Technical Report

AGH University of Science and Technology June 2025

APLIKACJA OKIENKOWA NA SYSTEM GNU/LINUX DO KOMUNIKACJI Z PŁYTKĄ ESP32

Autor: Bartłomiej Damasiewicz

Akademia Górniczo-Hutnicza

Spis treści

Lista oznaczeń

ESP32	Mikrokontroler
QT	Biblioteka programistyczna C++ do tworzenia aplikacji graficznych i wieloplatformowych.
QSerialPort	Klasa Qt służąca do komunikacji z portem szeregowym (UART) w systemach Linux/Windows.
QTcpSocket	Klasa Qt umożliwiająca tworzenie klienta TCP do komunikacji przez sieć.
TCP/IP	Zestaw protokołów komunikacyjnych używanych w internecie i sieciach lokalnych.
UART	Uniwersalny asynchroniczny odbiornik-nadawca; interfejs szeregowy do komunikacji z mikrokontrolerami.
WiFi	Technologia bezprzewodowej transmisji danych w sieciach lokalnych, oparta na standardzie IEEE 802.11
QTest	Framework do testowania aplikacji QT

Wstęp

Projekt ten jest implementacją desktopowej aplikacji narzędziowej umożliwiającej komunikację z mikrokontrolerem ESP32 za pośrednictwem dwóch kanałów transmisji: UART oraz Wi-Fi. Aplikacja została zaprojektowana z wykorzystaniem frameworka Qt, który zapewnia wieloplatformowe środowisko graficzne, a także obsługę portów szeregowych, wykresów i sieci.

Głównym celem projektu było stworzenie prostego i funkcjonalnego interfejsu użytkownika, który umożliwia:

- wyszukiwanie dostępnych urządzeń po USB,
- łączenie się z ESP32 za pomocą portu szeregowego lub sieci TCP/IP,
- sterowanie diodą LED na płytce ESP32,
- odbieranie i wizualizację danych z czujnika (np. temperatury, wilgotności),
- rejestrowanie logów komunikacji.

Projekt został zrealizowany w oparciu o wzorce projektowe Strategia (umożliwiający wybór metody komunikacji) oraz Obserwator (do aktualizacji wykresu w reakcji na nowe dane). Zastosowanie tych wzorców pozwoliło uzyskać modularną i rozszerzalną architekturę, która ułatwia utrzymanie i przyszłą rozbudowę systemu o kolejne protokoły lub źródła danych.

ESP32, działające w trybie Access Point, pełni rolę prostego serwera TCP, który odbiera polecenia sterujące oraz wysyła dane sensoryczne w ustalonym formacie. Z kolei aplikacja po stronie komputera działa jako klient TCP lub jako master portu szeregowego i integruje funkcjonalności z poziomu jednego interfejsu graficznego.

Projekt został wykonany w środowisku GNU/Linux (system Ubuntu 24.04) i może być wykorzystywany jako narzędzie diagnostyczno-edukacyjne lub punkt wyjścia do bardziej zaawansowanych systemów komunikacji z mikrokontrolerami.

1. Funkcjonalność

Aplikacja umożliwia użytkownikowi przeprowadzenie podstawowej komunikacji oraz wymiany danych z mikrokontrolerem ESP32 poprzez dwa niezależne kanały transmisji: interfejs szeregowy UART oraz sieć Wi-Fi. Interfejs graficzny aplikacji został zaprojektowany w sposób intuicyjny, z podziałem na zakładki i panele funkcyjne.

Główne funkcjonalności aplikacji to:

1.1.1 Wykrywanie urządzeń i nawiązywanie połączenia

- Wyszukiwanie portów szeregowych – po naciśnięciu przycisku „Search” aplikacja skanuje dostępne porty COM (np. /dev/ttyUSB0) i wyświetla je w rozwijanym menu.
- Połączenie przez UART – użytkownik może wybrać wykryty port i połączyć się z urządzeniem z wykorzystaniem standardowych parametrów (baud rate 9600, brak parzystości, 8 bitów danych, 1 bit stopu)
- Połączenie przez Wi-Fi (TCP) – użytkownik może ręcznie podać adres IP i port serwera TCP działającego na ESP32, aby nawiązać połączenie przez sieć lokalną.

