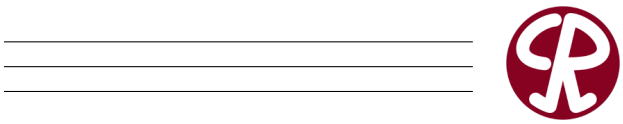
Projekt

Wizualizacja Danych Sensorycznych

RoboVision

Marcin Bober, 249426



Prowadzący:
dr inż. Bogdan Kreczmer

Katedra Cybernetyki i Robotyki
Wydziału Elektroniki
Politechniki Wrocławskiej

24 maja 2021

Spis treści

1 Charakterystyka tematu projektu

Projekt ma na celu stworzenie aplikacji okienkowej, która poprzez połączenie internetowe będzie w stanie wydawać polecenia do robota mobilnego, sterować nim, a także pobierać informacje z czujników i wizualizować je.

2 Podcele i etapy realizacji projektu

Projekt powdzielony będzie na kilka pomniejszych celów tak, aby każdy z nich mógłbyć osobno rozwijany.

Lista podelów:

- Zapoznanie się z dostępną literaturą związaną z tematem oraz zdobycie informacji niezbędnych do zrealizowania zadania.
- Przygotowanie graficznego szkicu aplikacji wraz z rozplanowaniem funkcionalności.
- Zdefiniowanie protokołu komunikacyjnego, struktury ramek przesyłanych danych i implementacja interfejsu sieciowego.
- Parsowanie danych odbieraych z robota.
- Przygotowanie wizualizacji zebranych danych.
- Obsługa joysticka.
- Implementacja algorytmu sterowania i przesyłanie wyników do urządzenia.

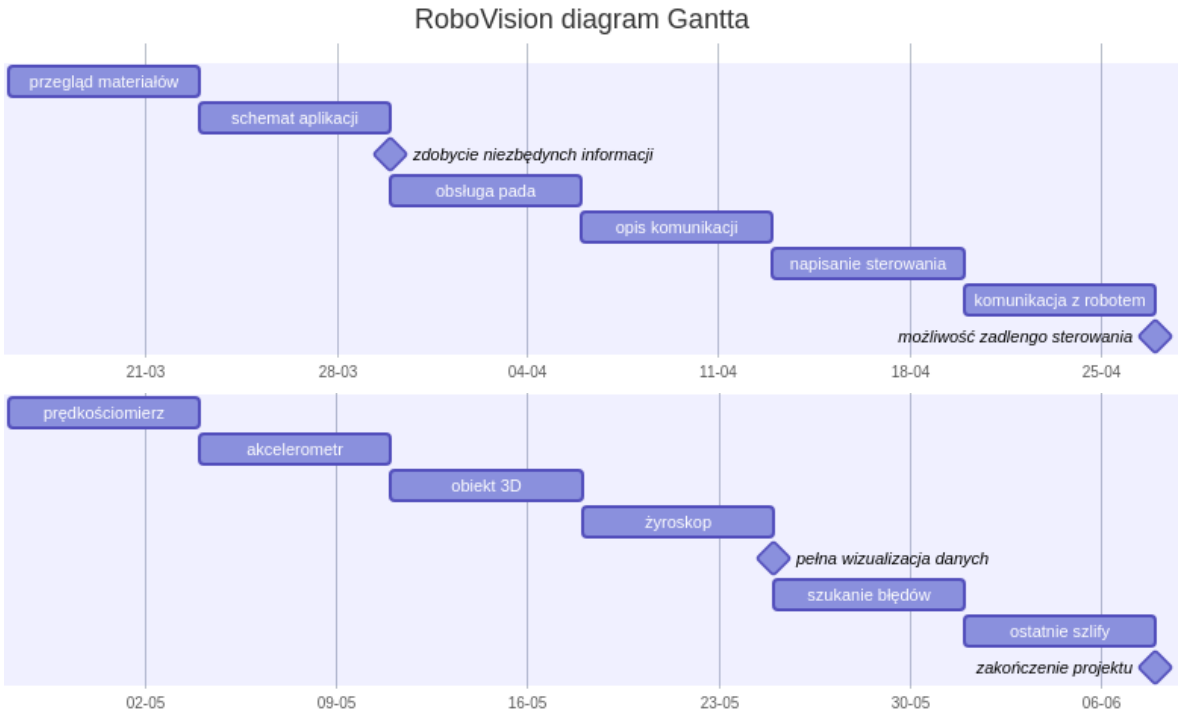
3 Specyfikacja finalnego produktu

Aplikacja będzie w stanie wizualizować dane odbierane z czujników robota. Będą to między innymi:

- wskazania akcelometru,
- wskazania żyroskopu,
- aproksymacja poziomu baterii,
- odległość przeszkody zczytanej z przedniego czujnika ultradzwiękowego,
- prędkość rzeczywista pojazdu z enkoderów.

