanie się ze środowiskiem Qt oraz sposobami wizualizacji danych przy użyciu jego klas. Jednym z problemów, który został również poruszony w ramach pracy był sposób komunikacji modułów i przesył danych. Jednym z ważnych powodów podjęcia się realizacji tego tematu była chęć głębszego poznania biblioteki OpenGL.

3 Opis projektu

Projekt składał się z dwóch części. Pierwsza polegała na stworzeniu aplikacji głównej dla komputerów PC przy użyciu środkowiska Qt. Druga polegała na stworzeniu aplikacji symulującej robota uruchamianej na telefonie z systemem Android.

3.1 Aplikacja główna

Aplikacja główna powstała w środowisku Qt. Pozwala ona użytkownikowi na połączenie się z robotem lub symulatorem, a następnie sterowanie nim. Połączenie ustanawiane jest przy użyciu obiektu klasy QSerialPort, który obsługuje **port szeregowy**. Po połączeniu się z urządzeniem możliwe są trzy operacje:

- przemieszczenie robota,
- kalibracja silnika krokowego (zmiana kąta akustycznego sonaru)
- wykonanie skanowania otoczenia

Status wykonywanych operacji publikowany jest w oknie logów. Dzięki temu użytkownik dowiaduje się czy ustanowiono połączenie i z jakim portem oraz czy operacje, których wykonania zażądał rzeczywiście się wykonały.
Rezultaty skanowania otoczenia przedstawiane są w głównym oknie aplikacji komputerowej. Wizualizowane są przy użyciu odziedziczonych klas QOpenGLWidget oraz QOpenGLFunctions, które obsługują bibliotekę **OpenGL**.

3.2 Symulator robota

Aplikacja symulująca robota powstała w środowisku Android Studio i przeznaczona jest na telefony komórkowe z systemem Android w wersji nie niższej niż 2.3 . Aplikacja pozwala na załadowanie pliku OBJ zawierającego mapę otoczenia. Następnie po naciśnięciu przycisku 'Zasymuluj' otrzymywane są rezultaty skanowania otoczenia z zadanej pozycji robota. Skanowanie otoczenia odbywa się najprostszą metodą (poprzez prowadzenie prostych linii i sprawdzanie przecięć z krawędziami).
W ramach projektu symulator miał komunikować się z komputerem. Niestety nie udało mi się tego zrealizować. W trakcie realizacji projektu skupiłem się na tym, aby aplikacja komputerowa zadziałała z robotem. Ponieważ z robotem można się komunikować jedynie przy użyciu portu szeregowego, a udało mi się znaleźć informacje według, których telefon również można połączyć z komputerem PC poprzez port szeregowy, postanowiłem skorzystać z klasy QSerialPort zamiast QtBluetooth. Jest to rzeczywiście możliwe, ale w trakcie realizacji projektu okazało się, że oba modele telefonów, na których mogłem testować symulator **nie posiadają funkcji SPP**, która właśnie odpowiada za możliwość takiego łączenia urządzenia z komputerem PC.

3.2.1 Możliwe rozwiązania

Ponieważ nie udało mi się rozwiązać problemu ze względu na ograniczenia czasowe, postanowiłem znaleźć możliwe rozwiązania:

- Pierwsze rozwiązanie polega na stworzeniu dodatkowego modułu w aplikacji głównej, który dziedziczy po klasie **QtBluetooth** i stworzeniu komunikacji na zasadzie serwer-klient. Zaletą rozwiązania jest zachowanie spójności części projektu przeznaczonej na PC, jednak można liczyć się z pogorszeniem czytelności zarówno GUI aplikacji jak i jej implementacji.
- Drugie rozwiązanie polega na stworzeniu drugiej aplikacji komputerowej, która jedynie łączy się poprzez Bluetooth z urządzeniem i przekierowuje odbierane i wysyłane dane na wirtualny port szeregowy. Rozwiązanie nie zachowuje spójności części projektu przeznaczonej na PC, ponieważ użytkownik zmuszony byłby do uruchamiania obu aplikacji oraz symulatora. Zaletą natomiast jest fakt, że zachowana byłaby czytelność głównej aplikacji komputerowej.
- Trzecie rozwiązanie polega na stworzeniu symulatora na komputerze PC tak jak to było w pierwotnych założeniach projektu i wysyłanie danych poprzez wirtualny port szeregowy.
- Czwarte rozwiązanie polega na znalezieniu telefonu z SPP oraz dokończeniu symulatora robota.

Najbardziej racjonalne rozwiązania wydają się **pierwsze i trzecie**, natomiast najmniej czwarte. To ostatnie odpada, ponieważ wiąże się to z nakładami finansowymi. Poza tym w takiej sytuacji użytkownikowi należałoby narzucić wymóg posiadania telefonu z funkcją SPP, a to zniechęciłoby potencjalnych zainteresowanych do korzystania z aplikacji.

3.3 Robot

Robot powstał w ramach projektu z kursu Roboty Mobilne. Jednak na nim znajduje się gotowa aplikacja, która jest w stanie przeskanować otoczenie oraz zasymulować jazdę robota. Aplikacja główna bardzo dobrze współpracuje z rzeczywistym robotem. Dzięki zastosowaniu modułu Bluetooth HC-05 robot komunikuje się z komputerem PC bezprzewodowo, a mimo to połączenie widziane jest jako połączenie szeregowe.

4 Funkcjonalność

4.1 Funkcjonalność skończona

Poniżej zamieszczona została lista funkcjonalności, którą udało się ukończyć w ramach projektu.

4.1.1 Aplikacja na komputerze PC

- Rysowanie na scenie 3D mapy otoczenia.
- Wczytywanie danych pomiarowych sonaru z pliku (w przypadku symulacji).
- Łączenie nowo odczytanej z poprzednio odczytaną mapą terenu.
- Możliwość sterowania widokiem 3D (rotacja oraz translacja).
- Wyświetlanie logów dotyczących przesyłanych komunikatów oraz interpretacja występujących błędów.
- Rysowanie na scenie 3D robota załadowanego z pliku STL.
- Udostępnienie pilota pozwalającego na sterowanie robotem (lub symulacją).
- Udostępnienie okna konfiguracji portów COM połączenia szeregowego.
- Połączenie z portem szeregowym oraz obsługa komunikacji.
- Wysłanie żądania i weryfikacja wiadomości zwrotnej **przemieszczenia robota**
- Wysłanie żądania i weryfikacja wiadomości zwrotnej **skanowania otoczenia**
- Wysłanie żądania i weryfikacja wiadomości zwrotnej **zmiany kąta akustycznego czujnika**

4.1.2 Symulator na urządzeniu

- Ładowanie plików OBJ, parsowanie oraz raportowanie wczytanych informacji
- Wykonanie skanowania otoczenia przy użyciu bardzo prostego algorytmu oraz wyświetlenie wyników skanowania.
- Nawiązanie połączenia Bluetooth z komputerem PC.