1.1.2 Sterowanie urządzeniem

- Sterowanie diodą LED – aplikacja umożliwia włączanie i wyłączanie diody LED na ESP32 za pomocą przycisków „LED ON” oraz „LED OFF”. Polecenia są przesyłane w formie prostych znaków (1, 0) przez UART lub TCP.
- Wysyłanie komunikatów – aplikacja posiada mechanizm wysyłania dowolnych wiadomości tekstowych do urządzenia.

1.1.3 Odbiór i przetwarzanie danych

- Odbiór danych z czujnika – ESP32 cyklicznie przesyła dane z czujnika. Aplikacja nasłuchuje odpowiedzi i filtruje linie z danymi.
- Rejestracja logów – każda wiadomość przychodząca i wychodząca jest zapisywana w logach z oznaczeniem czasu.

1.1.4 Wizualizacja danych

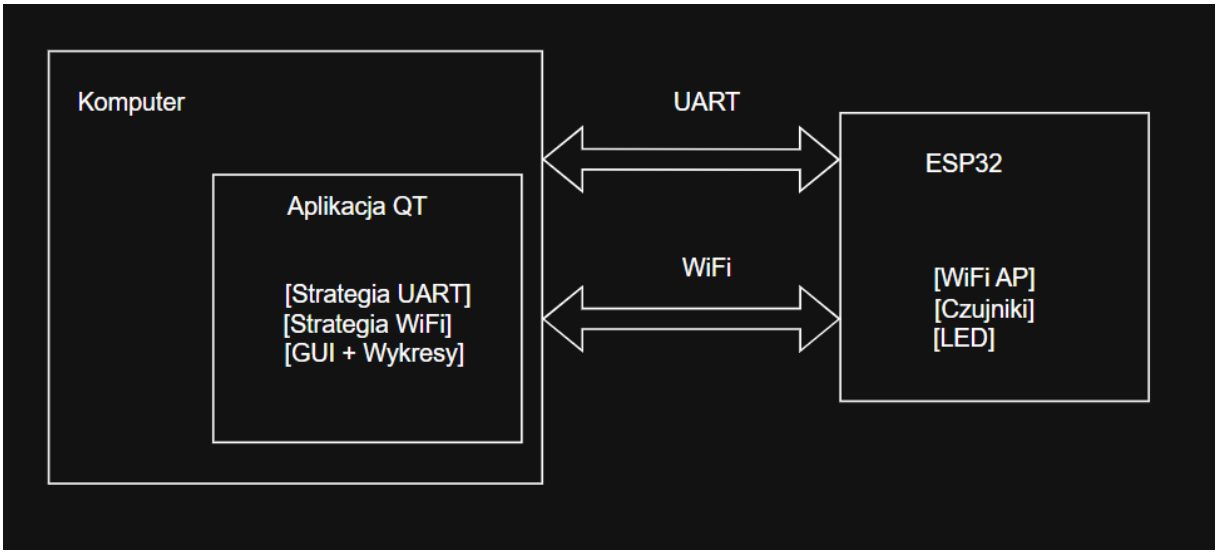
- Wykres danych sensorycznych – druga zakładka interfejsu zawiera dynamiczny wykres zbudowany z użyciem biblioteki QtCharts, który aktualizuje się w czasie rzeczywistym. Oś X reprezentuje czas (kolejne próbki), a oś Y – wartość zmierzoną przez czujnik.

1.1.5 Obsługa wielu strategii komunikacji

- Modularna architektura komunikacji – aplikacja korzysta ze wzorca strategii, co umożliwia łatwe przełączanie między komunikacją przez UART a TCP/IP. Użytkownik może samodzielnie zdecydować, z którego medium chce korzystać.

2. Analiza problemu

Współczesne systemy mikrokontrolerowe, takie jak ESP32, oferują bogaty zestaw funkcji komunikacyjnych – od interfejsów szeregowych, przez sieci bezprzewodowe, aż po protokoły internetowe. Jednak wciąż pojawia się potrzeba tworzenia dedykowanych narzędzi klienckich, które umożliwiają wygodną komunikację z urządzeniami w sposób bardziej przystępny niż terminal tekstowy. Głównym celem było opracowanie uniwersalnego narzędzia komunikacyjnego wspierającego zarówno tryb przewodowy (UART), jak i bezprzewodowy (Wi-Fi TCP), z dodatkowymi funkcjami diagnostycznymi i wizualizacyjnymi.