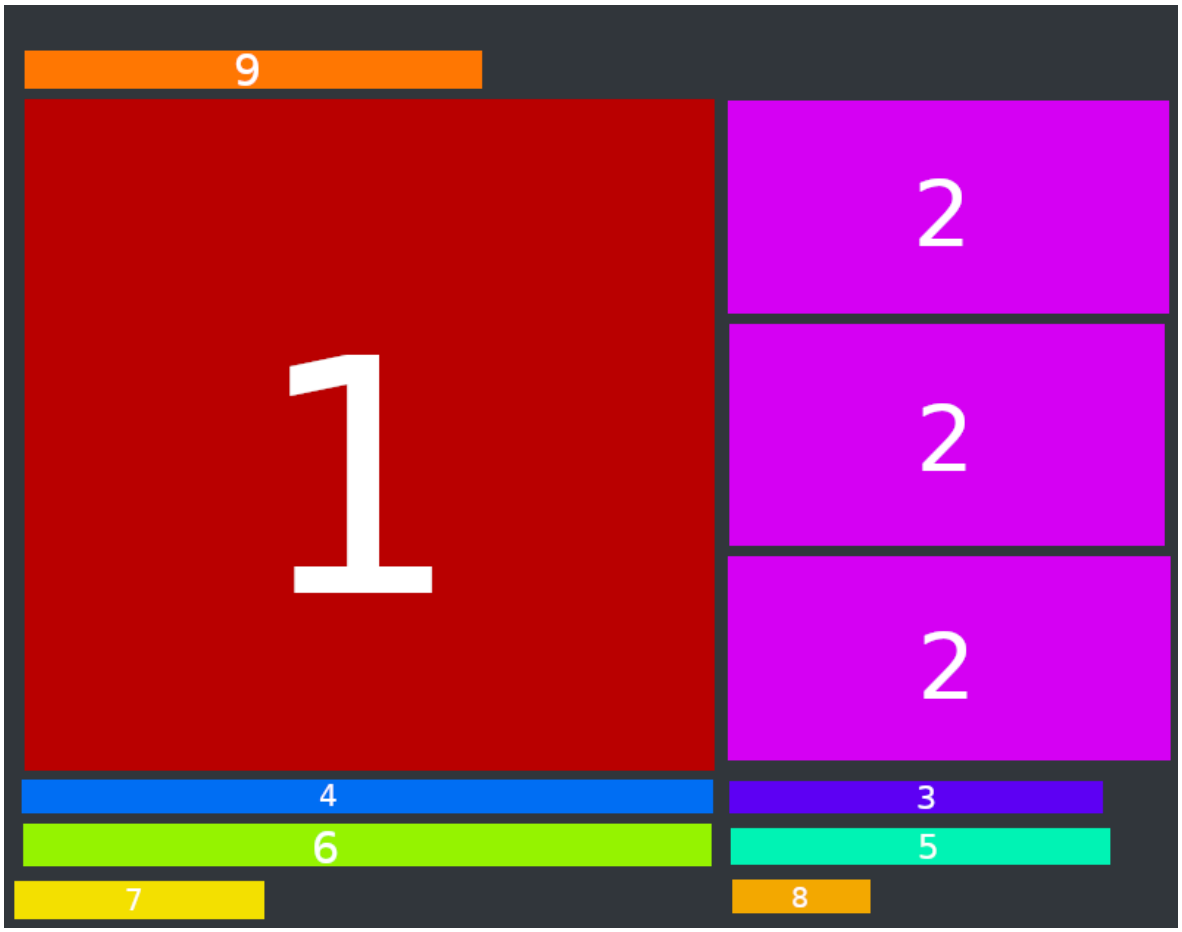
4 Terminarz realizacji poszczególnych podcelów (z dokładnością do 1 tygodnia)

- 22 marca 2020 – zakończenie przeglądu materiałów związanych z danym tematem
- 29 marca 2020 – przygotowanie schematu widoku aplikacji
- 12 kwietnia 2020 – oprogramowanie obsługi joysticka
- 19 kwietnia 2020 – zdefiniowanie protokołu komunikacji i budowy przesyłanych ramek
- 26 kwietnia 2020 – przygotowanie logiki sterownia
- 4 maja 2020 – implementacja dwustronnej komunikacji z robotem
- 10 maja 2020 – wizualizacja wskazań prędkości i naładowania baterii
- 17 maja 2020 – wizualizacja wskazań akcelometru
- 24 maja 2020 – przygotowanie wizualizacji obiektu 3D
- 31 maja 2020 – implementacja obracania obiektu 3D przy użyciu żyroskopu
- 7 czerwca 2020 – szukanie błędów i testowanie wszystkich funkcji
- 14 czerwca 2020 – ostateczne testy działania aplikacji



Rysunek 1: Diagram Gantta

5 Projekt interfejsu graficznego



Rysunek 2: Szablon interfejsu graficznego

Największy wycinek okna przeznaczony jest na prezentowanie modelu robota w trójwymiarze (1). Obiekt ten będzie obracał się zgodnie z wskazaniami akcelometru zamontowanego na realnym pojeździe. Będzie więc to wizualizacja ustawienia robota w przestrzeni.

Po prawej stonie widniejąc trzy wykresy prezentujące pomiary akcelometru w 3 osiach (2). Poniżej znajdują się kolejno wskaźniki opóźnienia komunikacji (3), prędkości liniowej pojazdu (4) i odległości od przeszkody (5), a także poziom naładowania baterii (6).

Na dolnej belce umieszczona jest informacja o podłączonym kontrolerze (7), i słownym statusie komunikacji z robotem (8).

Na szczycie aplikacji znajduje się belka narzędziowa (9), która zawiera opcję nawiązania/zerwania połączenia, wyjście z programu i informację o autorze aplikacji. Po wybraniu funkcji połączenia z robotem, wyświetli się dodatkowe okienko z możliwością wprowadzenia adresu sterowanego obiektu i przycisk umożliwiający inicjację połączenia.

6 Komunikacja z robotem

Połączenie z robotem odbywa się poprzez sieć WiFi. W pierwszej kolejności nawiązywane jest połączenie przy użyciu protokołu TCP. Jeśli się ono powiedzie to uruchamiana jest dodatkowa transmicja z wykorzystaniem UDP. Dzięki takiej koncepcji mamy dwa niezależne kanały komunikacji. Pierwszy służy do przesyłania danych które mają niski piorytet czasowy, potrzebują potwierdzenia odebrania i ewentualnej retransmisji danych. Druga droga komunikacji powstała, aby przesyłać ciągły strumień nowych danych. Zależy nam na jaknajniższym opóźnieniu, a ewentualny błąd trasmisji nie jest krytyczny, bo inforamcje te szybko się przedawniają i są zastępowane przez nowe, świeższe.

Każda ramka zaczyna się od wybranej dużej litery alfabetu, określającej rodzaj przesyłanych danych. Dla protokołu TCP są to:

- P - ping,
- D - dystans przeszkody,
- B - bateria,
- S - realna predkość.

Natomiast dla protokołu UDP wyróżniamy:

- E - moc silników,
- A - akcelometr,
- G - żyroskop.