4.2 Funkcjonalność nieskończona

Poniżej zamieszczone została lista funkcjonalności, których nie udało się ostatecznie wykonać.

4.2.1 Symulator na urządzeniu

- Wysłanie danych do aplikacji wizualizującej poprzez interfejs komunikacyjny (Bluetooth)
- Odbieranie żądania przemieszczenia robota na scenie i przesłanie wyniku operacji
- Odbieranie żądania wykonania symulacji i przesłania wyników operacji

5 Protokół komunikacji

Na potrzeby realizacji projektu powstał protokół komunikacji, bazujący na poniższych typach wiadomości:

- 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

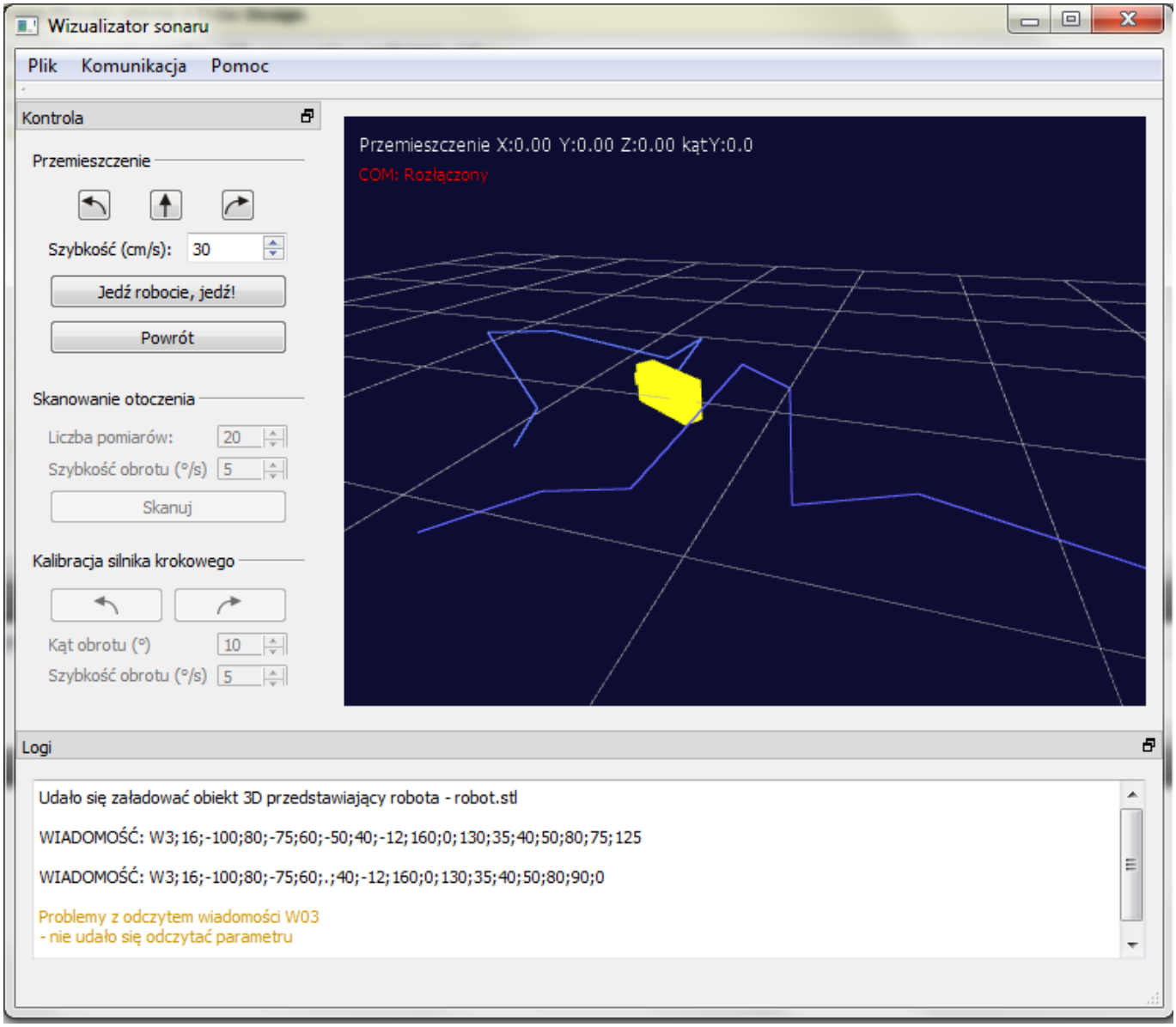
- Rezultat wykonania przemieszczenia robota, który jest odpowiedzią na wiadomość W5.

