PROJEKT

**Wizualizacja Danych Sensorycznych**

# MiniSumo "*MiniWiluś*" — wizualizacja czujników

**Jakub Wilczyński 275459**

Prowadzący: dr inż. Bogdan Kreczmer

Katedra Cybernetyki i Robotyki
Wydziału Elektroniki, Fotoniki i Mikrosystemów
Politechniki Wrocławskiej

16 maj 2025

# Spis treści

# 1 Charakterystyka tematu projektu

Celem projektu jest stworzenie interaktywnej wizualizacji działania czujników robota minisumo. Wizualizacja ta będzie przedstawiać robota na środku ekranu, a także obszary działania jego czujników, w tym czujników odbiciowych, czujnika linii oraz czujnika inercyjnego. W ramach tej wizualizacji każdy z czujników będzie miał przypisany odpowiedni obszar działania, który będzie zmieniać kolor w zależności od wykrywania obiektów.

- **Czujnik odbiciowy**: Obszar działania czujników odbiciowych będzie reprezentowany jako trójkąt, który symuluje pole widzenia czujnika. Kolor trójkąta będzie zmieniał się z zielonego na czerwony, gdy czujnik wykryje przeszkodę. Zmiana koloru sygnalizuje, że robot napotkał obiekt w zasięgu czujnika.
- **Czujnik linii**: Czujnik linii, umieszczony na robocie, będzie monitorował obecność linii na podłożu. Gdy czujnik wykryje linię, obszar wokół niego zmieni kolor, sygnalizując, że robot jest na tej linii.
- **Czujnik inercyjny**: Jeśli robot zostanie uderzony, czujnik zarejestruje to zdarzenie i na ekranie pojawi się strzałka wskazująca kierunek, z którego nadeszło uderzenie. Strzałka będzie animowana i wskazywać dokładny punkt, z którego zostało wymierzone oddziaływanie.

# 2 Podcele realizacji projektu

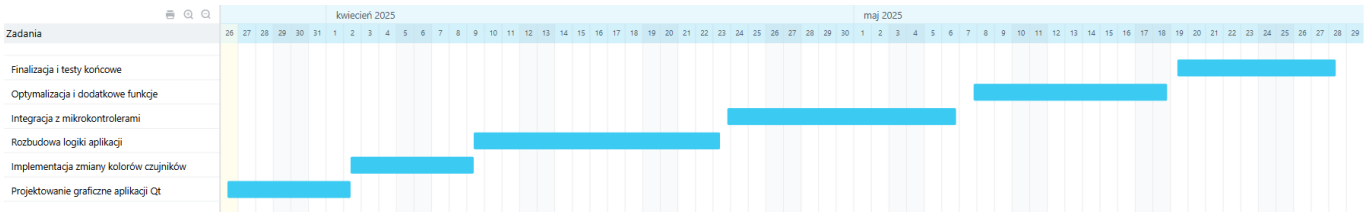
- **Opracowanie graficznej formy aplikacji (26 marca – 1 kwietnia)**
  - Stworzenie projektu interfejsu użytkownika w Qt.
  - Opracowanie centralnego widoku robota i obszarów czujników.
  - Implementacja statycznych kształtów wizualizujących pola działania czujników.
- **Implementacja dynamicznej wizualizacji czujników na bazie testowych danych (2–22 kwietnia)**
  - Implementacja logiki zmiany kolorów czujników odbiciowych i linii.
  - Wprowadzenie mechanizmu generowania testowych danych.
  - Implementacja wykresu IMU z podstawową wizualizacją kierunku uderzenia.
  - Optymalizacja podstawowej funkcjonalności UI.
- **Integracja z mikrokontrolerem i odbiór rzeczywistych danych (23 kwietnia – 16 maja)**
  - Implementacja komunikacji Qt — ESP-S3 — ATmega328P.
  - Odbiór i interpretacja rzeczywistych danych z czujników.

  - Aktualizacja wizualizacji na podstawie przesyłanych wartości.
  - Testy i poprawki związane z wydajnością i stabilnością aplikacji.
- **Optymalizacja i finalizacja aplikacji (17–29 maja)**
  - Poprawa płynności wizualizacji i redukcja opóźnień.
  - Implementacja funkcji zapisu i odczytu danych (np. pliki CSV).
  - Testy końcowe, usuwanie błędów i przygotowanie końcowej wersji.
  - Finalna prezentacja projektu.

