Politechnika Wrocławska

Wizualizacja danych sensorycznych

Projekt

Manipulacja wirtualnym modelem płytki uruchomieniowej

Autor:
Michał Antoszko

Prowadzący:
Dr inż. Bogdan Kreczmer

Wrocław, 22 maja 2018

Politechnika
Wrocławska

Spis treści

1 Opis projektu

Celem projektu jest stworzenie wirtualnego, interaktywnego modelu płytki uruchomieniowej. W tym celu zostanie stworzona aplikacja komputerowa, która zwizualizuje ułożenie rzeczywistej płytki uruchomieniowej w przestrzeni. Wykorzystana zostanie do tego płytka *Arduino Uno* w połączeniu z modułem zawierającym akcelerometr i żyroskop. Akcelerometr będzie służył do rejestracji przemieszczenia, natomiast żyroskop będzie wskazywał orientacje obiektu. Wizualizacja zostanie stworzona przy wykorzystaniu biblioteki *OpenGLWidget*, która integruje biblioteki graficzne *qt* ze specyfikacją *API – OpenGL*. Do stworzenia writualnego modelu 3D płytki wykorzystane zostanie oprogramowanie do modelowania grafiki 3D – *Blender*.

2 Założenia i harmonogram

2.1 Moduł wczytywania modelu i jego wyświetlanie

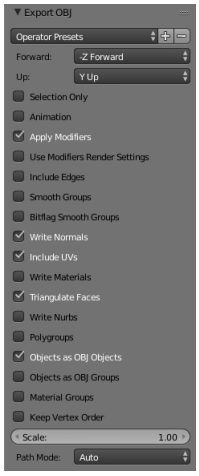
Zakres prac będzie obejmował stworzenie interfejsu do wczytywania modelu 3D w formacie *wavefront (.obj)*. Aby model mógł zostać wczytany, należy zastosować odpowiednią formę eksportowania. Na grafice *1.* przedstawione są opcje, które należy zaznaczyć podczas eksportu modelu. Dodatkowo będzie możliwe wyświetlenie wczytanego modelu (bez tekstur), wraz z możliwością rotacji i przemieszczenia (bez komunikacji z płytką).

2.2 Arduino Uno i moduł z sensorami

Zakres prac będzie obejmował wykonanie schematu połączeń elektrycznych pomiędzy płytką uruchomieniową *Arduino Uno* i modułem z sensorami oraz zapewnienie komunikacji pomiędzy nimi.

2.3 Komunikacja pomiędzy Arduino i PC

Zakres prac będzie obejmował ustalenie ramki danych i zapewnienie komunikacji pomiędzy *Arduino Uno* i komputerem klasy *PC* wykorzystując do tego protokół *UART*. Dane te zostaną wykorzystane do synchronizacji położenia wirtualnego modelu płytki z rzeczywistą płytką *Arduino Uno*.



Rysunek 1: Opcje eksportowania obiektu w programie Blender

2.4 Interfejs graficzny – wykresy

Zakres prac będzie obejmował stworzenie interfejsu graficznego wraz z historią zmian wartości mierzonych przez czujniki. Historia będzie wyświetlana na wykresach, pomiędzy którymi będzie można się przełączać.

2.5 Model 3D płytki

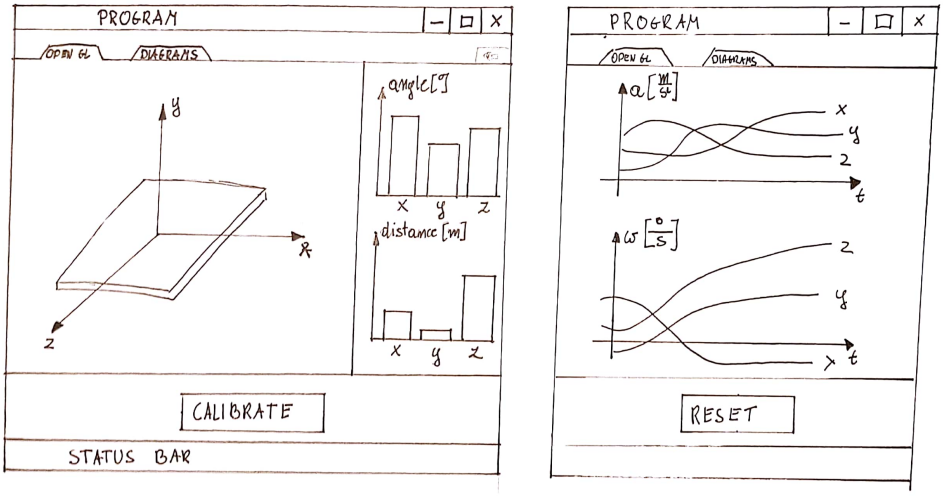
Zakres prac będzie obejmował wykonanie modelu płytki *Arduino Uno*, który będzie zawierał adekwatne do rzeczywistości rozłożenie portów i elementów elektronicznych, natomiast pozbawiony będzie ścieżek i tekstur. Do modelowania zostanie wykorzystane oprogramowanie *Blender*.