Wszystkie paczki danych zakończone są średnikiem, przed którym znajduje się ośmiobitowy cykliczny kod nadmiarowy. Niestety ze względu na różnorodność transmitownych informacji, w tym miejscu kończą się cechy wspólne poszczególnych ramek.

6.1 Ping

Jest to najprostrza z obecych tu ramek. Nie przenosi żadnych informacji. Oznacza jedynie koniecność odesłania do nadawcy identyczniej wiadomości, aby można było wyznaczyć chwilę czasowe, niezbędne do obliczenia opóźnienia transmisji.

Forma ramki to: $P\#$;

Gdzie # oznacza CRC.

6.2 Dystans przeszkody

Przesyła informacje z robota o odległości odczytanej z czujnika ultradzwiękowego.

Forma ramki to: $D < uint8\_t > \#$;

Gdzie # oznacza CRC. Przed nim znajduje się wartość odległości wyrażonej w centymetrach, w zakresie 0-100cm.

6.3 Bateria

Przesyła informacje z robota o poziomie baterii.
Forma ramki to: $B < uint8\_t > \#$;

Gdzie # oznacza CRC. Przed nim znajduje się poziom baterii wyrażonej w procentach, w zakresie 0-100%.

6.4 Prędkość

Przesyła informacje z robota o prędkości na kołach.
Forma ramki to: $S < uint8\_t >< uint8\_t > \#$;

Gdzie # oznacza CRC. Przed nim znajdujdują się dwie wartości oddzielone spacją odnoszące się do prędkości poszczególnych kół wyrażonej w metrach na sekunde.

6.5 Moc silników

Przesyła informacje z aplikacji do robota o zadanej mocy silników.
Forma ramki to: $E < uint8\_t >< uint8\_t > \#$;

Gdzie # oznacza CRC. Przed nim znajdujdują się dwie wartości oddzielone spacją odnoszące się do zadanej mocy poszczególnych kół wyrażonej w procentach. Zakresy tych wartości musza mieścić się od 0 do 100%

6.6 Akcelometr

Przesyła informacje z robota o aktualnych wskazaniach akcelometru.
Forma ramki to: $A < uint8\_t >< uint8\_t >< uint8\_t > \#$;

Gdzie # oznacza CRC. Przed nim znajdujdują się trzy wartości oddzielone spacją odnoszące się do aktualnych wskazań akcelometru.

6.7 Żyroskop

Przesyła informacje z robota o aktualnych wskazaniach żyroskopu.
Forma ramki to: $G < uint8\_t >< uint8\_t >< uint8\_t > \#$;

Gdzie # oznacza CRC. Przed nim znajdujdują się trzy wartości oddzielone spacją odnoszące się do aktualnych wskazań żyroskopu.

7 Wykres akcelometru

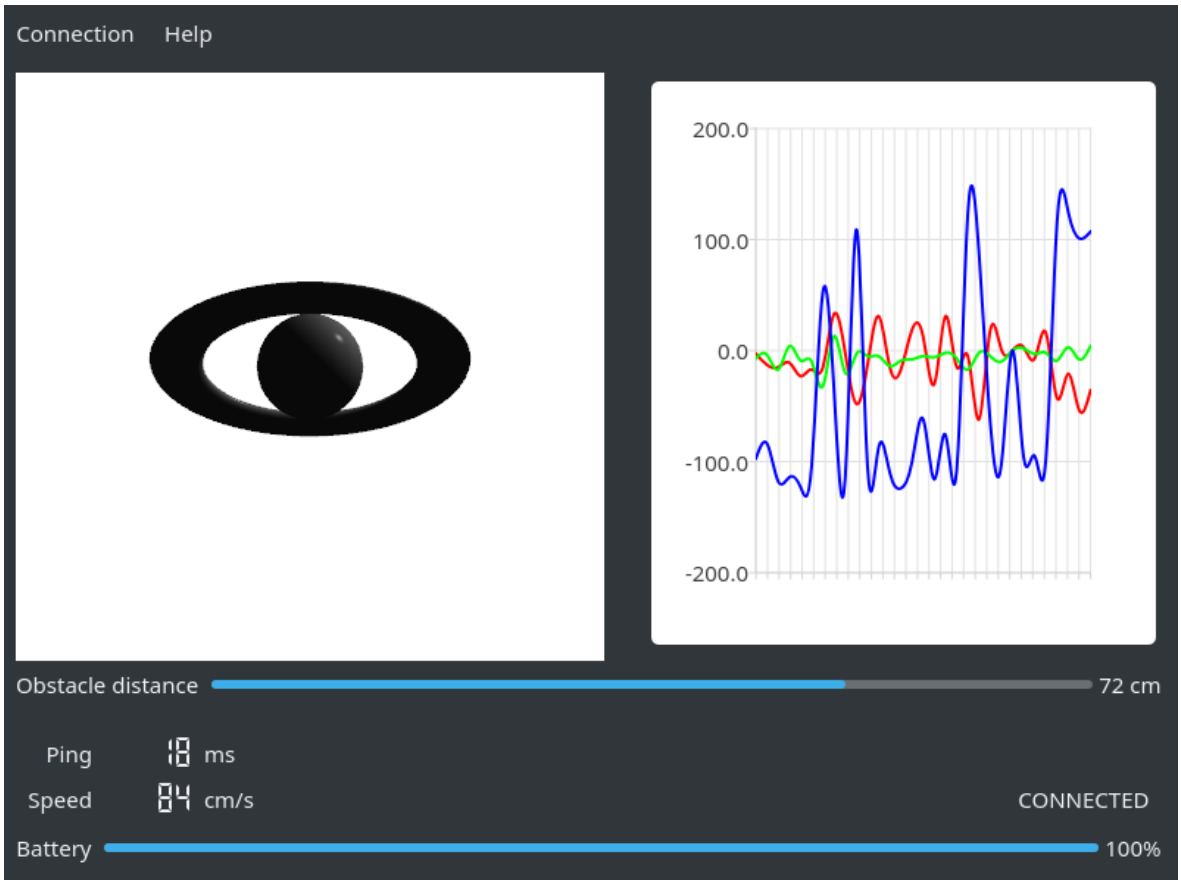
Jednym z założeń projektu było umieszczenie wykresów (patrz rys. 7) przyśpieszeń wszystkich trzech osi sterowanego pojazdu. Została podjęta decyzja scalenia tych wskazań do jednego dużego wykresu w celu zwiększenia czytelności. Przełożyło się to także na znaczną poprawę wydajności aplikacji. W celu dalszej optymalizacji zaimplementowałem algorytm, który pozbywa się wskazań wychodzących poza okno aplikacji.