# 3 Kamienie milowe

- **Milestone 1 – 26 kwietnia: Wstępne Wyniki**
  - Działająca aplikacja okienkowa Qt z dynamiczną zmianą kolorów sensorów na podstawie testowych danych.
  - Wstępna struktura do przyszłej integracji z mikrokontrolerami.
- **Milestone 2 – 16 maja: Zaawansowane Wyniki**
  - W pełni funkcjonalna aplikacja połączona z mikrokontrolerem.
  - Odczyt rzeczywistych danych czujników.
  - Wizualizacja wykresu IMU ze strzałką wskazującą kierunek uderzenia.
- **Milestone 3 – 29 maja: Finalne Wyniki**
  - Gotowy system połączony z robotem.
  - Ostateczne poprawki wydajnościowe i optymalizacyjne.
  - Kompleksowe testy końcowe.

# 4 Diagram Gantta



# 5 Harmonogram realizacji projektu

- **Tydzień 1 (26 marca – 1 kwietnia)**
  - Projekt graficzny aplikacji w Qt.
  - Stworzenie widoku głównego robota i obszarów czujników.
  - Implementacja statycznych obszarów wykrywania dla czujników.
- **Tydzień 2 (2–8 kwietnia)**
  - Implementacja zmiany kolorów czujników linii i odbiciowych na podstawie testowych danych.
  - Stworzenie mechanizmu generowania sztucznych danych do symulacji.
- **Tydzień 3 (9–15 kwietnia)**
  - Rozszerzenie systemu wizualizacji o podstawowy wykres IMU.
  - Implementacja pierwszej wersji dynamicznej zmiany kierunku uderzenia na wykresie IMU.
- **Tydzień 4 (16–22 kwietnia)**
  - Optymalizacja interfejsu użytkownika i poprawki.
  - Testy aplikacji na bazie sztucznie generowanych danych.
- **Tydzień 5 (23–29 kwietnia)**
  - Implementacja komunikacji Qt — ESP-S3.
  - Wstępne testy przesyłu danych z czujników do aplikacji.
- **Tydzień 6 (30 kwietnia – 6 maja)**
  - Integracja z ATmega328P i pełny odbiór rzeczywistych danych z czujników.
  - Dopasowanie wizualizacji do realnych wartości.
- **Tydzień 7 (7–13 maja)**
  - Testy poprawności odczytu i interpretacji danych z czujników.
  - Optymalizacja wydajności aplikacji.
- **Tydzień 8 (14–16 maja)**
  - Dopracowanie wykresu IMU i wizualizacji historii danych.
  - Zaawansowane wyniki: aplikacja w pełni połączona z robotem.
- **Tydzień 9 (17–22 maja)**
  - Optymalizacja kodu i poprawa wydajności renderowania.

  - Dodanie funkcji zapisu i odtwarzania danych.

- **Tydzień 10 (23–26 maja)**
  - Testy końcowe i poprawki błędów.
  - Finalizacja animacji i UI.

- **Tydzień 11 (27–29 maja)**
  - Ostateczne testy aplikacji z robotem.
  - Prezentacja końcowych wyników projektu.

