

PROJEKT

WIZUALIZACJA DANYCH SENSORYCZNYCH

Założenia projektowe

Wizualizacja samopozycjonującej się platformy fotowoltaicznej

Albert LIS, 235534

Termin: Śr 17:05

Prowadzący:
dr inż. Bogdan KRECZMER

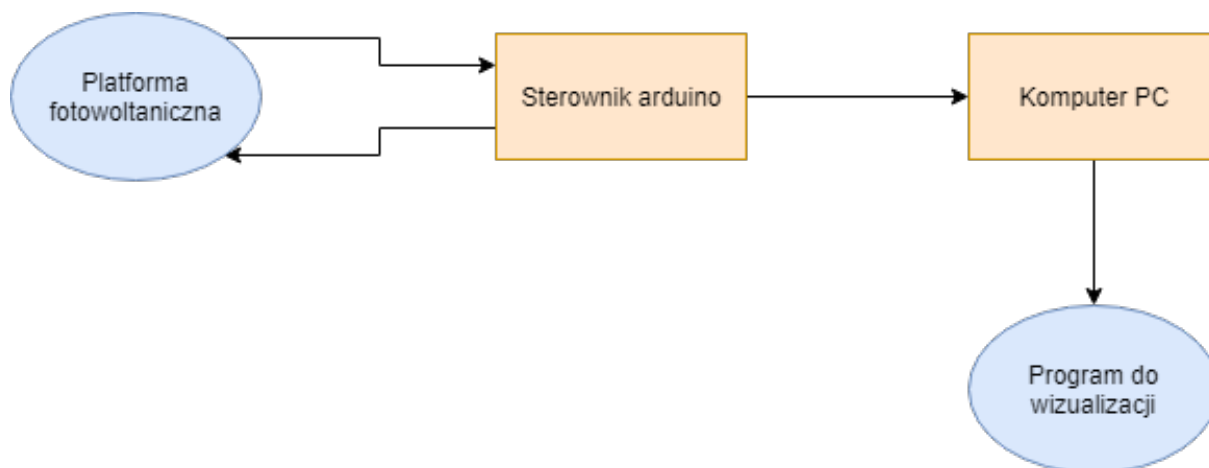
2 czerwca 2019

Spis treści

1	Opis projektu	2
2	Założenia projektowe	2
2.1	Komunikacja	2
2.2	Wizualizacja	2
3	Harmonogram pracy	3
3.1	Zakres prac	3
3.2	Kamienie milowe	3
3.3	Wykres Gantta	4
4	Projekt interfejsu graficznego	5
4.1	Funkcjonalność UI	5
4.2	Funkcjonalność aplikacji	5
4.3	Graficzna reprezentacja aplikacji	5
5	Wstępne rezultaty	7
5.1	Zmiany w projekcie	7
5.2	Zrealizowane zadania	7
6	Zaawansowane rezultaty	8
6.1	Komunikacja	8
6.2	Graficzny interfejs użytkownika	8
7	Rezultaty prawie końcowe	10
7.1	Zrealizowane zadania	10
7.2	Zmiany w projekcie	11

1 Opis projektu

Celem projektu jest stworzenie wizualizacji 3D platformy fotowoltanicznej. Platforma jest sterowana za pomocą mikrokontrolera i czterech czujników. Dzięki temu ma możliwość podążania za najintensywniejszym źródłem światła i pozycjonowania się w sposób umożliwiający optymalne korzystanie z energii słonecznej. Dane o pozycji platformy zostaną przesłane do komputera PC. W komputerze zostanie uruchomiona aplikacja pozwalająca pokazywać aktualną pozycję platformy.



Rysunek 1: Architektura systemu

2 Założenia projektowe

2.1 Komunikacja

1. Połączenie ze sterownikiem
Realizowane za pomocą modułu Wi-Fi ESP8266 i protokołu UDP/TCP lub bez łączności bezprzewodowej z użyciem portu szeregowego.
2. Połączenie modułu Wi-Fi z mikrokontrolerem
Realizowane za pomocą portu szeregowego.

2.2 Wizualizacja

1. Środowisko
Zostanie wykorzystany silnik graficzny UNITY w darmowej wersji.
2. Modele
Zostaną wygenerowane za pomocą programu Blender.
3. Tekstury
Zostaną stworzone za pomocą programu GIMP lub pobrane z dowolnej internetowej bazy z darmowymi teksturami.