Wszystkie te działania sprawiły że dodanie tej funkcionalności nie wpłynęło znacząco na ogólną złożoność obliczeniową aplikacji.

Wykres wyskalowany został w przedziale od -2G do +2G. Każda z odczytywanych osi ma swój własny kolor. W przypadku osi Z jest to kolor niebieski. Dla osi X wybrany został kolor zielony, natomiast osi Y przypadł kolor czerwony.

8 Prototyp aplikacji

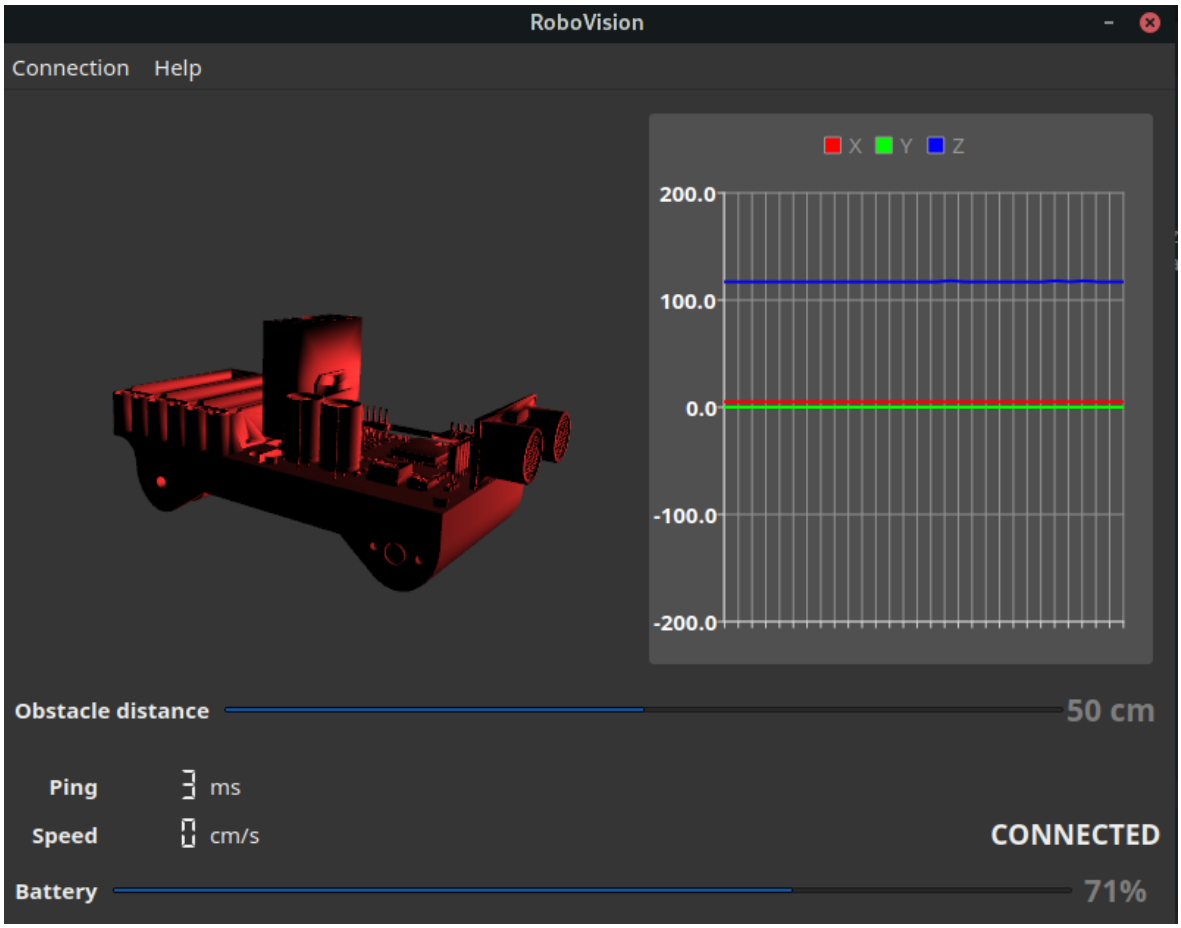
Obrazek (patrz rys. 3) pokazujący zaawansowany etap prac nad aplikacjom.



Rysunek 3: Późny prototyp aplikacji

9 Ostateczne rezultaty

Wszystkie zaplanowane kamienie milowe zostały osiągnięte. Aplikacja (patrz rys. 4) może poszczycić się działającą obsługa joysticka, gotową komunikacją z robotem, wykresem przeciążeń, algorytmem różnicowego układu sterowania oraz działającym wskaźnikiem opóźnienia, prędkości liniowej i poziomu baterii. Została również zaimplementowana wizualizacja obiektu 3D.



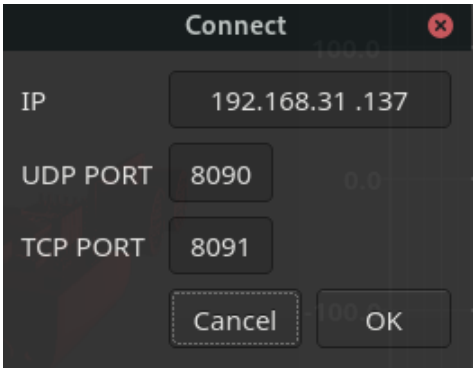
Rysunek 4: Gotowa aplikacja

9.1 Okienko inicializacji połączenia

Aby uprościć interfejs urzytkownika, zdecydowałem się umieścić wszystkie aspekty związane z połączeniem w osobnym, specjalnie zaprojektowanym do tego celu okienku (patrz rys. 5). Wywoływane jest ono poprzez menu narzędziowe w zakładce "Connection". Znajdują się w nim sformatowane pola tekstowe, w które należy wpisać adres urządzenia docelowego oraz porty serwera UDP i TCP.