# 6 Projekt graficznego interfejsu użytkownika

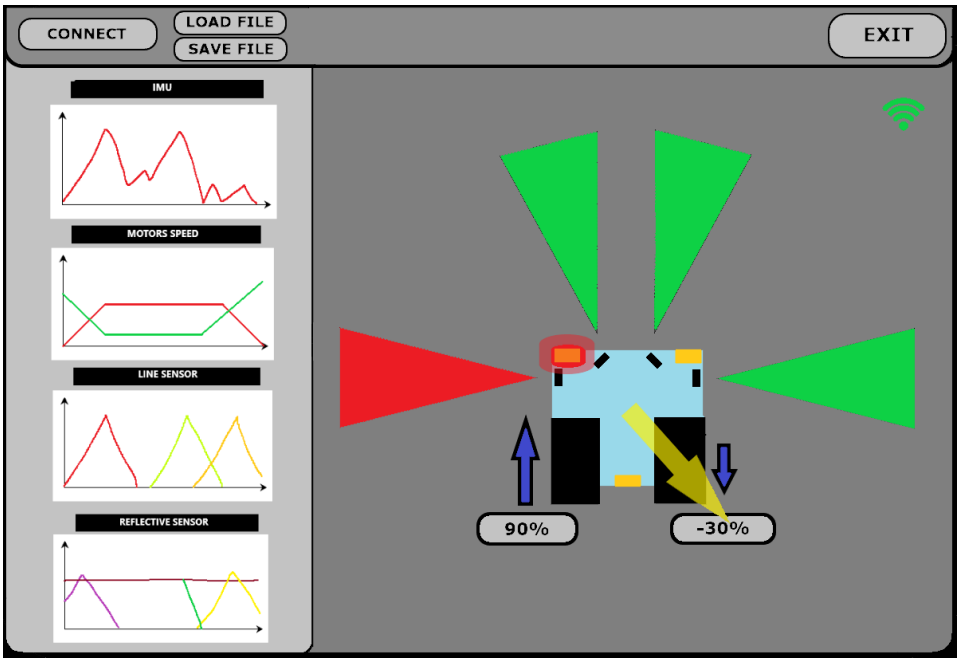
Projekt graficzny interfejsu użytkownika został zaprojektowany w taki sposób, aby w przejrzysty i intuicyjny sposób prezentować działanie czujników robota minisumo. Centralnym elementem interfejsu jest wizualizacja robota, który znajduje się na środku ekranu. Dookoła niego rozmieszczone są obszary działania poszczególnych czujników, których stan jest dynamicznie aktualizowany na podstawie napływających danych. Po lewej stronie okna znajduje się stały panel wykresów, umożliwiający analizę działania czujników i silników w czasie rzeczywistym. Na górze ekranu umieszczony jest pasek narzędzi zawierający opcje zarządzania połączeniem oraz plikami.

## 6.1 Funkcjonalności aplikacji

Aplikacja dostarcza następujące funkcjonalności:

- **Wizualizacja robota i jego czujników** — Centralny widok przedstawiający robota minisumo z obszarami działania czujników.
- **Dynamiczna zmiana koloru czujników odbiciowych** — Cztery trójkąty wokół robota zmieniają kolor z zielonego na czerwony, gdy wykryją przeciwnika.
- **Wizualizacja czujników linii** — Trzy żółte prostokąty symbolizujące czujniki linii, które w momencie wykrycia linii wyświetlają obok siebie czerwony znacznik.
- **Wskazanie kierunku uderzenia** — Żółta strzałka pojawia się na ekranie, wskazując stronę, z której robot został uderzony.
- **Wskaźniki prędkości silników** — Obok kół robota znajdują się niebieskie strzałki, które:
  - Powiększają się wraz ze wzrostem wartości PWM.
  - Obracają się o 180 stopni, gdy koło ma się kręcić do tyłu.
  - Pod nimi wyświetlana jest wartość PWM w procentach (wartości ujemne oznaczają ruch wsteczny).

- **Panel wykresów** — Po lewej stronie ekranu znajduje się panel, zawierający:
  - **Wykres IMU** — Wizualizacja orientacji i ruchu robota.
  - **Motor Speed** — Prędkości silników lewego i prawego jako oddzielne linie na wykresie.
  - **Line Sensor** — Trzy linie różnych kolorów odpowiadające poszczególnym czujnikom linii.
  - **Reflective Sensor** — Cztery linie różnych kolorów reprezentujące każdy czujnik odbiciowy.
- **Opcje ładowania i zapisu danych** — Możliwość zapisu i odczytu parametrów oraz logów pracy czujników.
- **Integracja z mikrokontrolerem** — Pobieranie rzeczywistych danych z ESP-S3 i ATmega328P.
- **Pasek narzędzi** — Znajdujący się na górze ekranu pasek narzędzi zawiera:
  - Pole **CONNECT** do nawiązywania połączenia z robotem.
  - Pole **LOAD FILE** do wczytywania parametrów i logów.
  - Pole **SAVE FILE** do zapisywania danych.
  - Pole **EXIT** do zamykania aplikacji.
- **Ikonka połączenia Wi-Fi** — W prawym górnym rogu znajduje się ikonka sygnalizująca status połączenia:
  - Zielona ikonka oznacza aktywne połączenie.
  - Czerwona ikonka oznacza brak połączenia.