  `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`



Rysunek 1: Wygląd aplikacji

6 Wygląd aplikacji

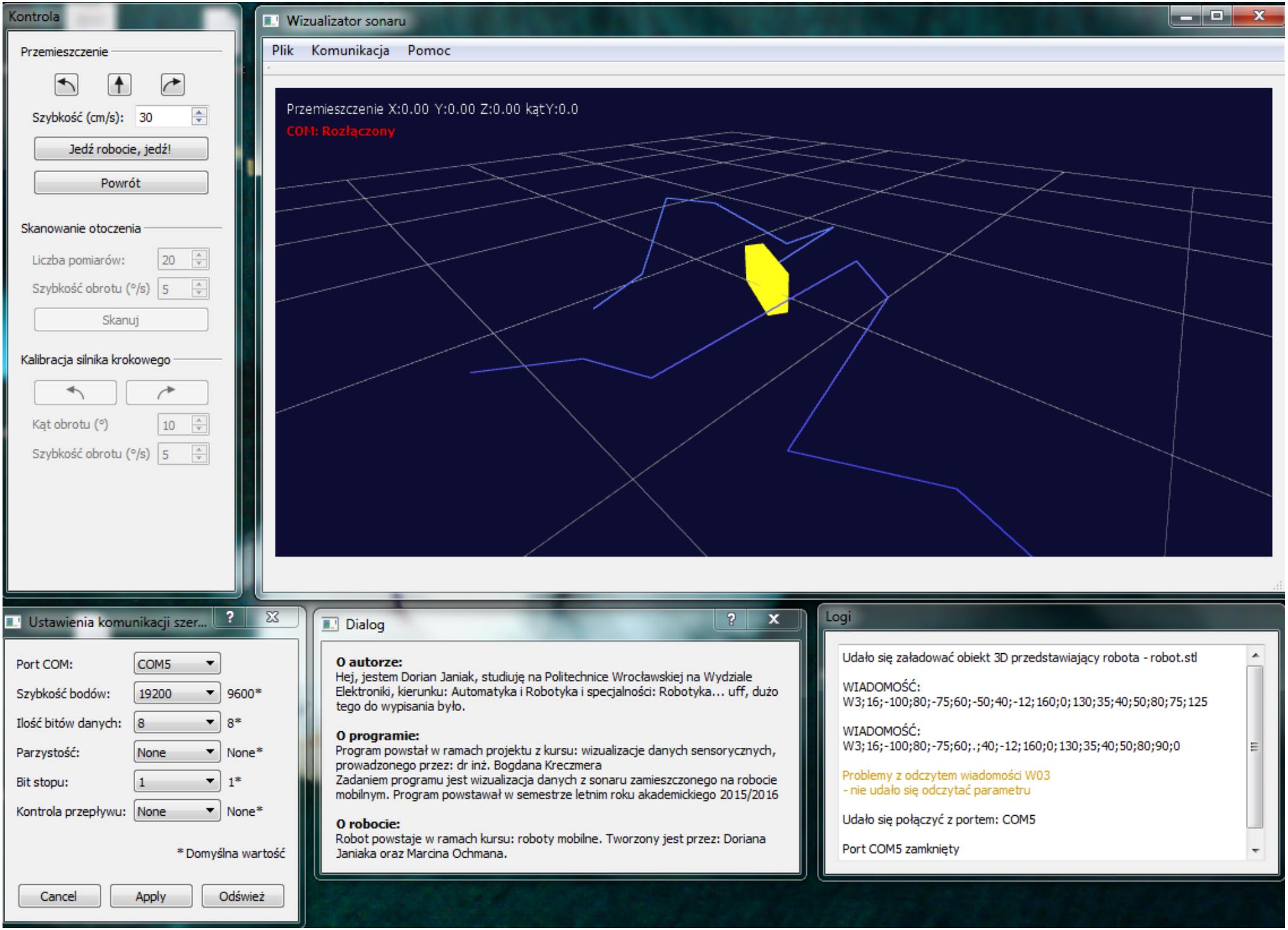
6.1 Aplikacja główna

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

6.1.1 Widżety dokowane

Aplikacja składa się z głównego okna widocznego na rysunku 1. 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 2). 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



Rysunek 2: Otwarte kilka okien oraz widżetów - otwieranie na kilku monitorach

- kolor czerwony - błąd (ale nie krytyczny)

Widżet **Kontrola** składa się z grup:

- 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.

6.1.2 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:

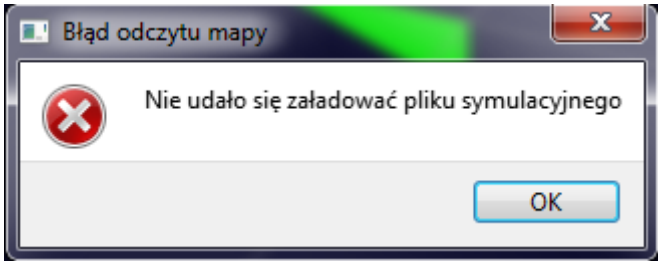
- wyniki skanowania - widoczne na screenie (linie w odcieniach niebieskiego). Każdy dodatkowy pomiar jest podnoszony względem poprzedniego (wzdłuż osi Y) oraz jest rozjaśniany jego kolor. Pozwala to odróżnić kolejne pomiary od siebie. Jest to w szczególności przydatne gdy okaże się, że w trakcie jednego 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.

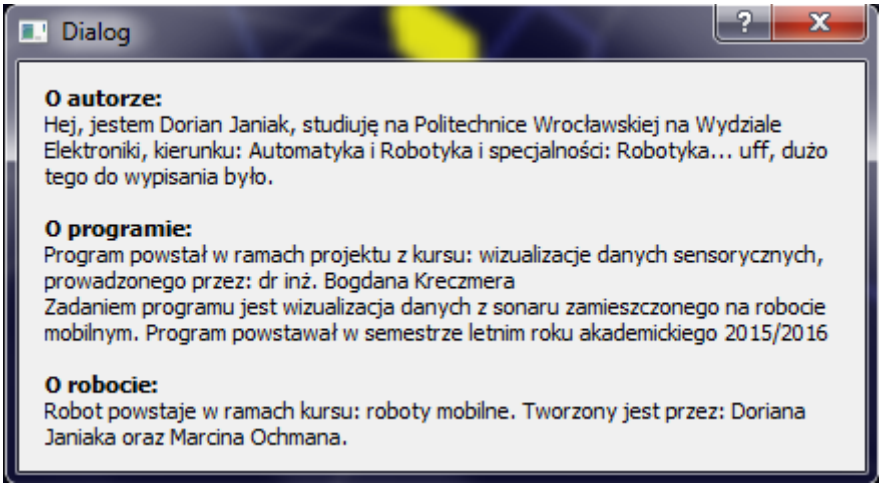
6.1.3 Dodatkowe okna

Poważniejsze błędy, wymagające uwagi użytkownika są raportowane okienkiem błędu.



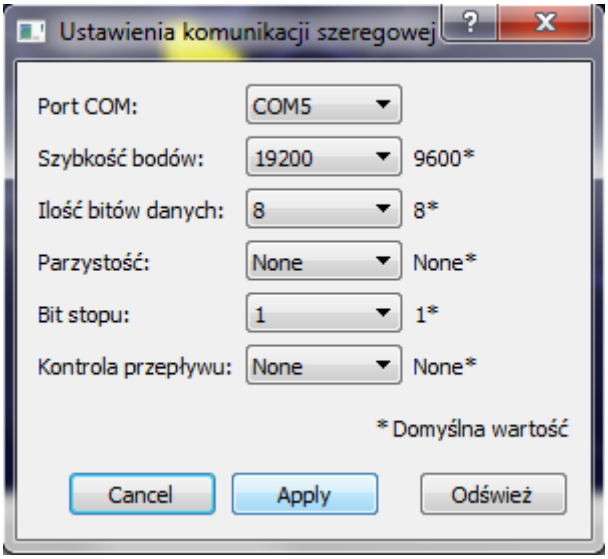
Rysunek 3: Komunikat błędu

Po wybraniu odpowiedniej opcji z paska menu otwiera się okno z informacjami o autorstwie.