2.6 (Opcjonalne) Graficznie rozbudowany model

Zakres prac będzie obejmował dodanie do istniejącego już modelu 3D UV map i adekwatnych do nich tekstur, które zostaną wykonane w oprogramowaniu *Substance Painter*.

2.7 Projekt interfejsu graficznego

Interfejs graficzny programu będzie składał się z jednego okna, w którym znajdą się dwie karty, pomiędzy którymi użytkownik będzie mógł się przełą-



Rysunek 2: Odręczny szkic interfejsu

czać w czasie rzeczywistym. Pierwsza z nich – karta *OpenGL*, będzie zawierać okno z animacją płytki uruchomieniowej. Dodatkowo będzie możliwość wysunięcia menu z wykresami, które będą pokazywały dokładnie te same wartości co animacja, mianowicie kąt obrotu i przebytą odległość od środka, z osobna dla każdej osi. W związku z nieuniknionym dryfem zostanie zaimplementowana opcja ponownej kalibracji urządzenia, którą będzie można aktywować za pomocą przycisku *CALIBRATE*. Druga karta – *DIAGRAMS* będzie przedstawiać historię zmian wartości odczytywanych przez czujniki połączone z Arduino. Dla każdej z osi będzie rysowany wykres przyspieszenia i prędkości kątowej w zależności od czasu. Użytkownik będzie miał możliwość wymazania historii za pomocą przycisku *RESET*. Idea zaprezentowana jest graficznie na rys. 3.

2.8 Wykres Gantta

Rysunek 3: Terminy realizacji poszczególnych zadań

3 Opis rozwiązań technicznych

3.1 Zawartość pliku .obj i jego opis

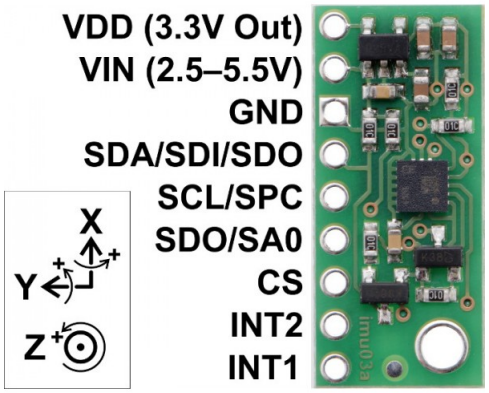
W celu wczytania modelu należy parsować plik .obj, który należy najpierw wyeksportować z pliku .blend korzystając z instrukcji zamieszczonych w punkcie 2.1 dokumentacji. W prawidłowo wyeksportowanym pliku .obj jedyne symbole, które powinny się pojawić to te, które występują w tabeli 1. Eksportowany model z założenia ma być jednym elementem, więc symbol obiektu powinien pojawić się tylko raz. Linijki komentarza i obiektu nie są ważne z technicznego punktu widzenia, dlatego można je pominąć. Wartości następujące po symbolach v, vt i vn to współrzędne odpowiednich elementów modelu, które poszczególne symbole reprezentują. Wartości występujące po symbolu f prezentują się w następujący sposób: a/b/c d/e/f g/h/i. Każda trójka wartości oznacza osobny wierzchołek, który tworzy twarz (ang. face - przestrzeń rozpinana przez wierzchołki) modelu (przeważnie trójkąt). Wartości a, d, g symbolizują indeksy wierzchołków, które tworzą daną twarz. Wartości b, e, h symbolizują, które współrzędne tekstur wybrać, natomiast c, f, i symbolizują, które *normale* należy wykorzystać przy tworzeniu danej twarzy. Warto zwrócić uwagę na to, że indeksy liczone są od 1, a nie od 0, tak jak w języku C++.