Projekt graficzny 1: Projekt graficzny aplikacji

## 6.2 Scenariusze działania aplikacji

Poniżej przedstawiono kluczowe scenariusze użycia aplikacji, które ilustrują sposób interakcji użytkownika z interfejsem oraz funkcjonalności aplikacji. Każdy scenariusz opisuje kroki niezbędne do uzyskania konkretnego efektu wizualizacji i analizy danych.

### 6.2.1 Scenariusz 1: Wykrywanie przeciwnika przez czujniki odbiciowe

1. Uruchomienie aplikacji — robot wyświetla się na środku ekranu.
2. Czujniki odbiciowe mają zielony kolor (brak wykrytego obiektu).
3. Po wykryciu przeciwnika odpowiedni trójkąt zmienia kolor na czerwony.
4. Po usunięciu obiektu czujnik wraca do koloru zielonego.

### 6.2.2 Scenariusz 2: Wykrywanie linii przez czujniki linii

1. Robot przemieszcza się po powierzchni.
2. Jeśli czujnik linii wykryje linię, obok odpowiedniego prostokąta pojawia się czerwony znacznik.
3. Gdy robot opuści linię, znacznik znika.

### 6.2.3 Scenariusz 3: Wykrycie uderzenia

1. Robot odbiera uderzenie od przeciwnika.
2. Czujnik inercyjny rejestruje kierunek uderzenia.
3. Na ekranie pojawia się żółta strzałka wskazująca stronę, z której nadszedł cios.
4. Strzałka animuje się przez krótki czas, a następnie znika.

### 6.2.4 Scenariusz 4: Analiza danych na wykresach

1. Użytkownik analizuje panel wykresów po lewej stronie.
2. Wykresy aktualizują się na bieżąco na podstawie danych.
3. Możliwość analizy wartości czujników w czasie rzeczywistym.

### 6.2.5 Scenariusz 5: Ładowanie i zapis parametrów

1. Użytkownik naciska przycisk „LOAD FILE” na pasku narzędzi.
2. Otwiera się standardowe okno dialogowe wyboru pliku.
3. Po wybraniu pliku aplikacja wczytuje zapisane wartości parametrów.
4. Możliwość zapisania aktualnych danych do pliku CSV poprzez naciśnięcie przycisku „SAVE FILE” na pasku narzędzi.

#### 6.2.6 Scenariusz 6: Nawiązywanie i monitorowanie połączenia

1. Użytkownik naciska przycisk „CONNECT” na pasku narzędzi.
2. Aplikacja próbuje nawiązać połączenie z robotem.
3. Jeśli połączenie zostanie nawiązane, ikonka Wi-Fi w prawym górnym rogu zmienia kolor na zielony.
4. Jeśli połączenie zostanie utracone, ikonka Wi-Fi zmienia kolor na czerwony.

# 7 Opis komunikacji z ESP32

## 7.1 Cel protokołu

Celem komunikacji jest umożliwienie wymiany danych pomiędzy mikrokontrolerem ESP32 (zainstalowanym na robocie mobilnym) a aplikacją komputerową napisaną w Qt poprzez lokalną sieć WiFi z wykorzystaniem protokołu TCP/IP.

## 7.2 Warstwa transportowa

- Protokół: TCP (Transmission Control Protocol),
- Port: 1234 (ustalony na stałe, możliwość zmiany w aplikacji),
- Adres IP: przydzielony dynamicznie (DHCP), w przyszłości przydzielanie statyczne,
- Tryb połączenia: klient-serwer,
  - ESP32: serwer TCP,
  - Aplikacja Qt: klient TCP inicjujący połączenie.

## 7.3 Format wiadomości

Wiadomości przesyłane są w formacie tekstowym (UTF-8), jako linie zakończone znakiem nowej linii `\n`. Format opiera się na konwencji CSV (Comma-Separated Values).