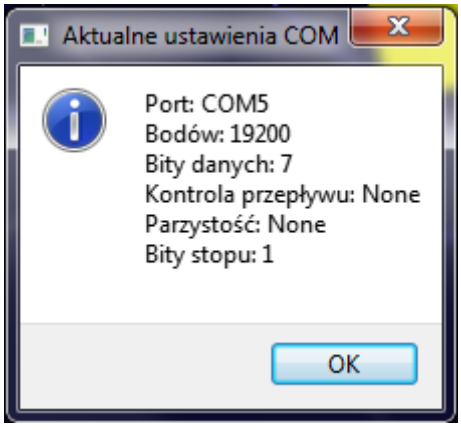
Rysunek 4: Informacje o autorze i projektach

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



Rysunek 5: Konfiguracja połączenia szeregowego

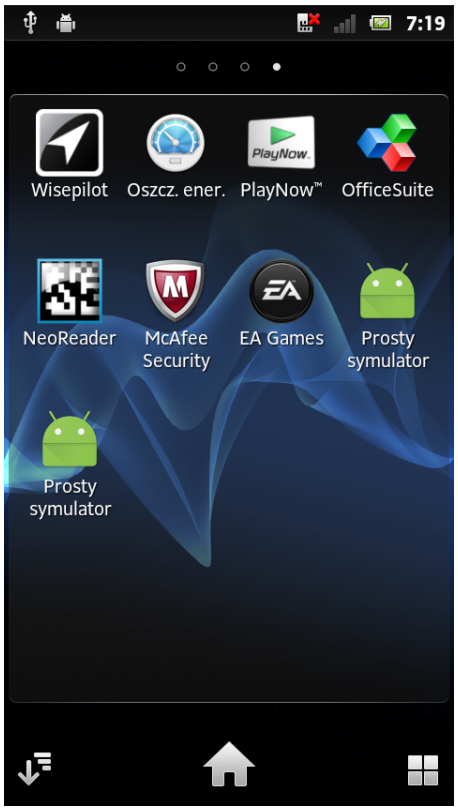
Po wybraniu opcji *Wyświetl informacje* z menu *Komunikacja* wyświetla się poniższe okno z podsumowaniem aktualnie ustawionych parametrów komunikacji poprzez port szeregowy.



Rysunek 6: Aktualnie ustawiona konfiguracja połączenia szeregowego

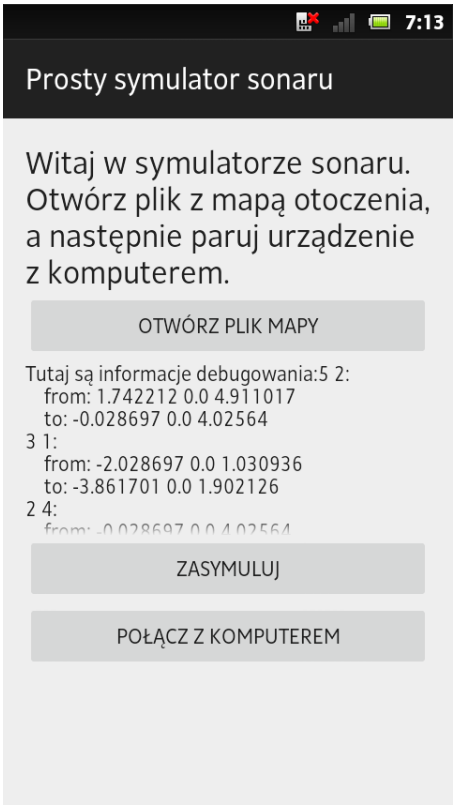
6.2 Symulator

Symulator nie został ostatecznie połączony poprzez Bluetooth z komputerem, a więc nie pełni swojej funkcji. Jednak mimo to powstał już wstępny prosty intefejs graficzny, który pozwala na wczytywanie danych wejściowych i wyświetlenie wyniku skanowania.
Poniższy screen przedstawia ikonę aplikacji w menu Android na telefonie Sony Xperia U.



Rysunek 7: Menu Android z aplikacją Prosty symulator

Okno aplikacji po załadowaniu pliku z mapą otoczenia przedstawia się następująco:

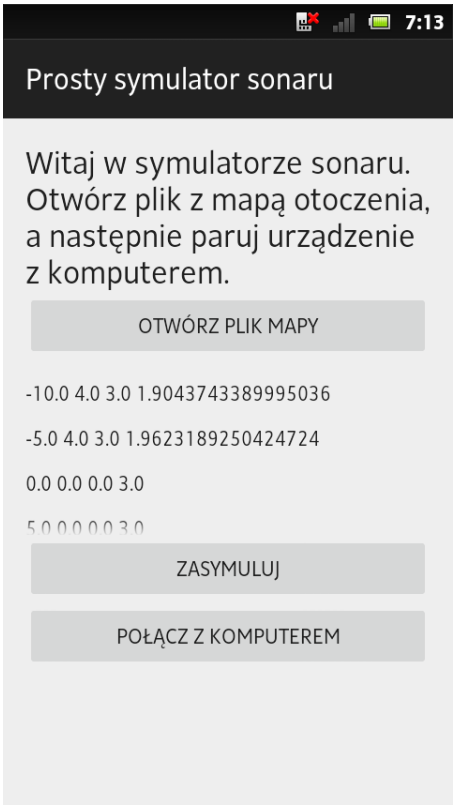


Rysunek 8: Załadowana mapa otoczenia

Aby załadować mapę otoczenia należy nacisnąć przycisk *Otwórz plik mapy*. W takiej sytuacji otworzy się okno (Intent) wyboru pliku OBJ, które zostało stworzone przez Alexandra Ponomarev'a.