3 Harmonogram pracy

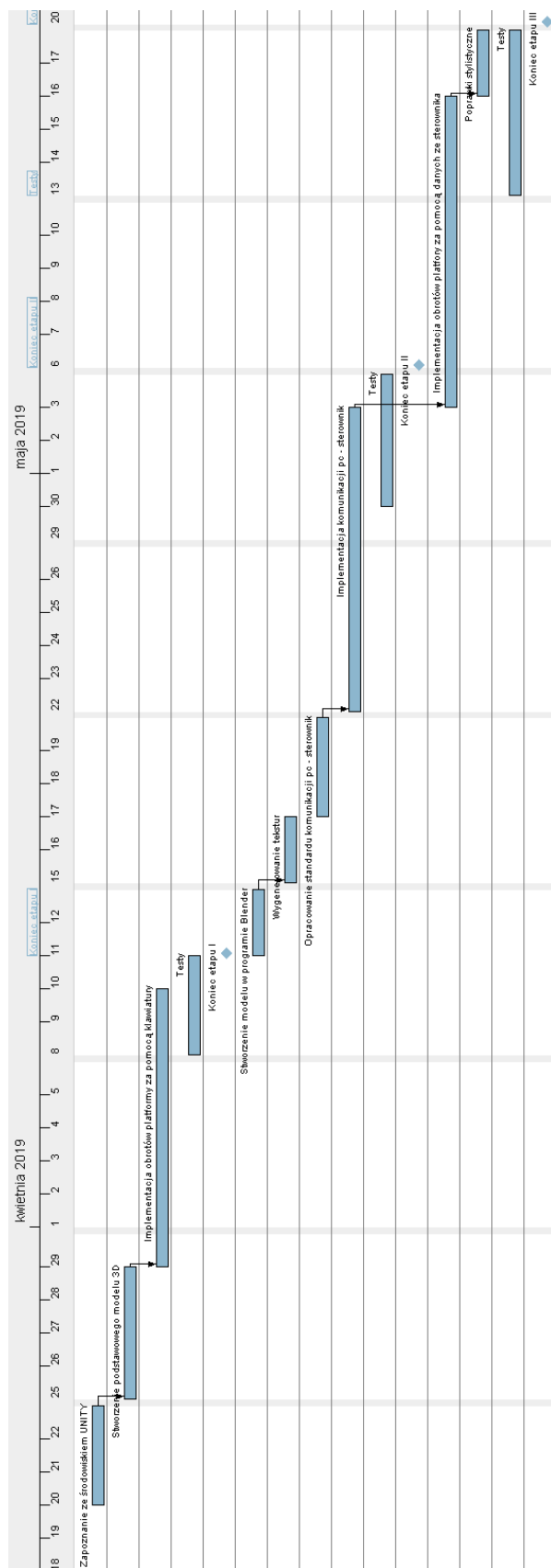
3.1 Zakres prac

1. Zapoznanie się ze środowiskiem UNITY
Stworzenie kilku prostych projektów tak aby zapoznać się ze środowiskiem i jego możliwościami.
2. Stworzenie podstawowego modelu 3D
Stworzenie prostego modelu platformy bez dbałości o detale.
3. Implementacja obrotów platformy za pomocą klawiatury
Stworzenie wizualizacji poruszania się modelu za pomocą strzałek na klawiaturze.
4. Stworzenie dokładnych modeli w programie Blender
Stworzenie dokładnego odwzorowania platformy z uwzględnieniem połączeń krawędzi.
5. Wygenerowanie tekstur
Stworzenie lub pobranie z internetu tekstur dla obiektów.
6. Opracowanie standardu komunikacji sterownik - PC
Zastanowienie się nad sposobem przesyłania informacji oraz ich kodowaniem.
7. Implementacja komunikacji sterownik - PC
Implementacja jednostronnej komunikacji między sterownikiem a PC.
8. Implementacja obrotów platformy za pomocą danych ze sterownika
Modyfikacja istniejącego sterowania w taki sposób aby zwizualizowany stan platformy zgadzał się z rzeczywistym.
9. Poprawki stylistyczne
Poprawa elementów które okazały się niedopracowane w trakcie projektu.