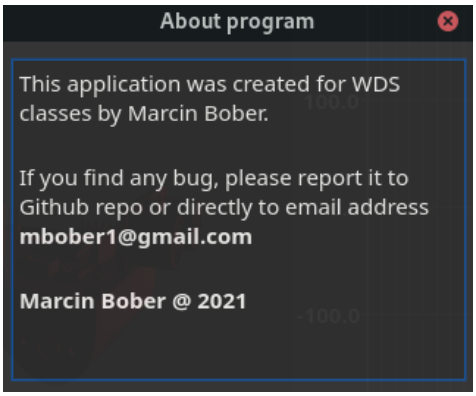
9.2 Okienko informacyjne

Okienko (patrz rys. 6) to można znaleźć w belce narzędziowej w zakładce "Help". Jego celem jest informowanie urzytkownika o autorze aplikacji i jej przeznaczeniu. Dodatkowo informuje ona, gdzie można zgłaszać napotkane błędy. Podobnie jak okienko z



Rysunek 5: Okienko konfiguracji połączenia

konfiguracją połączenia, wykorzustuje ono półprzezroczystość, co pozytywnie wpływa na ogólną estetykę aplikacji. W tej samej zakładce znaleźć można także odnośnik do repozytorium projektu.



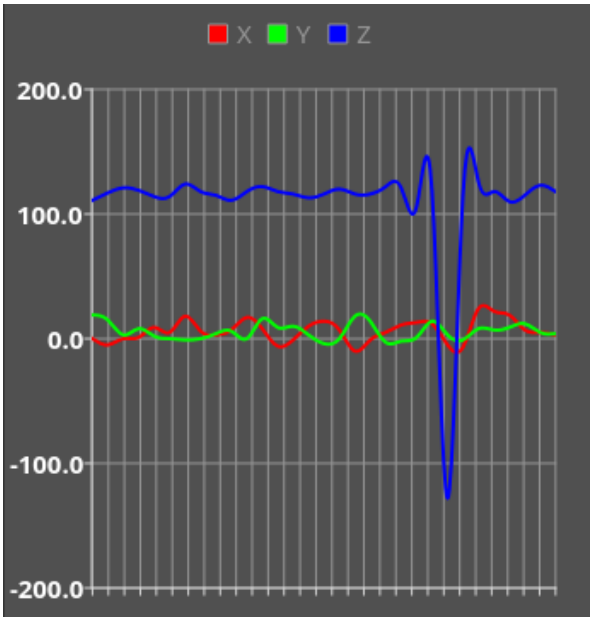
Rysunek 6: Okienko konfiguracji połączenia

9.3 Wykresy przeciążeń

Jednym z najefektywniejszych wizualnie elementów jest wykres przeciążeń (patrz rys.7). Jego zadaniem jest przedstwianie wskazań pobranych z akcelometru robota. Pozwala nam to w bardzo przejrzysty sposób dowiedzieć się jakie siły działają na nasz pojazd. Został on wykonany przy użyciu modułu QTChart. Aby zmniejszyć jego wpływ na wydajność aplikacji wszystkie punkty, które nie mieszczą się na wykresie zostają usunięte. Ma to duży wpływ na użycie pamięci i procesora, w szczególności podczas dłuższej pracy z programem.

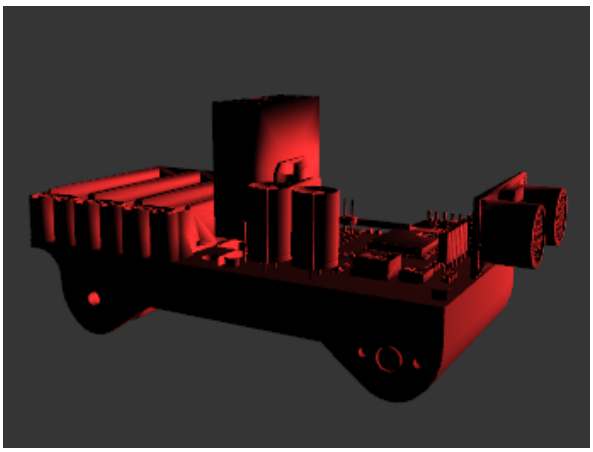
9.4 Wizualizacja 3D

Wizualizacja robota w trówymiarze (patrz rys. 8), jest najbardziej złożonym obliczeniowo elementem całego programu. Jego celem jest przekazanie urzytkownikowi jaknajdokładniejszego odwzorowania pochylenia pojazdu względem podłoża. Jest to niezastąpione podczas podjazdów na strome zbocza lub przeprawy w trudnym terenie. Na potrzeby tej części aplikacji, został stworzony bardzo dokłady model 3D sterowanego robota. Obraca się on w zależności od danych uzyskanych z żyroskopu robota. Po wielu



Rysunek 7: Wykres przeciążeń

przeprowadzonych testach zdecydowałem się na czerwone światło padające z przodu kamery, poniewa takie ustawienia dają najlepszy efekt wizualny.