Rysunek 9: Wybór pliku OBJ

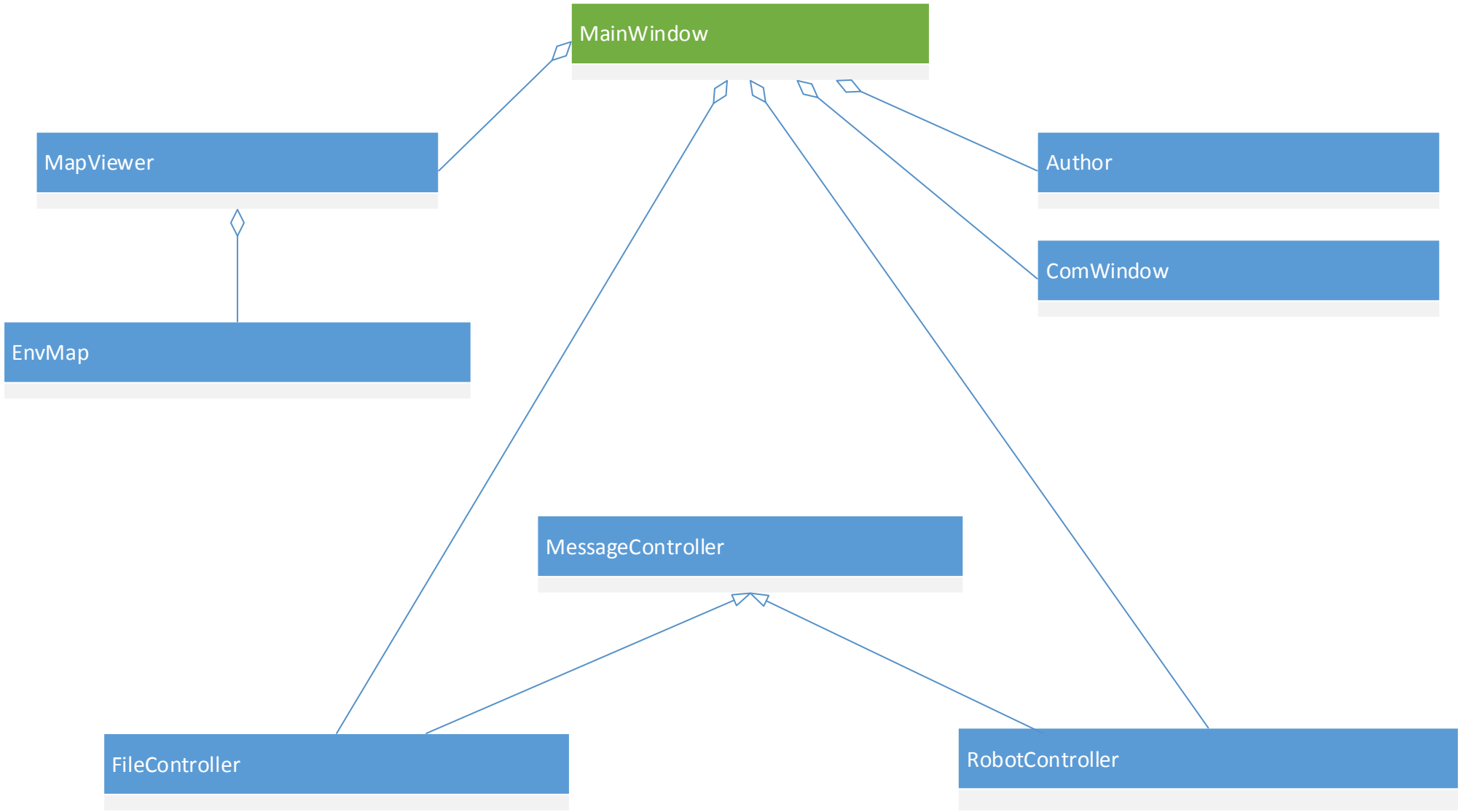
Natomiast po wciśnięciu przycisku *Zasymuluj* zostanie obliczony symulowany wynik skanowania i zaprezentowany tak jak w poniższym oknie:

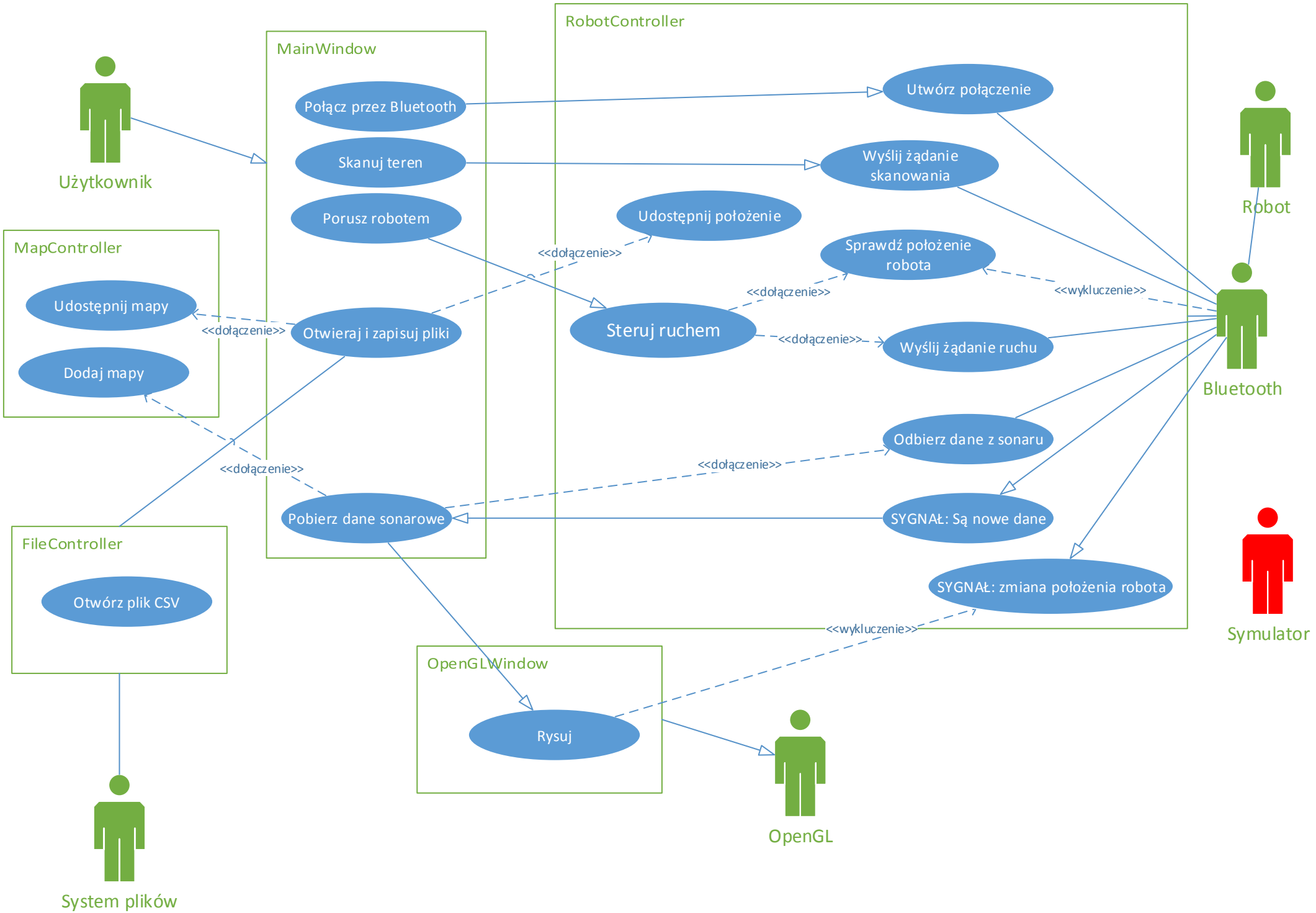


Rysunek 10: Wynik symulacji

7 Diagramy

Poniżej zaprezentowany został diagram klas dla głównej aplikacji komputerowej. Analizę należy rozpocząć od klasy textbfMainWindow, która odpowiada głównemu oknu aplikacji. Przechowuje ona obiekty pozostałych klas. **MessageController** odpowiada za interpretowanie i przygotowywanie wiadomości potrzebnych do komunikacji z urządzeniami oraz odczytem danych z plików. Będzie ona dziedziczona przez klasy **FileController** (obsługuje operacje plikowe) oraz **RobotController** (zarządza robotem). Ta ostatnia zawiera w sobie obiekt klasy **QSocketPort**, która obsługuje komunikację poprzez port szeregowy. Zapewni to klasie RobotController komplet funkcji potrzebnych do sterowania robotem. Klasa **MapViewer** dziedziczy od klasy QOpenGLWidget oraz QOpenGLFunctions i odpowiada za sterowanie widokiem 3D. Klasa **EnvMap** przechowuje komplet informacji związany z danymi skanowania. Przechowuje ona ją w postaci zbioru wierzchołków przestrzennych, które mogą zostać przekazane do kontekstu OpenGL, oraz dodatkowych zmiennych przechowujących informacje związane ze skalą obiektu, koloru materiału, punktu centralnego czy kątu obrotu.
MainWindow przechowuje również obiekty klas **Author** oraz **ComWindow**, które odpowiednio: wyświetlają informację o autorze aplikacji w nowym oknie (powstał do tego celu osobny formularz authors.ui) oraz okno konfiguracji połączenia szeregowego (również osobny formularz comwindow.ui). Na diagramie przypadków natomiast widać, że obiekt, podpisany ”Symulator” został wypełniony kolorem czerwonym, ponieważ nie udało się go połączyć poprzez Bluetooth z komputerem.


