

|  |  |  |  |
| --- | --- | --- | --- |
| Projekt zespołowy | | | |
| **Kierunek** | *Informatyka* | **Termin** | *Czwartek 14:15* |
| **Temat** | *Projekt elastycznej aplikacji do zarządzania urządzeniami IoT w oparciu o bibliotekę QT* | **Zgłaszający** | *InterElcom* |
| **Skład grupy** | *Adam Krizar 241276*  *Katarzyna Czajkowska 242079*  *Mateusz Gurski 242089*  *Szymon Cichy 235093*  *Arkadiusz Cichy 236011* | **Nr grupy** | *-* |
| **Prowadzący** | *Dr inż. Jan Nikodem* | **data** | *2 marca 2020* |

Sprawdziła: Katarzyna Czajkowska  
Zatwierdził: Mateusz Gurski

Spis treści

[1. Organizacja pracy 3](#_Toc34924903)

[1.1. Zarządzanie projektem 3](#_Toc34924904)

[1.2. Aplikacja desktopowa 3](#_Toc34924905)

[1.3. Backend 3](#_Toc34924906)

[1.4. Zbieranie informacji 3](#_Toc34924907)

[2. Wymagania 3](#_Toc34924908)

[3. Założenia 3](#_Toc34924909)

[4. Opis zadania 4](#_Toc34924910)

[5. Środowisko 6](#_Toc34924911)

[5.1. Instalacyjne 6](#_Toc34924912)

[5.2. Programistyczne 6](#_Toc34924913)

[6. Sprzęt 6](#_Toc34924914)

[6.1. Jednostki centralne (Mikrokontrolery) 6](#_Toc34924915)

[ ESP82 6](#_Toc34924916)

[ Mikrontroler Arduino Uno/Nano 7](#_Toc34924917)

[6.2. Czujniki 7](#_Toc34924918)

[ Czujnik DHT11 7](#_Toc34924919)

[7. Kosztorys 7](#_Toc34924920)

[8. Plan realizacji 8](#_Toc34924921)

[ Pierwszy punkt kontrolny [19.03] 8](#_Toc34924922)

[ Drugi punkt kontrolny [02.04] 8](#_Toc34924923)

[ Trzeci punkt kontrolny [23.04] 8](#_Toc34924924)

[ Instalacja [07.05] 8](#_Toc34924925)

[ Testy użytkownika [21.05] 8](#_Toc34924926)

[ Oddanie projektu do użytku [04.06] 8](#_Toc34924927)

[ Prezentacja naszych osiągnięć [10.06] 8](#_Toc34924928)

[9. Źródła 8](#_Toc34924929)

# Organizacja pracy

Autor Adam Krizar

## Zarządzanie projektem

Przygotowywanie dokumentacji, reprezentowanie grupy oraz zarządzanie projektem: Adam Krizar

Kontrola dokumentacji – korekta: Katarzyna Czajkowska

Ostateczne zatwierdzenie dokumentacji: Mateusz Gurski

## Aplikacja desktopowa

Projekt interfejsu graficznego, opracowanie portu na system Windows: Adam Krizar, Katarzyna Czajkowska

Obsługa protokołu HTTP: Mateusz Gurski

Obsługa protokołu MQTT: Adam Krizar, Katarzyna Czajkowska

## Backend

Obsługa HTTP po stronie IoT: Mateusz Gurski, Arkadiusz Cichy

Obsługa MQTT: Arkadiusz Cichy, Szymon Cichy

Obsługa funkcjonalności aplikacji (zapisywanie parametrów połączenia, listy urządzeń oraz odczyt tych plików): Adam Krizar, Arkadiusz Cichy, Szymon Cichy

Oprogramowanie modułów Wi-Fi: Arkadiusz Cichy, Szymon Cichy, Mateusz Gurski

## Zbieranie informacji

Znalezienie informacji na temat protokołu MQTT: Adam Krizar, Katarzyna Czajkowska

Znalezienie czujników/urządzeń IoT wspierających wybrane protokoły: Szymon Cichy

Wyszukanie dodatkowych protokołów komunikacji z urządzeniami IoT: Szymon Cichy, Arkadiusz Cichy

# Wymagania

Autor Adam Krizar

Poniższe wymagania są w przedstawione w sposób niezmieniony od zgłoszonego przez firmę InterElecom na stronie: <https://kpz.pwr.edu.pl/projekty-zespolowe/>.

Założeniem projektu jest utworzenie aplikacji działającej na kilku platformach w oparciu o bibliotekę QT i język C++. Jej elastyczność będzie polegała na możliwości zmiany protokołu komunikacji z urządzeniem IoT.

Wymagania, które powinna ona spełniać to:

 Użycie biblioteki QT oraz języka C++

 Stworzenie aplikacji działającej minimum na dwie platformy np. (Linux, Android).

 Stworzenie w aplikacji możliwości wyboru oraz sposobu dodawania nowych protokołów komunikacji z urządzeniem IoT.

 Obsługa w aplikacji minimum dwóch protokołów komunikacji z urządzeniem IoT np. (HTTP, MQTT).