Rysunek 1 Model działania systemu

2.1.1 Potrzeby funkcjonalne

- Detekcja i wybór dostępnych urządzeń – użytkownik powinien mieć możliwość łatwego znalezienia portów COM dostępnych w systemie.
- Możliwość wyboru metody komunikacji – interfejs powinien umożliwiać użytkownikowi przełączanie się między komunikacją szeregową a komunikacją sieciową (TCP).
- Dwukierunkowa transmisja danych – system musi umożliwiać zarówno wysyłanie komend sterujących (np. do LED), jak i odbieranie danych z czujników.

- Rejestracja historii komunikacji – każda wymiana informacji powinna być zapisywana w formie logu z uwzględnieniem czasu.
- Wizualizacja danych czujnikowych – dane przychodzące z czujnika (np. temperatura, wilgotność) powinny być prezentowane w czasie rzeczywistym na wykresie.

2.1.2 Wymagania techniczne

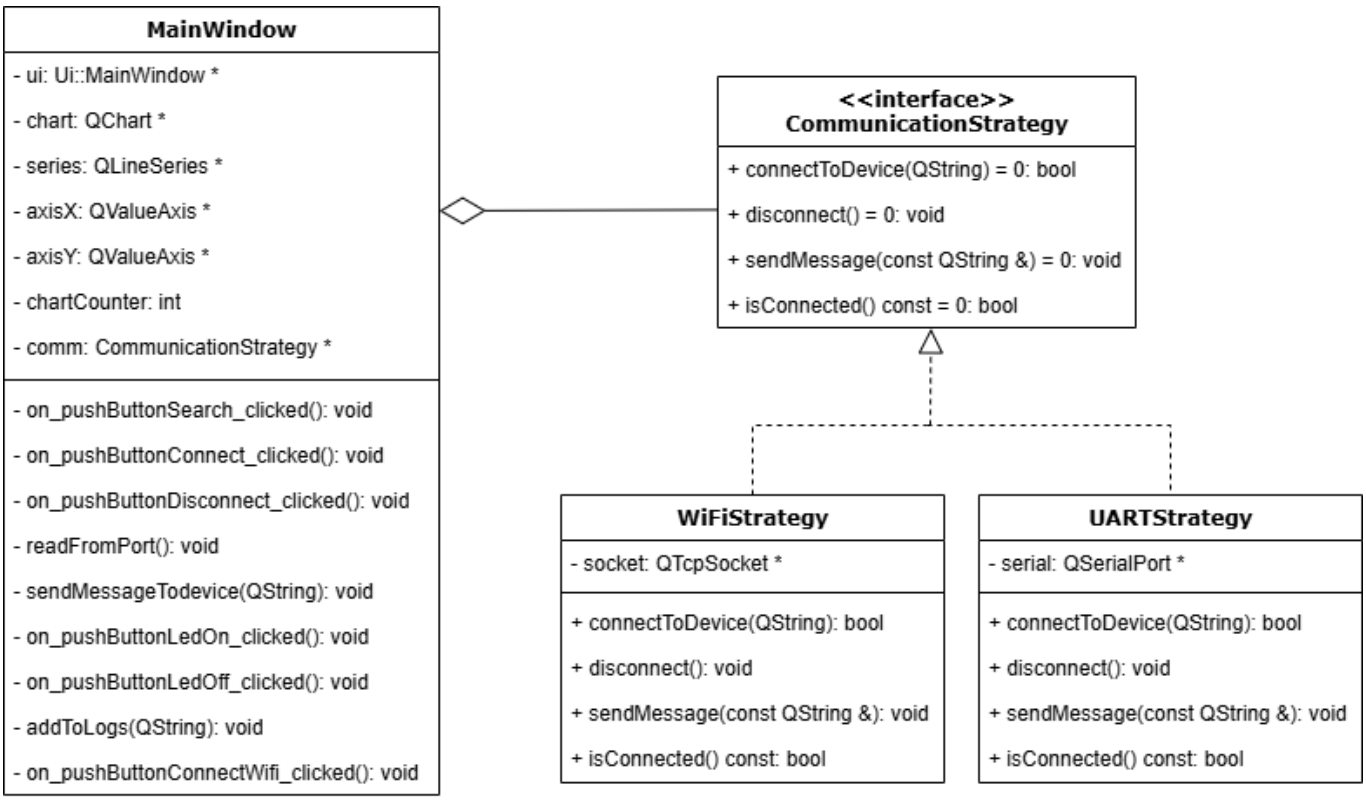
- Obsługi asynchronicznego odbioru danych z portu szeregowego i gniazda TCP,
- Implementacji wzorca Strategia dla obsługi różnych kanałów komunikacji,
- Użycia wzorca Obserwator do aktualizacji danych wizualnych (np. wykresów),
- Możliwości działania na systemie GNU/Linux bez potrzeby instalowania dodatkowych sterowników (dla UART).