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

8 Stanowisko testowe oraz wyniki skanowania

Kompletne stanowisko pracy składało się z laptopa, na którym tworzone były programy w odpowiednich środowiskach:

- aplikacja główna powstawała w środowisku Qt 5.4, przy użyciu aplikacji Qt Creator w wersji 3.3 .
- symulator powstawał w środowisku Android Studio 1.2 przy użyciu API w wersji 22.0.1 (Android 5.1) z kompatybilnością dla wersji 10.0.0 (Android 2.3.3 Gingerbeard).
- program dla robota opartego na zestawie Nucleo powstawał przy użyciu systemu Keil i biblioteki mbed dla stm32f411 w programie Keil microVision5.

Na rysunku 11 przedstawiony został fragment robota, który jest w stanie realizować funkcję skanowania otoczenia oraz symulować przemieszczenie robota.

Na rysunku 12 przedstawione zostało natomiast całe stanowisko pracy przy użyciu którego realizowany był projekt. Widać na nim robota, telefon z uruchomionym symulatorem oraz laptopa, na którym znajduje się aplikacja główna.

8.1 Skanowanie mieszkania

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.

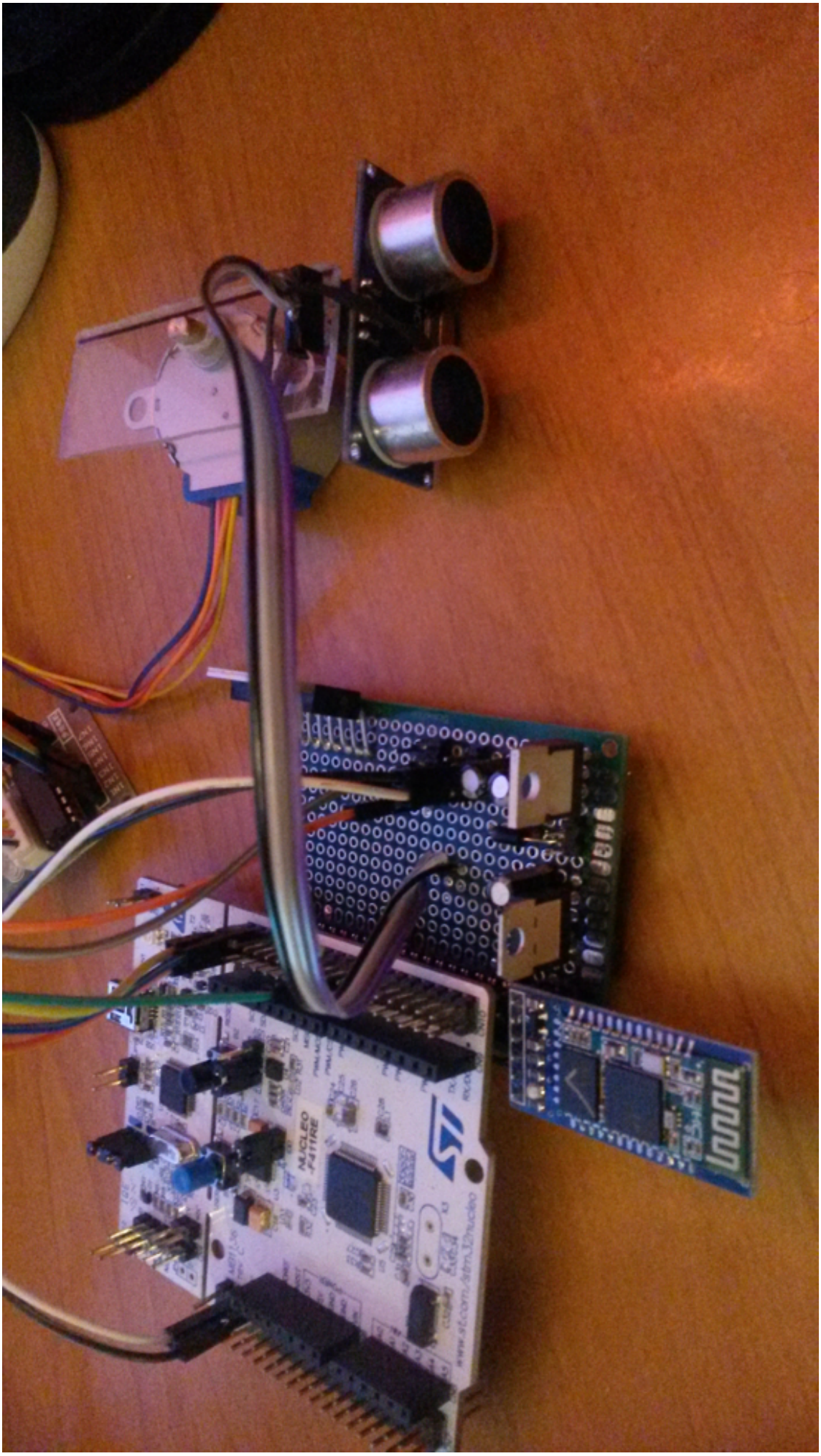
9 Podsumowanie

Projektu nie udało się całkowicie zrealizować tak jak było to od początku zaplanowane. Nie powstał w pełni żaden symulator, który pozwalałby na pracę aplikacji bez robota. Głównym powodem takiego stanu rzeczy było nie przetestowanie możliwości zaimplementowanie połączenia bezprzewodowego. Wnioskiem na przyszłość w tej sytuacji jest to, że zanim rozpocznie się zaawansowane prace nad aplikacją należałoby najpierw wykonać testy skupione na temacie połączeń sieciowych.
Odnośnie samej aplikacji komputerowej - spełniła ona wszystkie założenia, a w sumie nawet więcej niż to było zakładane przed rozpoczęciem realizacji projektu. Dodana została możliwość wyświetlania logów, kalibracji silnika czy też konfiguracji połączenia szeregowego. Należałoby jedynie w niej poprawić komunikację poprzez port szeregowy, aby odbywała się ona na osobnym wątku, ponieważ oczekiwanie na odpowiedź robota blokuje interfejs.
Symulator nie jest w stanie komunikować się z aplikacją komputerową. Zaimplementowany został jednak pełny algorytm potrafiący zasymulować działanie czujnika odległościowego.
Głównym problemem niezaimplementowania do końca funkcji połączenia z symulatorem były ograniczenia czasowe.