**Format ramki:**

```
czas_ms,tof1,tof2,tof3,tof4,line1,line2,line3,motor1,motor2,imuX,imuY\n
```

**Przykład:**

```
500,1,0,1,1,0,0,1,60,-100,33.11,-1.55\n
```

### 7.4 Opis pól ramki danych

| Pole | Typ | Opis |
|---|---|---|
| czas_ms | int | Znacznik czasu (w milisekundach) |
| tof1–4 | bool | Dane z czujników ToF (0/1) |
| line1–3 | bool | Dane z czujników linii (0/1) |
| motor1–2 | int | Wartości PWM |
| imuX, imuY | float | wartości kolejno X i Y z czujnika inercyjnego |

Tabela 1: Struktura ramki danych

### 7.5 Synchronizacja i transmisja

- Połączenie inicjowane jest przez aplikację (klient),
- ESP32 obsługuje jedno połączenie jednocześnie,
- Dane są wysyłane cyklicznie (np. co 100–500 ms),
- Każda linia danych jest kompletna i niezależna.

### 7.6 Właściwości ramki danych

- Format: tekstowy, UTF-8 (ASCII kompatybilny),
- Rozdzielacz pól: przecinek `,`,
- Znak końca: nowa linia `\n`,
- Przykładowa długość ramki: 33 bajty — długość ta wynika z liczby znaków ASCII (cyfry, przecinki, minusy, kropki).

## 8 Wstępne rezultaty i aktualny stan projektu

### 8.1 Zrealizowane funkcjonalności

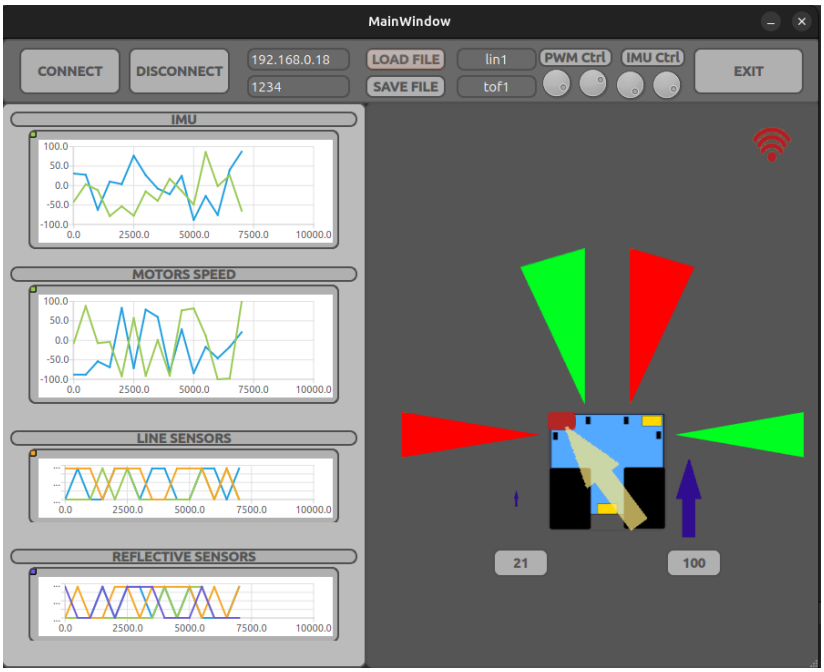
W obecnej wersji projektu zaimplementowano następujące elementy:

- Komunikacja TCP z mikrokontrolerem ESP32 i odczyt danych z czujników.
- Obsługa odczytu z pliku CSV (tryb offline).
- Dynamiczne wizualizacje na wykresach:
  - prędkości silników,
  - czujników odbiciowych,
  - czujników linii,
  - danych z IMU.
- Interaktywna reprezentacja robota z podświetleniem aktywnych czujników.

- Wizualizacja wartości PWM za pośrednictwem strzałek oraz dokładnej wartości wyświetlającej się pod strzałką.
- Wizualizacja czujnika inercyjnego.

### 8.1.1 Dodane funkcjonalności

- Dodano możliwość rozłączenia (przycisk `Disconnect`),
- Dodano manualne sterowanie czujnikiem odbiciowym i czujnikiem linii za pomocą przycisków, które zmieniają stany czujników.
- Dodano manualne sterowanie PWM za pomocą dwóch pokręteł,
- Dodano ręczne ustawianie wartości IMU za pomocą dwóch pokręteł
- Dodano pola tekstowe umożliwiające ręczne wprowadzenie adresu IP i portu serwera ESP32.