# Założenia

Autor Adam Krizar

Bazując na zgłoszonych wymaganiach opracowaliśmy następujące cele naszego projektu:

- Wsparcie dla dwóch platform: Dla systemu Linux i oraz w mobilnej wersji dla systemu android

- Wykorzystanie biblioteki QT w najnowszej wersji (5.14 ) oraz języka C++

- Zaimplementowane zostanie wsparcie dla dwóch protokołów komunikacji z urządzeniami IoT: HTTP oraz MQTT.

- Obsługa czujników w sieci lokalnej (LAN)

- Obsługa do 80 urządzeń IoT naraz. Wymaga odpowiedniego routera!

- Zbudowanie i zaprogramowanie dwóch czujników w celu zaprezentowania funkcjonalności programu.

- Zaimplementowanie menu pomocy z instrukcją obsługi aplikacji

- Parowanie urządzeń: Możliwość zapisywania parametrów połączenia z urządzeniami w celu szybkiego odbudowania sieci po na przykład: utracie zasilania.

- Możliwość edycji sparowanego połączenia: Zmiana adresu IP urządzenia IoT lub zmiana protokołu komunikacji z urządzeniem.

- Możliwość nadawania nazw podłączanym urządzeniom w celu łatwej identyfikacji.

- Możliwość zapisywania danych do plików co pewien przedział czasu regulowany przez użytkownika: (30 sekund, minuta, 5 minut, 10 minut, godzina)

# Opis zadania

Celem projektu jest zaprojektowanie i zaprogramowanie stabilnej i przejrzystej aplikacji do komunikacji z urządzeniami IoT przy pomocy dwóch różnych protokołów komunikacji. Szczególną uwagę należy zwrócić na projekt interfejsu aplikacji.

Wszystkie opcje muszą być dobrze opisane oraz w jasny sposób przekazywać użytkownikowi do czego służą. Ważne jest także zaimplementowanie systemu pomocy, który w sposób opisowy wytłumaczy użytkownikowi, jak używać funkcji zawartych w aplikacji. Zostaną również przygotowane i zaprogramowane dwa urządzenia testowe, które posłużą do testowania i prezentacji działania aplikacji.

Aplikacja pozwalać będzie na wyświetlanie danych z czujników podłączonych do urządzeń IoT. Dane uzyskiwane będą przy użyciu protokołu komunikacji HTTP lub MQTT, zależnie od tego który z nich jest obsługiwany przez urządzenie IoT od którego chcemy je uzyskać.

W przypadku komunikacji HTTP, aplikacja będzie pełnić role klienta http a urządzenie IoT zaprojektowane w celu przetestowania jej możliwości zaprojektowane zostanie jako serwer http, który na prośby od klientów – urządzeń na których zainstalowana zostanie aplikacja – będzie odsyłał dane z czujników.

W przypadku komunikacji przy użyciu protokołu MQTT, aplikacja, jak i urządzenie IoT będą pełnić rolę klientów MQTT i komunikować się one będą za pośrednictwem MQTT brokera. Plan wdrożenia obu sposobów komunikacji przedstawiony został na poniższych diagramach.

Obraz zawierający zrzut ekranu

Opis wygenerowany automatycznieObraz zawierający zrzut ekranu

Opis wygenerowany automatycznie

Rysunek 2. Plan wdrożenia komunikacji przy użyciu protokołu MQTT

Rysunek 1. Plan wdrożenia komunikacji przy użyciu protokołu HTTP

# Środowisko

## Instalacyjne

Łączność między komputerami na których zainstalowana zostanie aplikacja a urządzeniami IoT będzie odbywać się przez sieć lokalną poprzez łącze przewodowe bądź z użyciem transmisji bezprzewodowej Wi-Fi.

Wymagania sprzętowe dla naszej aplikacji są trudne do precyzyjnego określania na etapie projektowym. Zakładamy jednak, że każdy sprzęt, na którym może działać nowoczesny 64 bitowy system operacyjny (Windows 10, dystrybucje Linuxa z interfejsem graficznym) będzie wystarczający.

## Programistyczne

Do budowy aplikacji wykorzystany zostanie język C++ i framework Qt.

Framework Qt zostanie wykorzystany w najnowszej dostępnej wersji 5.14.1.

Do stworzenia aplikacji desktopowej użyte zostaną narzędzia Qt Creator oraz Qt Designer, które znacząco ułatwią prace nad interfejsem użytkownika i resztą aplikacji dzięki swojej integracji z frameworkiem Qt. Wykorzystanie aplikacji Qt Designer znacząco ułatwia projektowanie strony graficznej aplikacji. Wykorzystanie jej zapewnia dostęp do prostego graficznego interfejsu umożliwiającego umieszczanie elementów interfejsu użytkownika poprzez wybranie ich z listy obiektów wspieranych przez framework Qt, a następnie umieszczenie ich w wybranym miejscu bez konieczności pisania kodu. Dzięki wykorzystaniu opcji layoutów wokół których zbudowane jest wiele aplikacji opartych o Qt, można uzyskać automatyczne skalowanie elementów interfejsu do rozmiarów okna.