3.2 Kamienie milowe

1. Implementacja działającej wizualizacji w oparciu o sterowanie klawiaturą
2. Implementacja poprawnej komunikacji sterownik - PC
3. Implementacja wizualizacji w oparciu o dane ze sterownika

3.3 Wykres Gantta



Rysunek 2: Diagram Gantta

4 Projekt interfejsu graficznego

4.1 Funkcjonalność UI

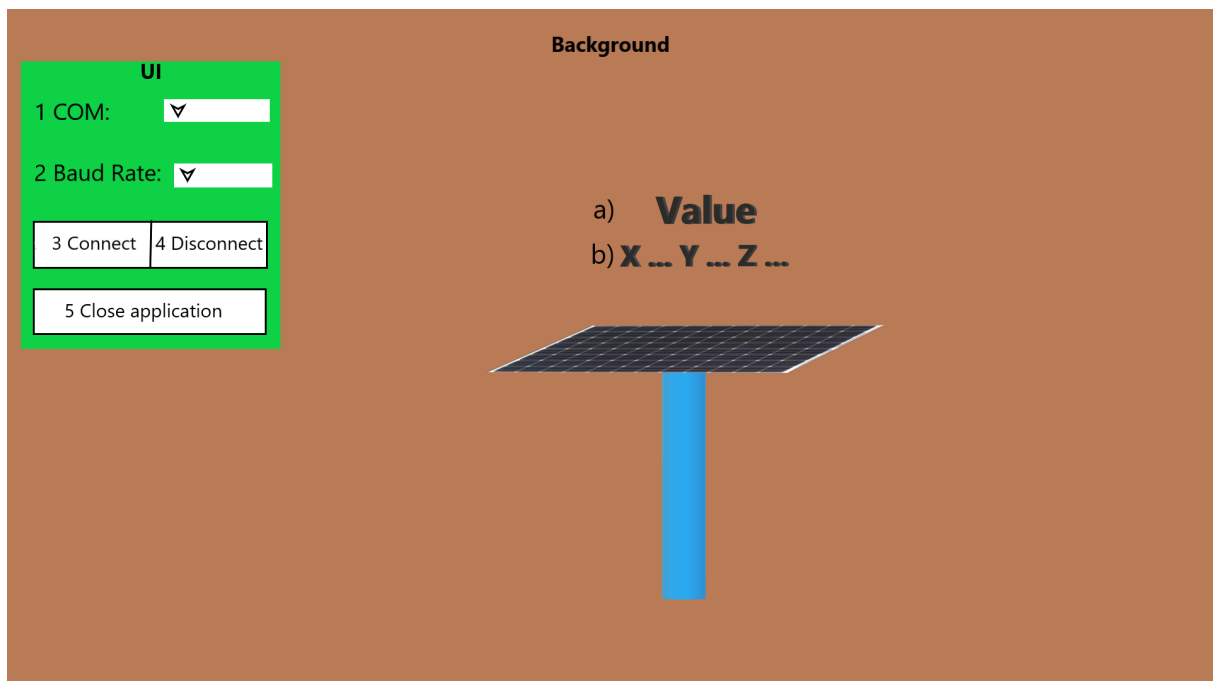
1. Lista wyboru nazwy portu szeregowego
Powinna umożliwić wybranie portu do którego podłączony jest sterownik.
2. Lista wyboru prędkości połączenia
Powinna zawierać takie prędkości wyrażone w bodach względem których przesłanie pakietu danych będzie trwało mniej niż $1/60[s]$.
3. Przycisk nawiązania połączenia
Przycisk umożliwiający nawiązanie połączenia ze sterownikiem po wybraniu odpowiednich parametrów.
4. Przycisk zakończenia połączenia
Przycisk umożliwiający zakończenie połączenia ze sterownikiem.
5. Przycisk zamknięcia aplikacji
Przycisk umożliwiający zamknięcie aplikacji. Powinien realizować również akcję zamykania połączenia jeśli nadal by było ono otwarte.

4.2 Funkcjonalność aplikacji

1. Wyświetlanie aktualnej wartości natężenia światła
Wartość natężenia światła powinna być wyświetlana nad platformą np w postaci napisu 3D.
2. Wyświetlanie aktualnej pozycji
Powyżej/poniżej wartości natężenia światła powinna być wyświetlana informacja o aktualnej pozycji. Za pomocą schematu XYZ.

4.3 Graficzna reprezentacja aplikacji

I. Schemat



Rysunek 3: Wygląd aplikacji

II. Szczegółowy opis UI

1 COM

Lista modyfikująca parametr odpowiedzialny za nazwę portu w skrypcie obsługi portu szeregowego.

2 Baud Rate

Lista modyfikująca parametr odpowiedzialny za prędkość transmisji w skrypcie obsługi portu szeregowego.

3 Connect

Przycisk wywołujący funkcję odpowiedzialną za nawiązanie połączenia w skrypcie obsługi portu szeregowego.