2.1.3 Ograniczenia i założenia

- ESP32 działa jako punkt dostępowy Wi-Fi oraz serwer TCP nasłuchujący na wskazanym porcie (domyślnie 80),
- Dane z czujnika są przesyłane w prostym formacie tekstowym, co upraszcza parsowanie,
- Komunikacja przez UART wymaga ręcznego wskazania portu, a przez TCP – ręcznego wpisania adresu IP i portu.

3. Projekt techniczny

Projekt aplikacji został zaprojektowany w sposób modularny z uwzględnieniem wzorców projektowych. Głównym celem architektury było oddzielenie warstwy komunikacyjnej od logiki interfejsu użytkownika oraz zapewnienie łatwej rozbudowy o kolejne protokoły lub źródła danych.



Rysunek 2 Diagram klas

3.1.1 Architektura ogólna

System składa się z dwóch głównych komponentów:

- Aplikacja kliencka Qt (interfejs graficzny + logika komunikacji),
- Mikrokontroler ESP32 (serwer TCP / interfejs UART + czujnik + dioda LED).

Po stronie aplikacji klienckiej zastosowano wzorzec Strategia, pozwalający dynamicznie wybrać sposób komunikacji (UART lub TCP). Odbiór danych z czujnika jest przekazywany do wykresu i logów za pomocą mechanizmu sygnałów Qt, realizującego wzorzec Obserwator.

3.1.2 Opis Klas

CommunicationStrategy

Abstrakcyjna klasa bazowa definiująca interfejs dla wszystkich strategii komunikacyjnych. Zawiera metody do łączenia, rozłączania, wysyłania danych i sprawdzania stanu połączenia.

UARTStrategy

Klasa implementująca komunikację szeregową z użyciem QSerialPort. Obsługuje otwieranie portu, konfigurację parametrów transmisji oraz wysyłanie wiadomości.

WiFiStrategy

Strategia TCP/IP wykorzystująca QTcpSocket. Umożliwia połączenie z serwerem TCP na ESP32 oraz przesyłanie komend w postaci znakowej.

MainWindow

Główna klasa interfejsu graficznego. Odpowiada za obsługę przycisków, wybór strategii komunikacji, prezentację logów oraz aktualizację wykresu.

4. Opis realizacji

Proces realizacji projektu został podzielony na kilka kluczowych etapów: utworzenie interfejsu graficznego w Qt, implementacja strategii komunikacji (UART/Wi-Fi), integracja z mikrokontrolerem ESP32 oraz obsługa danych z czujnika. Poniżej przedstawiono najważniejsze fragmenty kodu wraz z ich omówieniem.

4.1.1 Tworzenie interfejsu graficznego

Interfejs użytkownika został zaprojektowany w narzędziu Qt Designer i zaimplementowany w klasie MainWindow. Główne elementy GUI to:

- przyciski do nawiązywania i zrywania połączenia przez UART i Wi-Fi,
- lista rozwijana z dostępnymi portami szeregowymi,
- przyciski sterujące diodą LED,
- pole tekstowe do wyświetlania logów komunikacyjnych,
- zakładka z dynamicznym wykresem do prezentacji danych z czujnika.

4.1.2 Implementacja wzorca strategii komunikacji

Zgodnie z wzorcem projektowym Strategia, utworzono wspólny interfejs CommunicationStrategy oraz dwie klasy dziedziczące: UARTStrategy i WiFiStrategy. Dzięki temu w klasie MainWindow można dynamicznie przełączać metodę komunikacji bez zmiany logiki aplikacji.