3.2 Moduł Pololu LSM6DS33

W projekcie zostało wykorzystane *Arduino Uno Rev3* w połączeniu z modułem zawierającym żyroskop i przyspieszeniomierz, wyprodukowanym przez Pololu o oznaczeniu LSM6DS33. Charakteryzuje się on niewielkimi rozmiarami, niskim poborem energii oraz prostą obsługą, ponieważ producent do-

Symbol	Znaczenie
#	komentarz
o	nazwa obiektu
v	współrzędne wierzchołka
vt	współrzędne tekstur wierzchołka
vn	współrzędne normali wierzchołka
f	indeksy wierzchołków tworzących twarz

Tabela 1: Symbole oznaczeń w pliku .obj

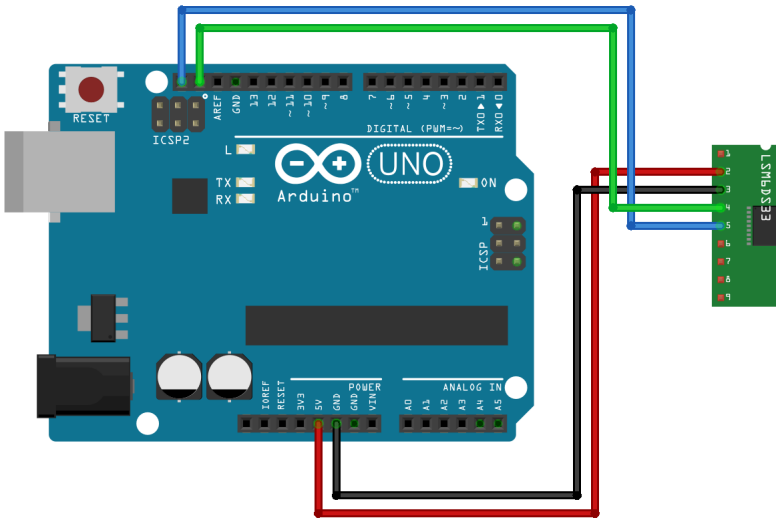


Rysunek 4: Rozłożenie ścieżek w układzie LSM6DS33 oraz rozłożenie osi

starcza bibliotekę kompatybilną z Arduino. Moduł może komunikować się poprzez I^2C lub *SPI*. W projekcie zostanie wykorzystany interfejs I^2faC, ponieważ został on stworzony z myślą o komunikacji pomiędzy dwoma układami scalonymi na niewielkie odległości, dodatkowo atutem jest wykorzystanie jedynie dwóch przewodów do komunikacji z płytką Arduino.

3.3 Port szeregowy

Komunikacja pomiędzy Arduino i komputerem PC odbywa się za pomocą przewodu USB z wykorzystaniem wirtualnego portu szeregowego z *BaudRate* ustawionym na 19200. Częstotliwość wysyłania danych z Arduino do komputera wynosi 50 Hz. Dane wysyłane są w formie tekstowej, przy czym



Rysunek 5: Schemat połączeń pomiędzy Arduino i modułem Pololu z wykorzystaniem I^2C

dla ułatwienia parsowania każdy pomiar oddzielony jest pionową kreską. Do odczytywania danych z portu szeregowego wykorzystana została biblioteka *QSerialPort*, która pozwala na kontrolę i weryfikację danych. Dodatkowo umożliwia synchronizację czasową, dzięki czemu dane odczytywane są tak szybko, jak zostaną wysłane. Nie występują również sytuacje, w których program wyprzedza odczyty z sensorów i się desynchronizuje.

3.4 Kalibracja czujników i konwersja odczytów

Czujniki wykorzystują 16 bitowe przetworniki, które są bardzo czułe na zakłócenia. W związku z powyższym zostały one ustawione tak, aby wykorzystywały jedynie 8 najbardziej znaczących bitów. Pozwala to obniżyć wpływ szumów i wykonać dokładniejsze pomiary. Kolejnym istotnym zadaniem, które należy wykonać przed rozpoczęciem użytkowania czujników jest ich kalibracja. W związku z brakiem dostępu do specjalistycznych narzędzi, kalibracja została przeprowadzona domowym sposobem. Płytka przez cały czas trwania kalibracji była umieszczona nieruchomo na płaskim podłożu, co pozwoliło na półautomatyczne dobranie offsetów. Przez minutę płytka zbierała odczyty z czujników z częstotliwością 1 Hz, co pozwoliło na wyliczenie średnich offsetów. Następnie po uwzględnieniu offsetów należało nieznacznie poprawić

ich wartości manualnie, co dla zastosowań tego projektu dało zadowalający efekt. Warto zwrócić uwagę na to, że dla zastosowań tego projektu wartość przyspieszenia ziemskiego przejawiająca się w pomiarach akcelerometru na osi z jest nieistotna z punktu widzenia układu odniesienia, dlatego została skompensowana. Przeliczanie zmierzonych wartości dyskretnych na wartości rzeczywiste wykonywane jest na komputerze, aby odciążyć płytkę Arduino. Przeliczniki znajdują się na 11. stronie dokumentacji modułu LSM6DS33.