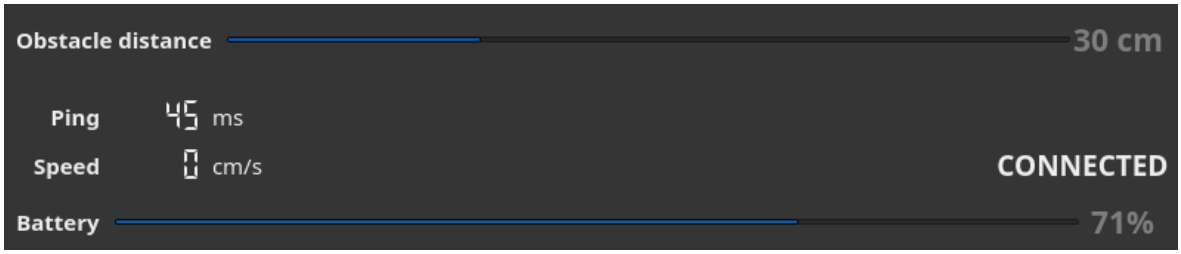


Rysunek 8: Wizualizacja 3D

9.5 Wskaźniki

Ostatnim ważnym elementem są wskaźniki (patrz rys. 9). Wyświetlają one dane z różnych czujników zainstalowanych na pojeździe. Pierwszy z nich otrzymuje odległość zmierzoną przez czujnik ultradzwiękowy zamontowany na przodzie. Jest to ważna informacja podczas zbliżana się do ścian lub innych obiektów. Pasek postępu wyskalowany jest do jednego metra, ponieważ powyżej tej wartości nie musimy się obawiać zderzenia.

Z lewej strony znajdują się inforamcje dotyczące obliczonego opóźnienia i prędkości jazdy. W szczególności ta pierwsza jest niezwykle potrzebna, bo sprawia że w prosty sposób dowiadujemy się o zbliżającym się końcu zasięgu. Z prawej strony znajduje się status połączenia z robotem. Natomiast poniżej udostępniona jest nam aproksymacja poziomu naładowania akumulatora.



Rysunek 9: Wizualizacja 3D

10 Eksperymenty

10.1 Próba połączenia

Etap konfiguracji i łączenia się z robotem oraz efekt widoczny w aplikacji. (patrz rys. 10)

10.2 Stanie na przeszkodzie

Stanie jedną gąsienicą na przeszkodzie i efekt widoczny w aplikacji. (patrz rys. 11)

10.3 Zjazd z górki

Zjeżdzanie przodem z obiektu i efekt widoczny w aplikacji. (patrz rys. 12)

10.4 Leżenie bokiem

Leżenie robota na boku i efekt widoczny w aplikacji. (patrz rys. 13)

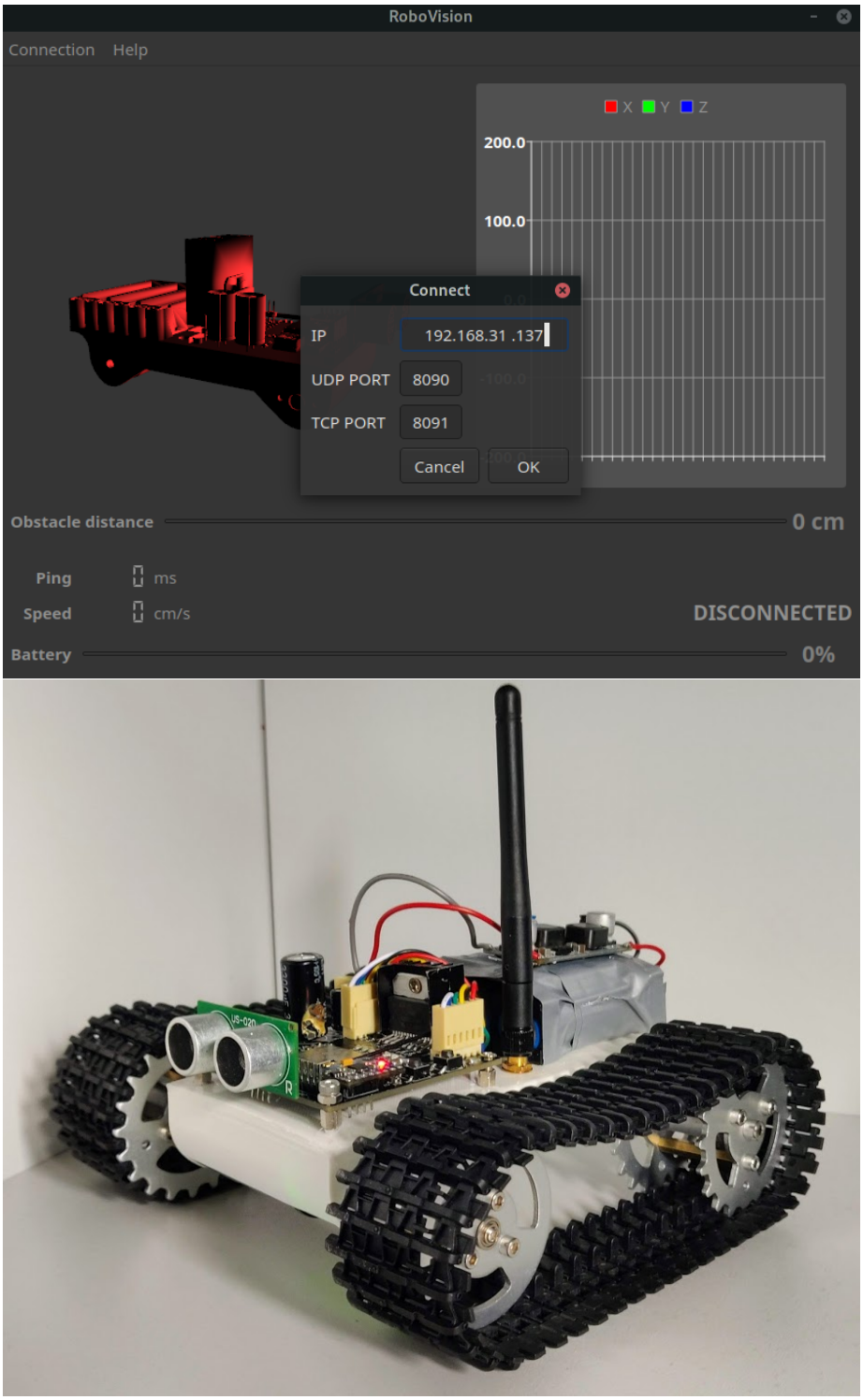
Literatura

[1] Jasmin Blanchette, Mark Summerfield.: C++ GUI Programming with Qt 4.