4.1.3 Odbieranie i przetwarzanie danych z ESP32

Dane z czujnika są przesyłane przez ESP32 za pomocą UART lub TCP. Aplikacja Qt analizuje odebrane linie i filtruje wartości, które następnie trafiają na wykres.

4.1.4 Obsługa mikrokontrolera ESP32

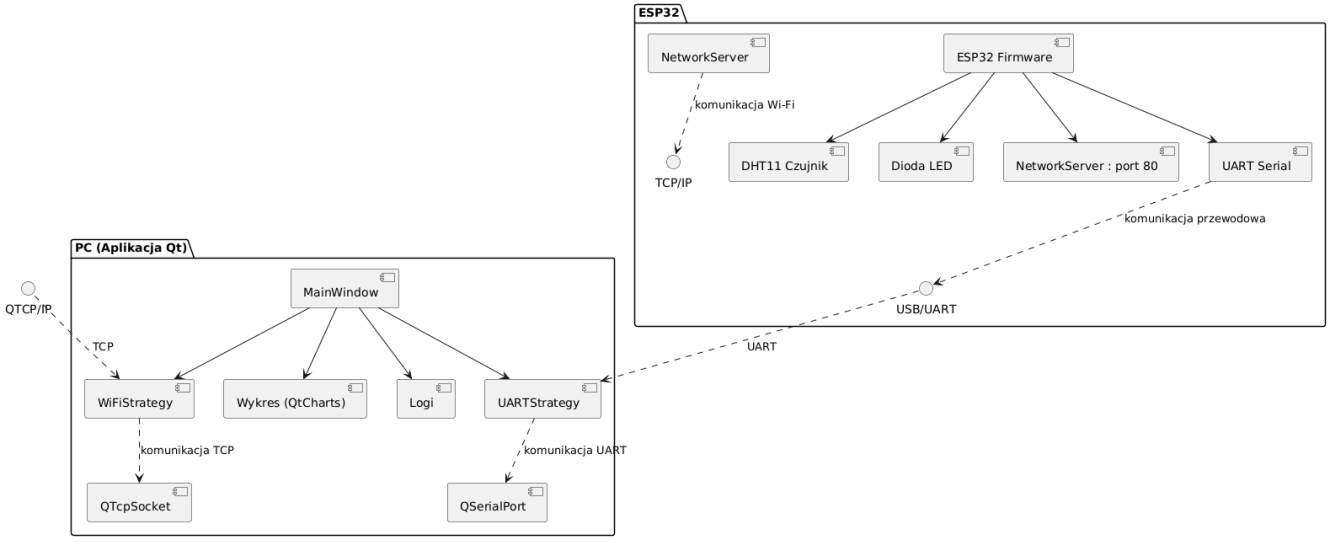
ESP32 pracuje w trybie Access Point i działa jako serwer TCP nasłuchujący na porcie 80. Obsługuje komendy oraz odpowiada danymi pobranymi z czujnika DHT-11.

4.1.5 Wizualizacja danych

Moduł wykresu został zaimplementowany z użyciem QtCharts. Aplikacja rysuje wykres w czasie rzeczywistym na podstawie danych przychodzących przez UART lub TCP, z osią X odpowiadającą kolejnym próbom, a osią Y wartościom czujnika.

4.1.6 Logowanie komunikacji

Każda akcja użytkownika oraz każda wymiana danych z ESP32 jest rejestrowana w widocznym logu. Dodatkowo każda wiadomość oznaczona jest sygnaturą czasową, co ułatwia analizę historii połączenia.



Rysunek 3 Diagram komponentów aplikacji Qt + ESP32

5. Opis wykonanych testów

W celu sprawdzenia poprawności działania komponentów aplikacji, w szczególności strategii komunikacji szeregowej (UARTStrategy), przeprowadzono testy jednostkowe z wykorzystaniem frameworka QTest. Testy miały na celu weryfikację podstawowych funkcji klasy odpowiedzialnej za komunikację UART z urządzeniem ESP32.

5.1.1 Narzędzie testowe

Testy zostały napisane w języku C++ z użyciem wbudowanego frameworka testowego Qt (QtTest). Uruchamiane były jako osobna aplikacja testowa budowana niezależnie od głównej aplikacji.