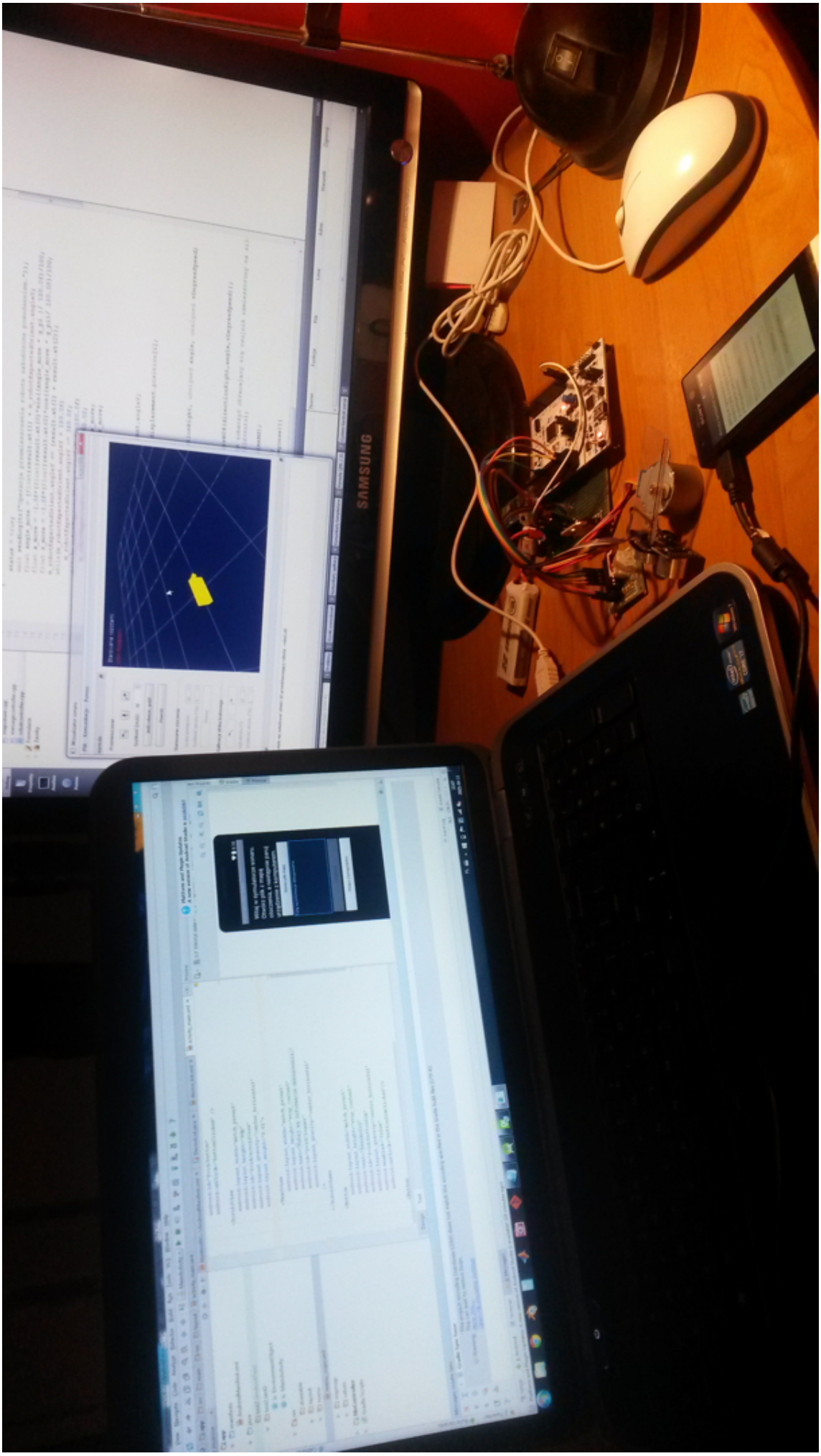
9.1 Subiektywna ocena

Projekt spełnił moje oczekiwania. Brakuje symulatora, który przydałby się tymbardziej, że robot zostanie zdemontowany przez co nie uda się wykorzystać potencjału aplikacji. W ramach projektu zrealizowałem kilka celów, które sobie narzuciłem:

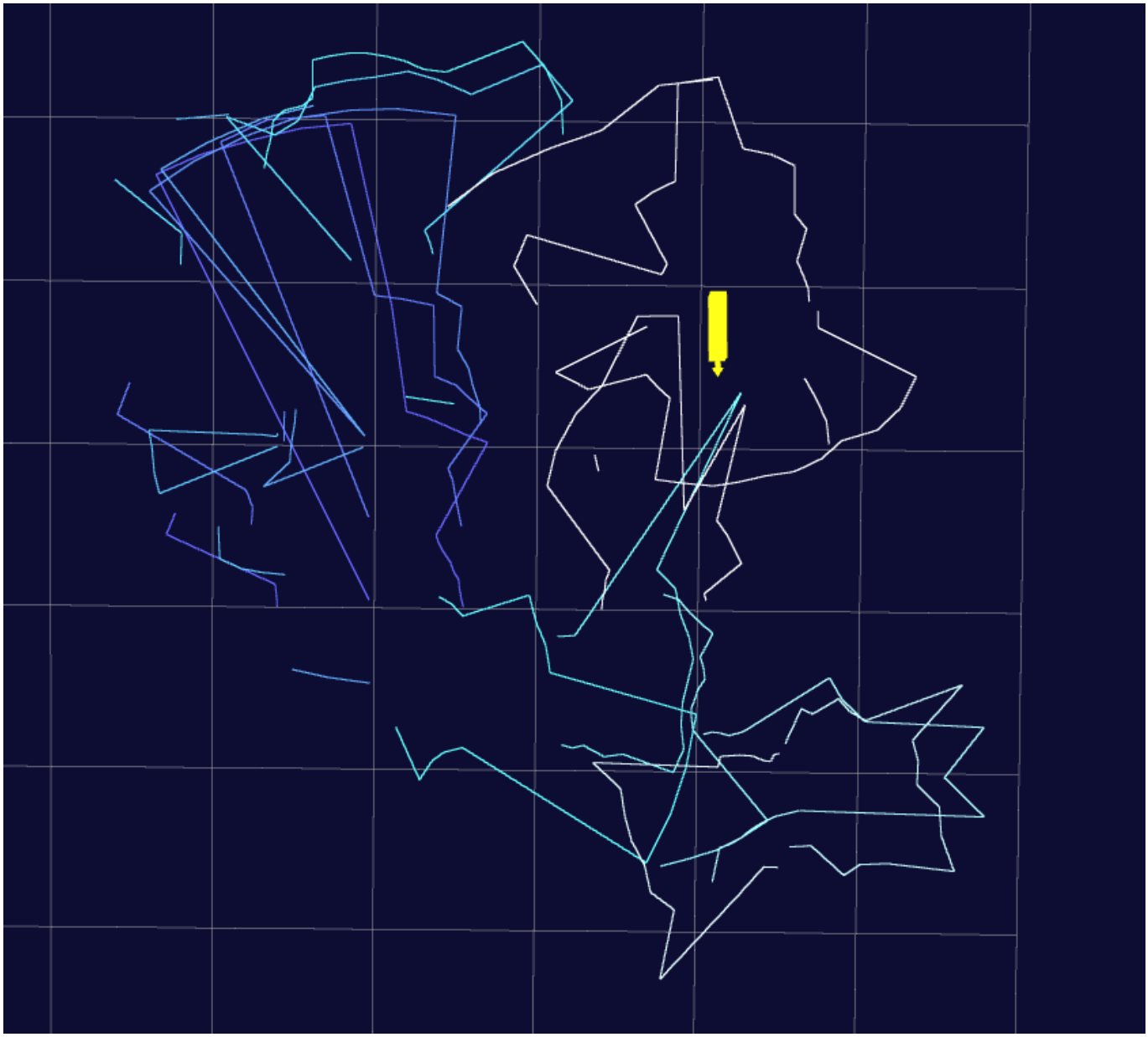
- Stworzyć aplikację, która będzie korzystała z biblioteki OpenGL - grafika 3D to jedno z moich głównych zainteresowań i projekt pozwolił mi zagłębić się w tajniki grafiki generowanej w czasie rzeczywistym.