4 Disconnect

Przycisk wywołujący funkcję odpowiedzialną za zamknięcie połączenia w skrypcie obsługi portu szeregowego.

5 Close application

Przycisk wywołujący skrypt odpowiedzialny za zamknięcie aplikacji.

III. Szczegółowy opis napisów interaktywnych

a) Value

Napis wyświetlający bieżącą wartość natężenia światła. Połączony ze skryptem rotacji aby dostosowywał swoje położenie względem kamery.

b) X ... Y ... Z ...

Napis wyświetlający aktualne położenie we współrzędnych kartezjańskich. Połączony ze skryptem rotacji aby dostosowywał swoje położenie względem kamery.

5 Wstępne rezultaty

5.1 Zmiany w projekcie

Nastąpiła zmiana środowiska programistycznego z Unity na Qt + OpenGL. To pociągnęło za sobą zmiany w harmonogramie pracy i podejście do projektu. Najpierw zostanie stworzona komunikacja między PC a sterownikiem a następnie wizualizacja 3D. Dodatkowo zmieni się format przesyłanych danych. Aktualnie przewiduję że pakiet danych będzie wyglądał następująco:

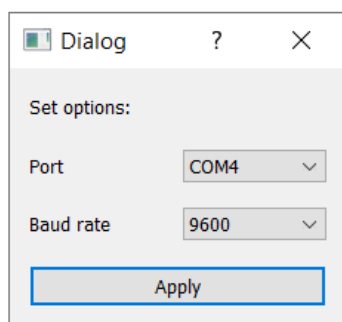
$$"H^1 \dots V^2 \dots L^3 \dots I^4 \dots CRC^5 \dots \backslash n^6".$$

Gdzie ... to poszczególne wartości. Natomiast separator to znak spacji.

5.2 Zrealizowane zadania

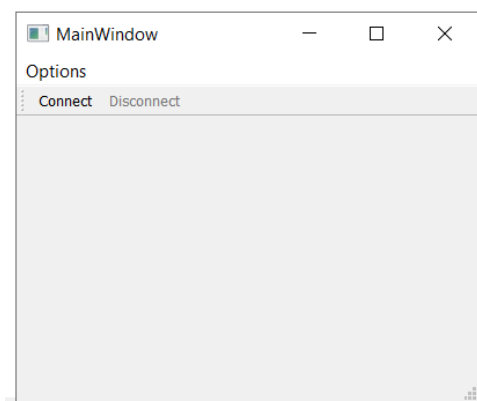
1. Graficzny interfejs użytkownika

Stworzyłem aplikację komunikującą się ze sterownikiem za pomocą portu szeregowego. Aplikacja po uruchomieniu prosi o podanie prędkości komunikacji w bodach oraz portu do którego został przyłączony sterownik. Domyślna prędkość to 9600 bodów. Natomiast lista portów wczytuje tylko te dostępne.



Rysunek 4: Okno opcji

Po zaakceptowaniu ustawień uruchamia się okno główne w którym mamy opcje Connect oraz Disconnect. Obie wzajemnie się wykluczają. Dodatkowo gdy połączenie jest aktywne wygaszona zostaje opcja zmiany ustawień połączenia.



Rysunek 5: Okno główne

¹rotacja horyzontalna

²rotacja wertykalna

³napięcie światła

⁴Zmierzone napięcie prądu

⁵32-bitowa suma kontrolna

⁶znak końca pakietu danych

2. Komunikacja

Komunikacja jest uruchamiana w osobnym wątku tak aby nie zakłócać pracy głównego okna. Port szeregowy został skonfigurowany z 8 bitami danych, bitem parzystości oraz bitem stopu. Aktualnie przesyłane dane wyświetlam za pomocą konsoli. Gdy suma kontrolna się nie zgadza wyświetlam komunikat o niepoprawnej ramce danych.

6 Zaawansowane rezultaty

6.1 Komunikacja

Zmianie uległ format danych. Zostały dodane dodatkowe pola. Oraz liczona jest suma kontrolna 8-bitowa. Nazwa sumy: CRC-8-Dallas/Maxim, Wielomian: 0x8C.

"H⁷... V⁸... L⁹... U¹⁰... I¹¹... P¹²... CRC¹³... \r\n¹⁴".