5.1.2 Zakres testów

Nazwa testu	Opis
testConnectToInvalidPort()	Sprawdza, czy metoda connectToDevice() prawidłowo odrzuca błędną nazwę portu (np. COM_fake, /dev/fake).
testConnectToValidPort()	(opcjonalny) Sprawdza połączenie z rzeczywistym portem szeregowym (należy podać port ręcznie w kodzie).
testIsConnectedBeforeAndAfterConnect()	Weryfikuje stan połączenia przed i po wywołaniu metody connectToDevice().
testSendMessageWhenDisconnected()	Sprawdza, czy metoda sendMessage() nie powoduje błędów ani wyjątków, gdy port nie jest otwarty.
testDisconnectBehavior()	Sprawdza, czy metoda disconnect() zamyka połączenie i aktualizuje stan.

6. Podręcznik użytkownika

Aplikacja została zaprojektowana jako graficzne narzędzie umożliwiające komunikację z mikrokontrolerem ESP32 w trybie szeregowym (UART) lub przez sieć Wi-Fi (TCP). Poniżej przedstawiono opis interfejsu oraz instrukcję obsługi.

6.1.1 Uruchomienie aplikacji

Aby uruchomić aplikację:

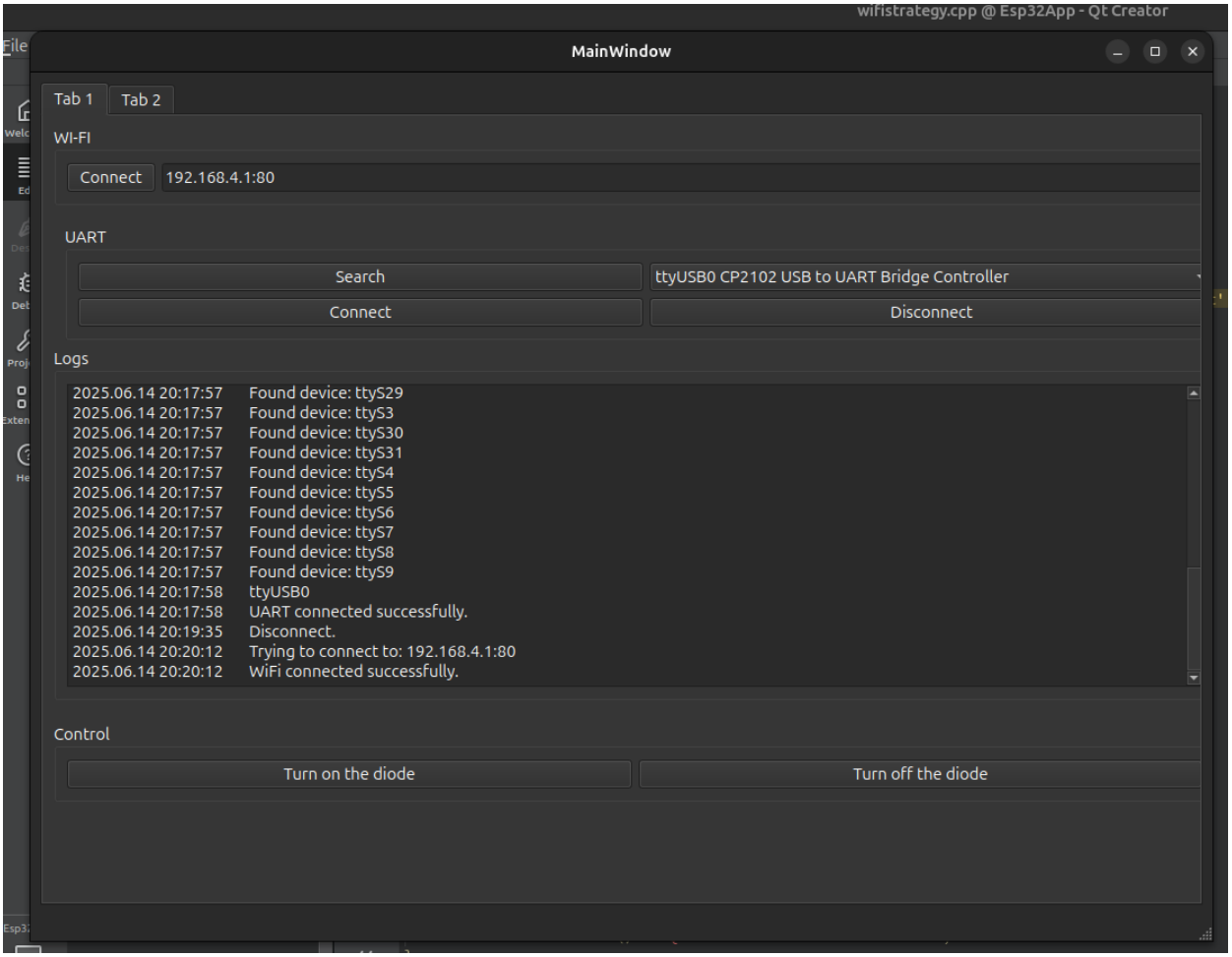
- Upewnij się, że ESP32 jest podłączone do komputera przez USB
- Skonfiguruj i wgraj program Esp32Comm.ino na płytkę
- Uruchom aplikację na komputerze z systemami GNU/Linux (najlepiej Ubuntu)
- W razie potrzeby skorzystaj z komendy do przyznania uprawnień do portu szeregowego: sudo chmod 777 /dev/ttyUSB0