Wygląd aplikacji 1: Wygląd aktualnej aplikacji

## 8.2 Elementy do ukończenia

Do realizacji pozostają:

- Zmiana ikony WiFi po nawiązaniu połączenia.
- Zmiana koloru strzałki w zależności od kierunku obrotów (ujemne PWM).
- Dodanie przewijania dla wykresów dla większej ilości danych.
- Skalowanie osi X w zależności od ilości danych.
- Dodanie osi z opisami i jednostkami do wykresów.

- Możliwość zapisu danych do pliku CSV (`SAVE FILE`).
- Przeniesienie pól IP i portu do osobnego okna dialogowego.
- Dodanie skalowania aplikacji.
- Dodanie możliwości zmiany języka interfejsu.

# 9 Rezultaty zaawansowane

W ramach dalszego rozwoju aplikacji wprowadzono szereg istotnych modyfikacji i usprawnień, które znacząco wpłynęły na funkcjonalność i użyteczność systemu. Poniżej przedstawiono szczegółowy opis wprowadzonych zmian oraz obszarów wymagających dalszej optymalizacji.

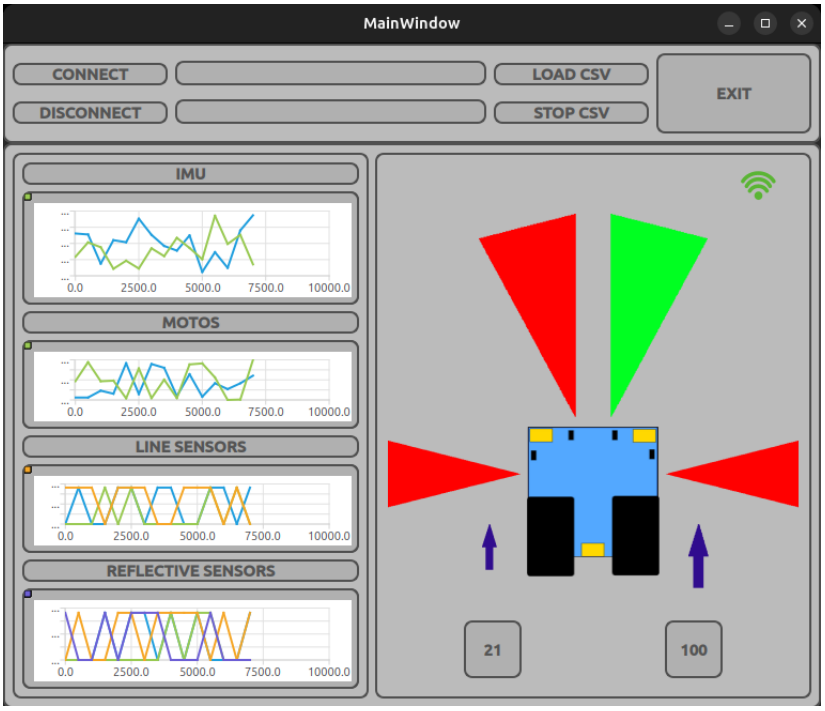
## 9.1 Wprowadzone modyfikacje

Zaimplementowano możliwość skalowania całej aplikacji, co pozwala na lepsze dostosowanie interfejsu do różnych rozmiarów ekranu. Jest to kluczowa funkcjonalność z punktu widzenia uniwersalności zastosowania systemu.

### 9.1.1 Interfejs użytkownika

W ramach optymalizacji interfejsu użytkownika wprowadzono następujące zmiany:

- Usunięto przyciski kontrolne z górnego panelu aplikacji (PWM CTR, IMU STR, LIN1, TOF1)
- Zaimplementowano dynamiczną zmianę koloru ikony Wi-Fi w zależności od stanu połączenia, co zapewnia lepszą informację zwrotną dla użytkownika




Wygląd aplikacji 2: Aktualny wygląd aplikacji po wprowadzonych modyfikacjach

## 9.2 Obszary wymagające optymalizacji

Zidentyfikowano następujące elementy wymagające dopracowania w kontekście skalowania:

- Pola czujników odbiciowych nie dostosowują swojej wielkości odpowiednio do zmian
- Strzałki przy kołach nie zmieniają wielkości podczas zmiany wartości PWM
- Konieczność implementacji strzałki IMU dla lepszej wizualizacji danych z czujnika