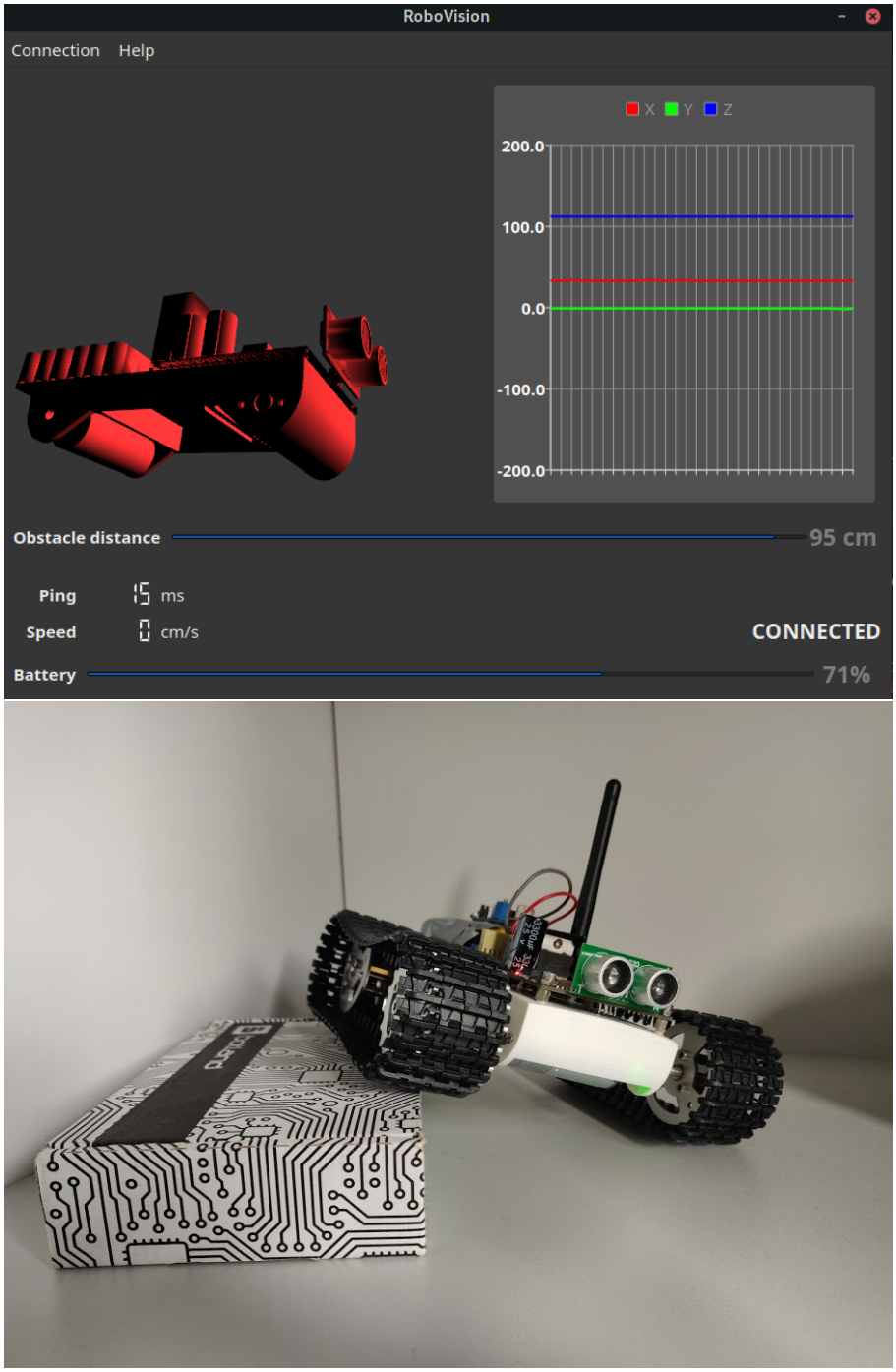
[2] Dokumentacja QT - `https://doc.qt.io/qt-5/`

[3] Calvin Hass - Robot Control - Joystick for Differential Steering.

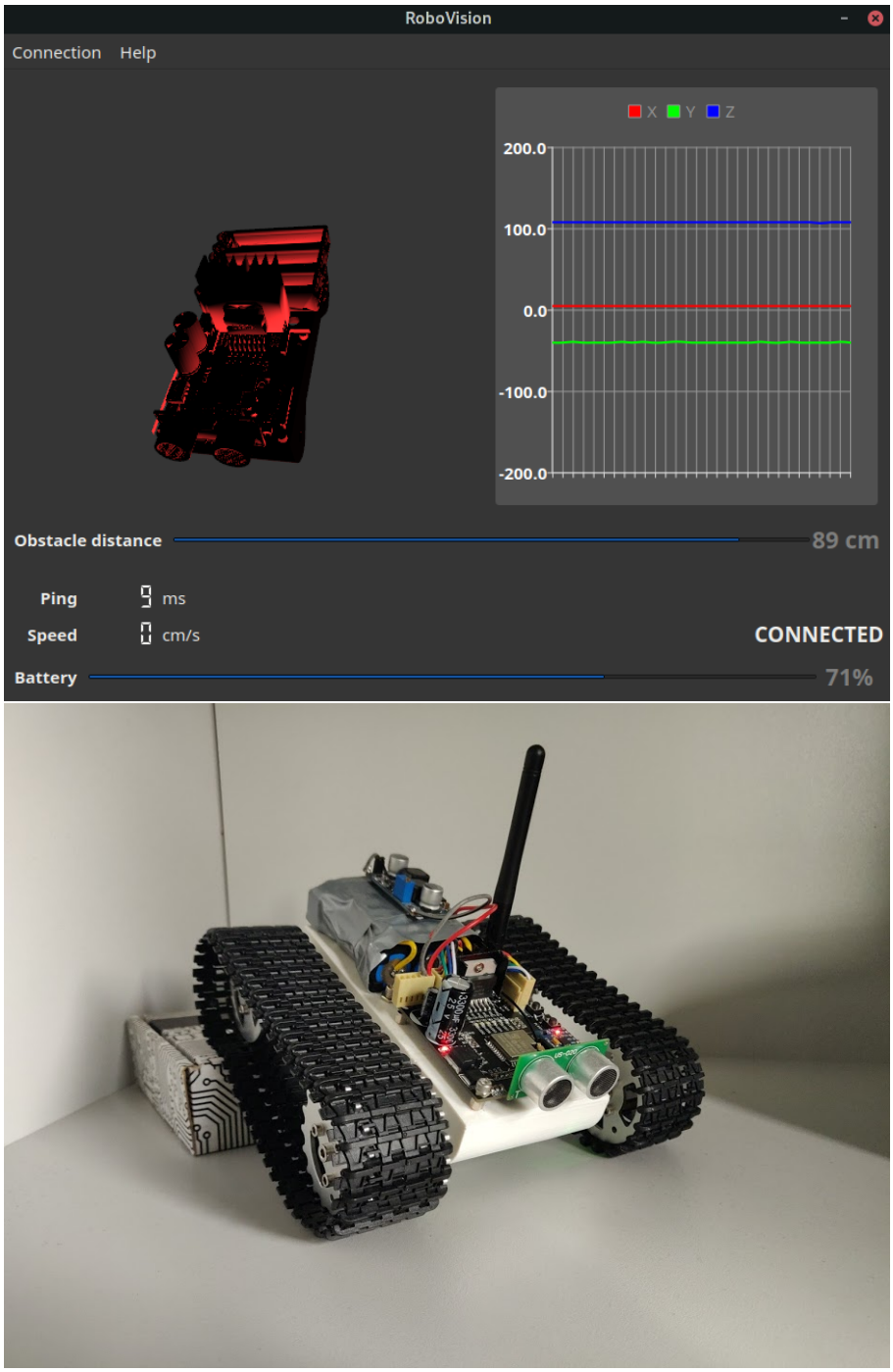
[4] Alan Ezust, Paul Ezust;: C++ i Qt - Wprowadzanie do wzorców projektowych.