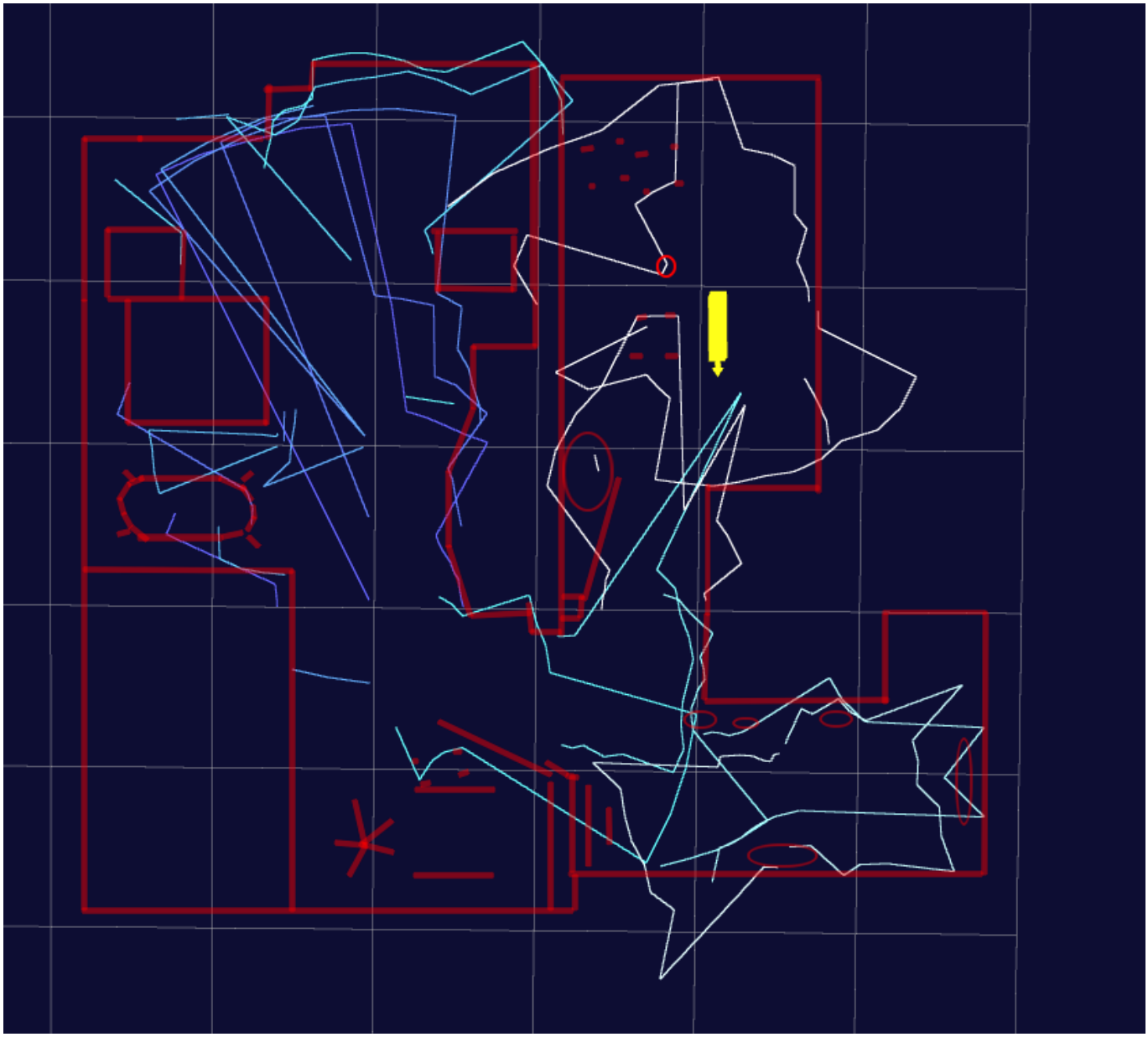
Rysunek 11: Fragment robota



Rysunek 12: Stanowisko pracy



Rysunek 13: Wynik skanowania



Rysunek 14: Wynik skanowania z naniesioną geometrią pomieszczeń

- Zapoznać się z zasadą działania sonaru i wynikami otrzymywanymi z takiego skanera
- Stworzyć stosunkowo łatwą w rozwoju aplikację - jest to moja pierwsza aplikacja, która przed rozpoczęciem tworzenia została w całości zaprojektowana. Dopiero na późniejszych etapach projekt był modyfikowany, ale ogólna koncepcja pozostawała taka sama.
- Zapoznać się z tworzeniem aplikacji na system Android.

Problemem jednak było to, że wszystkie te cele, które sobie narzuciłem były realizowane w ramach jednego projektu, przez co nie udało mi się ukończyć całości projektu w trakcie semestru. Oznacza to, że podejmując się następnych projektów powinienem ostrożniej narzucać sobie wymagania oraz szacować czas potrzebny na realizację zadań.