3.5 Wyliczanie kąta obrotu i przebytej odległości

Wartości otrzymane z czujników – przyspieszenie liniowe oraz prędkość kątową należy scałkować odpowiednią ilość razy, aby otrzymać szukane wartości. W celu otrzymania kąta obrotu należy scałkować prędkość kątową dla każdej osi z osobna, co daje wzór:

$$\alpha = \omega \cdot \Delta t + c, gdzie$$

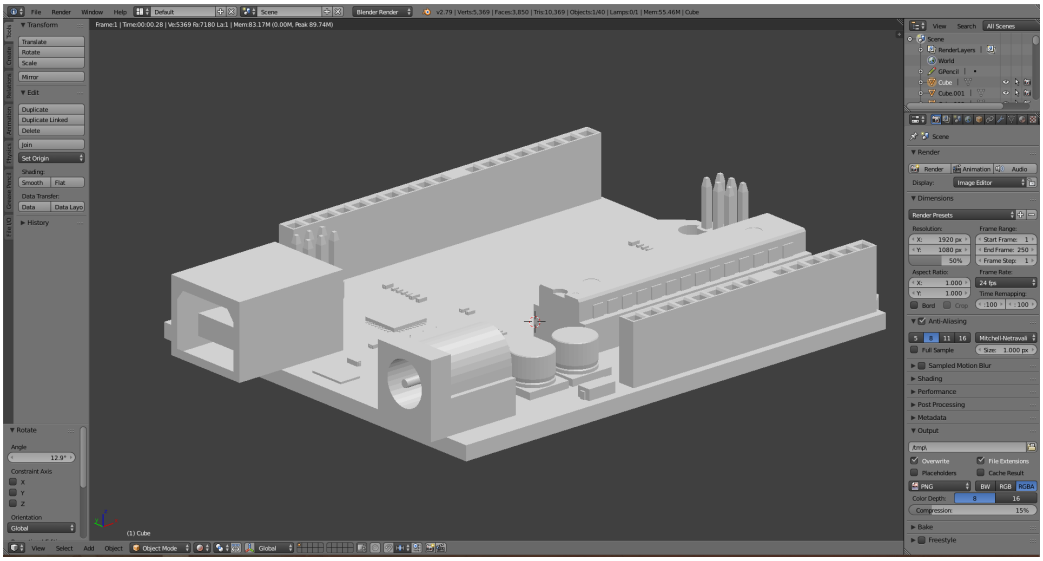
ω to prędkość kątowa, Δt to czas, w którym został wykonany pomiar, a c jest stałą całkowania. Natomiast jeżeli chcemy uzyskać przebytą drogę to należy scałkować podwójnie przyspieszenie liniowe. Otrzymujemy wtedy wzór:

$$x = (a \cdot \Delta t + c_1) \cdot \Delta t + c_2 = a \cdot \Delta t^2 + c_1 \cdot \Delta t + c_2$$

Wartość kąta obrotu jest dokładniejsza, ponieważ występuje jedynie całkowanie pojedyncze. W związku z występującym całkowaniem spotyka się problem dryfu, który powiązany jest ze stałą całkowania. Przez istniejący dryf, nawet, gdy płytka jest nieruchoma występują niewielkie szumy, w wyniku których otrzymujemy niezerowe wartości kąta obrotu i przebytej odległości. Aby zaradzić temu problemowi zostały wprowadzone niezależne od siebie progi, które mają pomijać wartości, które powstają z całkowania szumów. W celu zredukowania wpływu szumów zostały wprowadzone progi, od których wartości kąta i przemieszczenia są brane pod uwagę. Dla kąta jest to $\pm 0.1°$, natomiast dla odległości ± 0.1 cm.