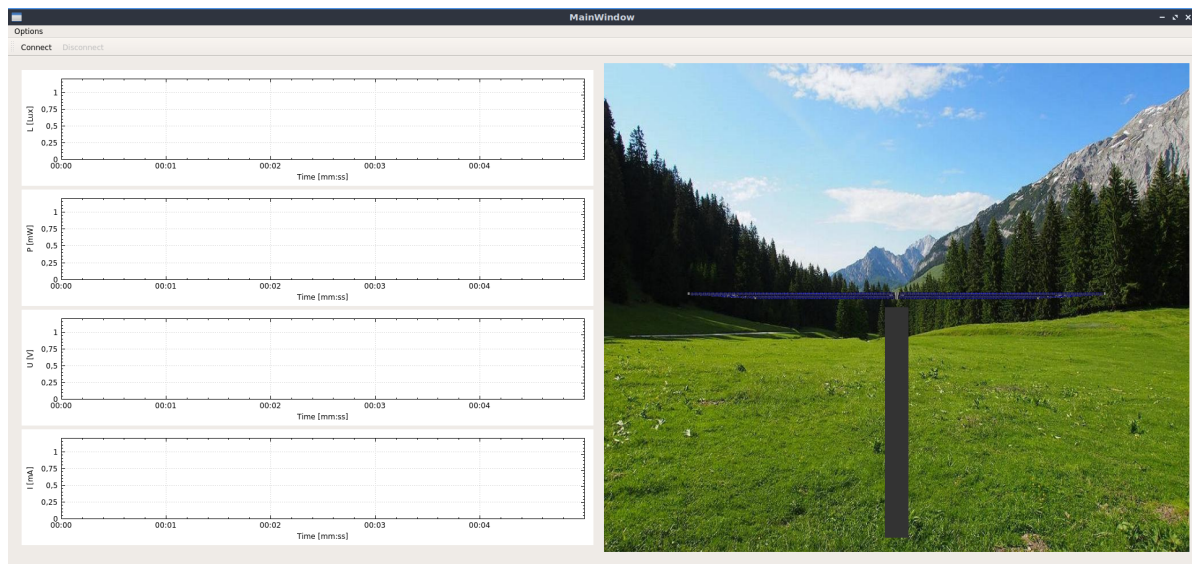
Przykładowe wyniki:

```
H 60 V 60 U 60 L 60 I 60 P 60 Checksum correct: 55
H 61 V 61 U 61 L 61 I 61 P 61 Checksum correct: 20
H 62 V 62 U 62 L 62 I 62 P 62 Checksum correct: a6
H 63 V 63 U 63 L 63 I 63 P 63 Checksum correct: d3
H 64 V 64 U 64 L 64 I 64 P 64 Checksum correct: b3
```