6.1.2 Interfejs użytkownika

Aplikacja składa się z dwóch głównych zakładek:

Zakładka „Sterowanie”

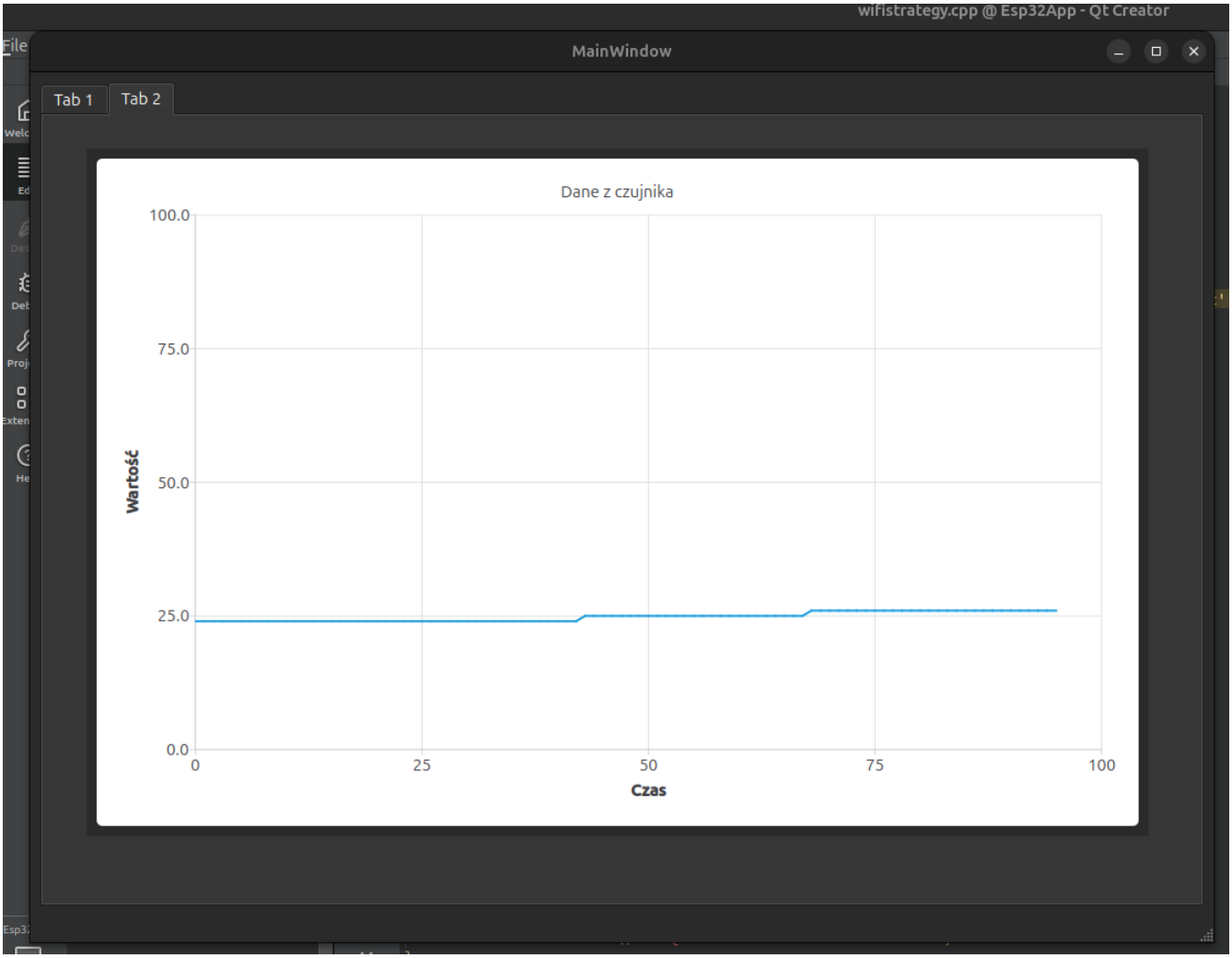
- Lista portów szeregowych – rozwijane menu z dostępnych portów (np. /dev/ttyUSB0).
- Przycisk „Search” – aktualizuje listę wykrytych portów.
- Przycisk „Connect” – nawiązuje połączenie szeregowe.
- Pole IP: PORT i „Connect” – umożliwia wpisanie adresu ESP32 (np. 192.168.4.1:80) i nawiązanie połączenia TCP.
- Przyciski „Turn on the diode” / „Turn off the diode” – wysyłają odpowiednio znak 1 lub 0 do
- Pole logów – pokazuje przebieg komunikacji, m.in. status połączenia i odebrane dane.