3.6 Model 3D wykonany w Blenderze

Rezultat końcowy w postaci modelu 3D (możliwie szczegółowego) powstał z połączenia elementów pierwotnych. Model zawiera wszystkie elementy wypukłe obiektu rzeczywistego o powierzchni większej niż 2 mm^2. Elementy



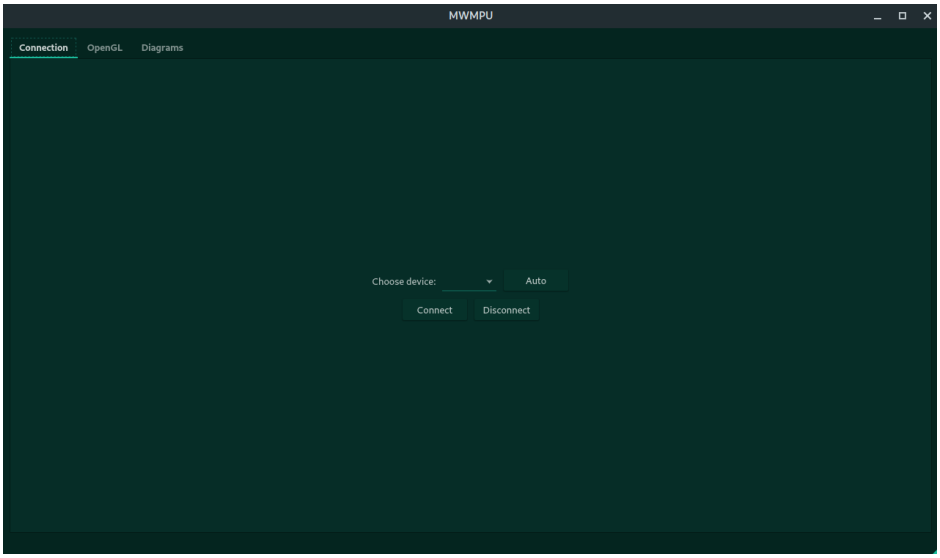
Rysunek 6: Trójwymiarowy model płytki Arduino wykonany w Blenderze

umieszczone są w modelu z dokładnością do 1 *mm* względem obiektu rzeczywistego. W tym celu wykorzystano rysunki techniczne ze strony producenta jak i pomiary wykonane własnoręcznie za pomocą miarki z podziałką dokładności 1 *mm*. W celu usprawnienia pracy nad modelem wykorzystane zostały *modifiery bool* i *array*. Pierwszy z nich pozwala wykonywać proste działania sumy i różnicy logicznej na dwóch zbiorach (grupach obiektów), dzięki czemu możliwe było wykonanie otworów w portach jak i w wejściach *GPIO*. Drugi *modifier* pozwala zwielokrotnić wykonany obiekt z dowolnym offsetem, dzięki czemu wystarczyło wykonać jeden port *GPIO*, a następnie zwielokrotnić go, aby powstała cała kolumna. Na całym modelu został wykonany *UV unwrapping*, dzięki któremu będzie można utworzyć tekstury dla obiektu.

3.7 Tekstury wykonane w Substance Painter

Do modelu 3D płytki zostały wykonane tekstury, które mają za zadanie oddać, jak w rzeczywistości wygląda płytka. Podzielone są one na tekstury:

- podstawowego koloru,
- metaliczności elementów,
- twardości/połysku,



Rysunek 7: Pierwsza karta interfejsu użytkownika

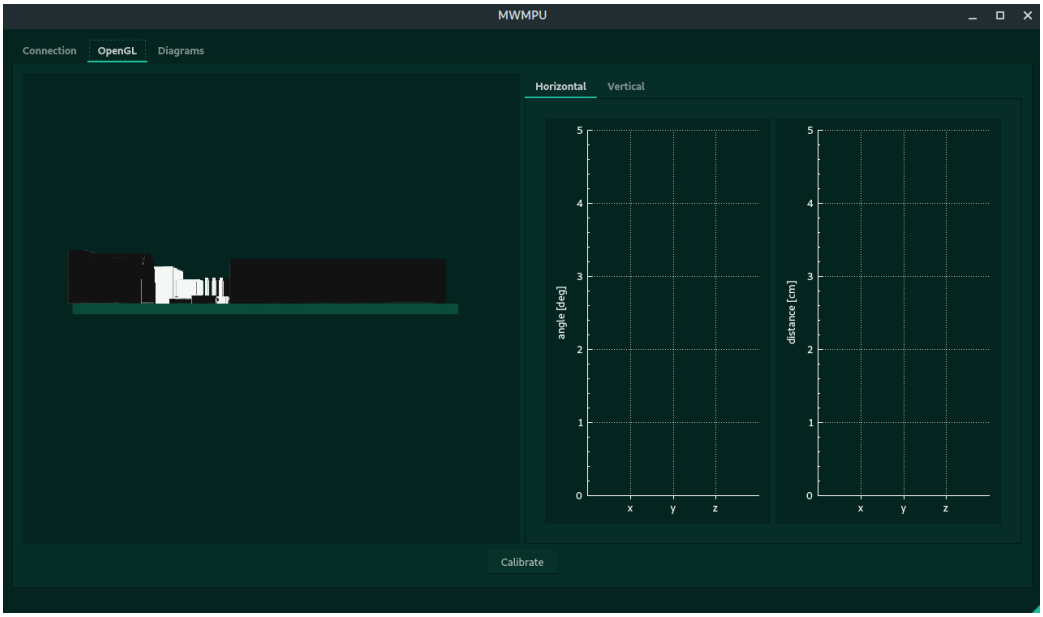
- wypukłości i wklęsłości elementów.