Rysunek 6: Przykładowe wyniki

6.2 Graficzny interfejs użytkownika

1. Okno główne

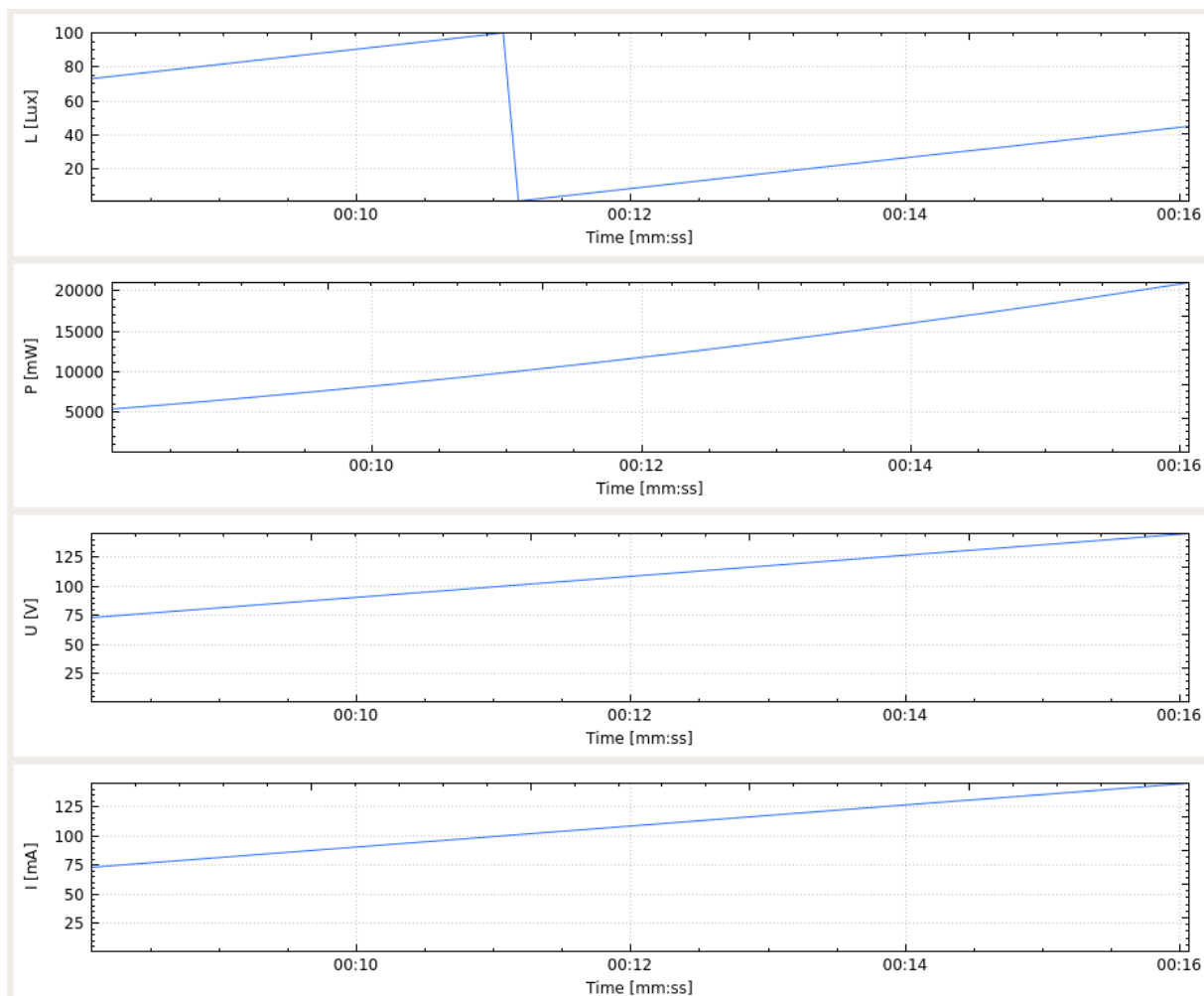


Rysunek 7: Okno główne

- ⁷rotacja horyzontalna
- ⁸rotacja wertykalna
- ⁹natężenie światła
- ¹⁰Napięcie
- ¹¹Prąd
- ¹²Moc
- ¹³8-bitowa suma kontrolna
- ¹⁴znak końca pakietu danych

Po lewej znajdują się wykresy wartości pomiarowych. Natomiast po prawej stronie mamy wizualizację platformy fotowoltaicznej.

2. Wykresy



Rysunek 8: Wykresy

Do rysowania wykresów używam biblioteki zewnętrznej `QCustomPlot`. Na osi pionowej znajdują się poszczególne wartości pomiarów. Na osi poziomej mamy czas w formacie minuty:sekundy. Wykres jest dynamiczny o stałym oknie czasowym. Na przykładowym rysunku 8 wynosi ono 8 sekund.

3. Wizualizacja



Rysunek 9: Wizualizacja

Zaimplementowałem dwuosiową rotację. Horyzontalna (cała platforma) oraz wertykalna (tylko panel fotowoltaiczny).

7 Rezultaty prawie końcowe

7.1 Zrealizowane zadania

Ze względu na wysoki poziom zaawansowania projektu ograniczyłem się do podstawowych zadań, a dostępne zasoby, wykorzystałem do realizacji projektów równoległych.

Zrealizowane zadania:

1. Stworzenie dokumentacji za pomocą generatora Doxygen.
2. Dopracowanie komunikacji między sterownikiem a komputerem.
 - Obliczenie ilości półkroków silnika krokowego
Ilość półkroków jakie wykonuje silnik w czasie pełnego obrotu wynosi 2734.
 - Wykorzystanie rzeczywistych pomiarów położenia horyzontalnego i wertykalnego.
 - Wykorzystanie rzeczywistych pomiarów natężenia światła.
 - Wykorzystanie dzielnika napięciowego do pomiaru przepływu prądu oraz napięcia.
3. Testy poprawności modyfikacji.

7.2 Zmiany w projekcie

Niestety w trakcie transportu nieodwracalnemu uszkodzeniu uległ panel fotowoltaiczny, co dyskwalifikuje jego użycie w projekcie. Na jego miejsce zostanie użyty fotorezystor, który będzie symulował jego działanie. Po pierwszych testach i wprowadzeniu współczynników korekcyjnych rozwiązanie działa poprawnie.