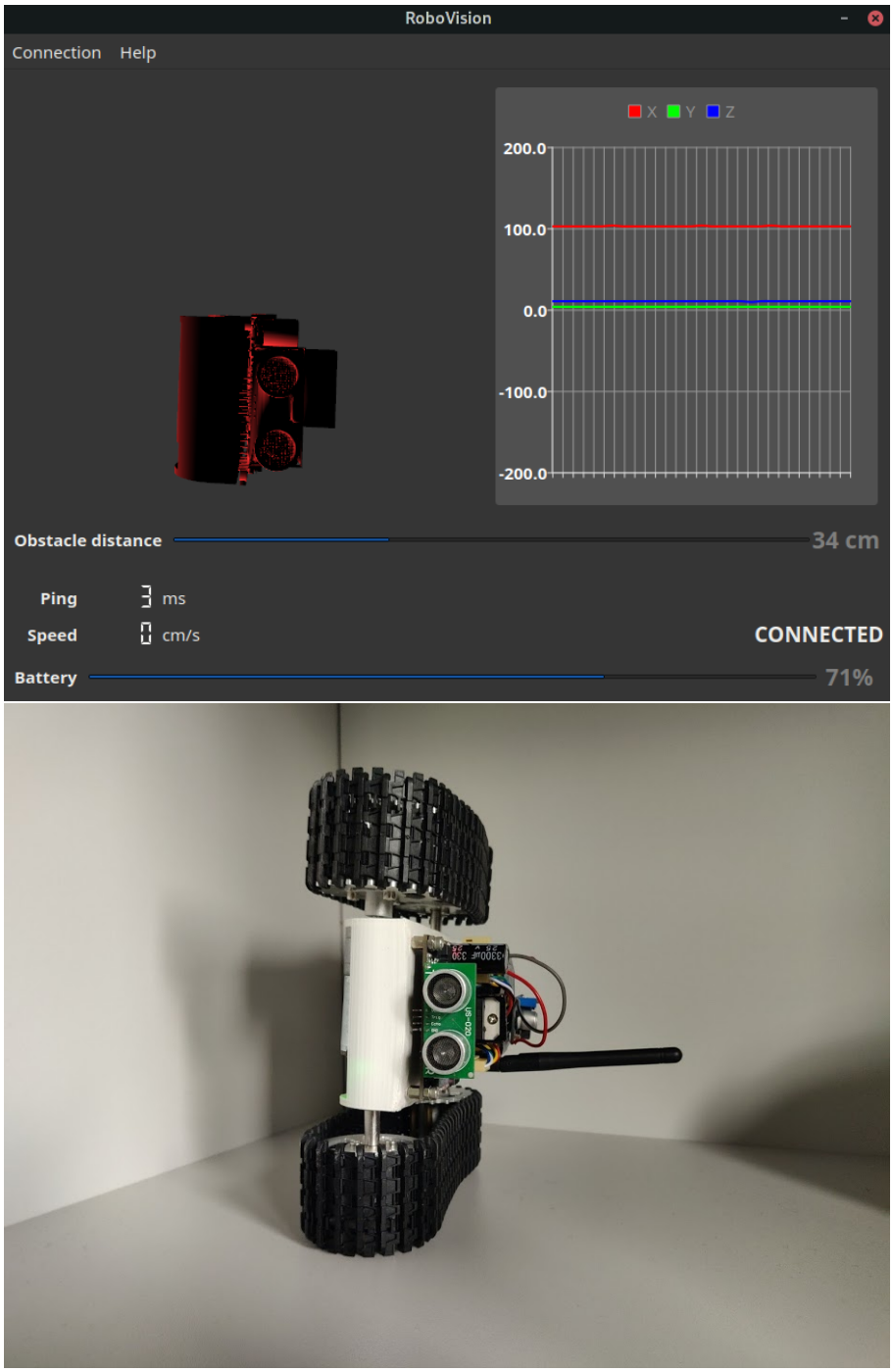
Rysunek 10: Próba połączenia



Rysunek 11: Stanie na przeszkodzie



Rysunek 12: Zjazd z górki



Rysunek 13: Stanie bokiem