3.8 Ostateczna forma interfejsu graficznego

Interfejs graficzny programu składa się z 3 kart:

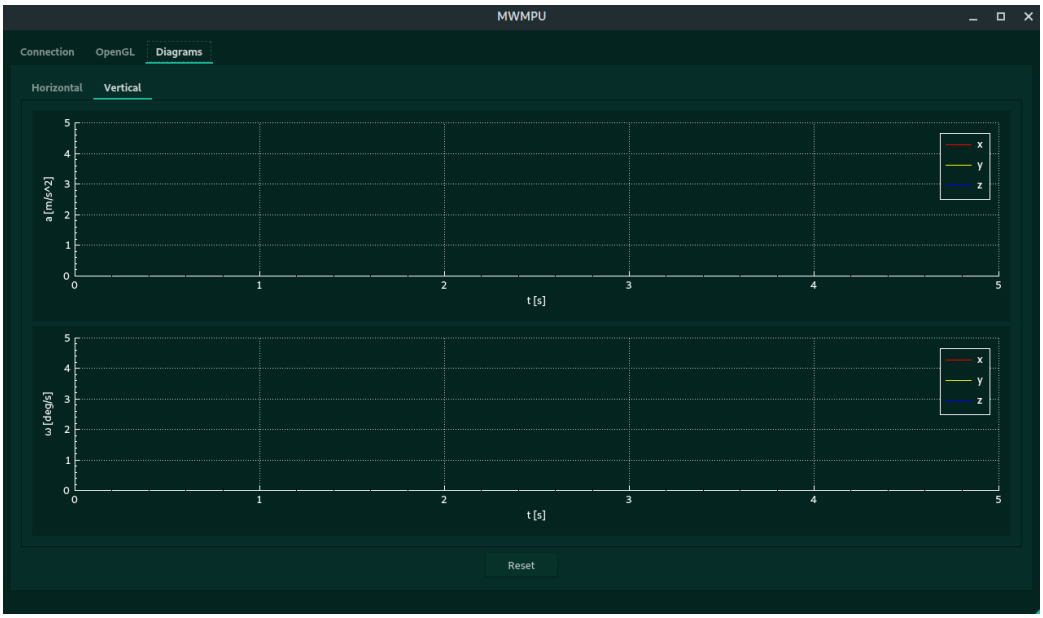
- Connection,
- OpenGL,
- Diagrams.

Karta *Connection* odpowiada za ustanowienie połączenia z portem szeregowym. Pozwala na wybranie urządzenia z listy i połączenie/rozłączenie się z nim. Przycisk *Auto* pozwala na automatyczne wybranie odpowiedniego urządzenia. Komunikaty związane z podejmowanymi czynnościami wyświetlane są na dole ekranu na pasku statusu. Następna karta – OpenGL zawiera okno z trójwymiarowym aktywnym renderem płytki Arduino oraz wykresy kątów obrotu i odległości od środka układu współrzędnych. Istnieje możliwość przełączania się pomiędzy dwoma ułożeniami wykresów – horyzontalnym i wertykalnym. Ostatnia karta przedstawia wykresy danych pozyskanych czysto z czujników, mianowicie wykresy przyspieszenia liniowego i prędkości kątowej.



Rysunek 8: Druga karta interfejsu użytkownika

Jak w poprzednim przypadku istnieje również możliwość zmiany ułożenia wykresów.



Rysunek 9: Trzecia karta interfejsu użytkownika