Rysunek 4 GUI aplikacji. Zakładka "Sterowanie"

Zakładka „Wykres”

Wykres czasu rzeczywistego z czujnika (np. temperatury), aktualizowany automatycznie po odebraniu danych



Rysunek 5 GUI aplikacji. Zakładka "Wykres"

7. Metodologia rozwoju i utrzymania systemu

7.1.1 Metodologia pracy

Projekt został zrealizowany zgodnie z podejściem iteracyjnym, w którym kolejne funkcjonalności były dodawane i testowane stopniowo. Główne etapy realizacji obejmowały:

1. Analizę wymagań – określenie potrzeb funkcjonalnych, kanałów komunikacji i zakresu projektu.
2. Projektowanie architektury – zastosowanie wzorców projektowych (Strategia, Obserwator), podział aplikacji na komponenty.
3. Implementację – rozwój modułów komunikacyjnych, GUI, logów i wykresu.
4. Testowanie – weryfikacja poprawności działania poszczególnych strategii komunikacji, integracja testów jednostkowych z użyciem QTest.
5. Walidację i integrację – testy końcowe z mikrokontrolerem ESP32 w rzeczywistym środowisku.

Każda iteracja kończyła się kompilacją, uruchomieniem i sprawdzeniem stabilności funkcjonalnej aplikacji.

7.1.2 Utrzymanie i rozwój systemu

Dzięki modularnej strukturze opartej na zasadach Object-Oriented Design, system jest łatwy do utrzymania i rozbudowy. W szczególności:

- Nowe strategie komunikacji (np. Bluetooth, MQTT, WebSocket) można dodać, implementując klasę dziedziczącą po CommunicationStrategy, bez modyfikacji kodu MainWindow.
- Nowe typy danych z czujników można obsłużyć, modyfikując lub rozbudowując logikę parsowania danych w module odbioru.
- Rozszerzenie wykresu o kolejne osie (np. wilgotność, napięcie) jest możliwe przez dodanie dodatkowych serii (QLineSeries) w zakładce wykresu.
- Eksport danych do plików CSV lub JSON może zostać dodany jako nowa funkcja GUI.

System nie wymaga zaawansowanej konserwacji – jedynie aktualizacji zależności Qt w razie zmiany środowiska. W przypadku migracji na inny system operacyjny niż GNU/Linux, niezbędna może być ponowna konfiguracja dostępu do portów COM.

Bibliografia

[1] Mateusz Patyk, Kurs Q, https://forbot.pl/blog/kurs-qt. [dostęp: 15.06.2025]

[2] "Qt Documentation." Qt, https://doc.qt.io. [dostęp: 15.06.2025]

[3] “Strategy”, https://refactoring.guru/design-patterns/strategy. [dostęp: 15.06.2025]

[4] Arduino Dokumentacja, https://docs.arduino.cc [dostęp: 15.06.2025]