Ze względu na konieczność obsłużenia dwóch różnych platform istnieją ograniczenia w wykorzystywaniu bibliotek natywnych dla systemów operacyjnych ze względu na możliwe braki w ich implementacji lub inne działanie na różnych platformach. Framework Qt zapewnia jednak dostęp do wielu gotowych klas, które zadziałają na każdej platformie wspieranej przez Qt bez konieczności przepisywania kodu od początku.

Zapewnia on też klasy, które znacznie ułatwią implementacje obsługi protokołów komunikacji takie jak:

QNetworkAccessManager – klasa, która umożliwia w prosty sposób implementacje protokołu HTTP.

QMqttClient – klasa, która umożliwia obsługę protokołu MQTT.

Kolejne klasy i wykorzystane biblioteki zostaną udokumentowane wraz z rozwojem aplikacji.

Po stronie urządzeń IoT planowane jest wykorzystanie takich narzędzi jak Arduino IDE. Planowane jest użycie takich bibliotek jak MQTT.h czy Ethernet.h, które pozwolą na zaimplementowanie protokołów http i MQTT.

Wybór konkretnych narzędzi i bibliotek programistycznych zostanie uzupełniony gdy uda się wybrać i zabezpieczyć konkretne urządzenia spełniające wymagania projektu.

W celu ułatwienia pracy w grupie wykorzystany zostanie system kontroli wersji. Repozytorium zostanie utworzone na platformie Github.

# Sprzęt

Autor: Adam Krizar

## Jednostki centralne (Mikrokontrolery)

### ESP8266/ESP01

Mikrourządzenie, które dzięki swoim niewielkim rozmiarom i niskim kosztom jest idealne do zastosowania w naszym projekcie. Istnieje wiele SDK, które ułatwiają pisanie programów oraz obsługę peryferiów.

Specyfikacje:

- 80 Mhz CPU

- 32 KiB instruction RAM

- 32 KiB instruction cache RAM

- 80 KiB user-data RAM

- 16 KiB ETS system-data RAM

- IEE 802.11 b/g/n Wi-Fi

- Ilość Pinów I/O: 16

- Koszt: 18,90 zł

Ponadto urządzenie wspiera wybrane przez nas standardy komunikacji.

Dodatkowe informacje dostępne są pod adresem: <https://en.wikipedia.org/wiki/ESP8266>

### Arduino Uno/Nano

Urządzenie pomimo gorszych specyfikacji i braku wbudowanego modułu Wi-Fi jest wspierane przez wielu pasjonatów i posiada rozbudowane biblioteki ułatwiające programowanie.

Specyfikacje:

- 16 Mhz CPU

- 32 KiB pamięci flas

- 2 KiB SRAM

- 1 KiB EEPROM

- Ilośc pinów I/O: 14 dla Arduino Nano, 22 dla Arduino Uno

- Zasilanie: 7-12V

- Koszt Arduino Uno: 92 zł, Arduino Nano 69 zł

### ESP32

Następca ESP8266/ESP01

Specyfikacje:

- Dual/Single Core pracujący z częstotliwością 160/240 MHz

- 520 KiB SRAM

- 448 KiB ROM

- Bluetooth v4.2 BR/EDR and BLE

- Wi-Fi 802.11 b/g/n

- Koszt 38,79 zł

Po przeanalizowaniu parametrów wybranych sprzętów zdecydowaliśmy, że najlepszym w naszym przypadku będzie Mikrokontroler ESP8266. Oferuje najlepszy stosunek jakości do ceny i całkowicie wystarczy do naszych zapotrzebowań. Oba warianty Arduino pomimo łatwości programowania na tej platformie zostały odrzucone ze względu na wysoki koszt zakupu oryginalnych urządzeń. Dodatkowo wymagają one dodatkowego modułu Wi-FI co dodatkowo zwiększa koszt takiego zestawu.

ESP32 pomimo oferowania najlepszych specyfikacji i obsługi aż dwóch protokołów komunikacji został odrzucony ze względu na wyższą cenę niż ESP8266. Dodatkowo wbudowany moduł łączności bluetooth nie jest nam potrzebny w tym projekcie.

## Czujniki

### Czujnik DHT11

Tanie rozwiązane dzięki któremu będziemy mogli odczytywać dane testowe z urządzenia IoT

Specyfikacje:

- Częstotliwość próbkowania max raz na sekundę.

- Odczyt wilgotności z zakresu 20-80% z dokładnością 5%.

- Odczyt temperatury z zakresu 0-50°C z dokładnością 2°C.

- Koszt: 4,90 zł

Czujnik ten został wybrany ze względu na swoją niską cenę. Służy on tylko jako przykład możliwości aplikacji i rodzaj czujnika nie ma tutaj znaczenia w końcowej implementacji projektu.

# Kosztorys

Autor Adam Krizar

Głównym kosztem w realizacji naszego projektu są urządzenia IoT koniecznie do testowania i prezentacji możliwości naszej aplikacji. Potrzebne są nam dwie platformy testowe:

Mikrokontroler ESP8266: <https://allegro.pl/oferta/esp8266-nodemcu-v3-wifi-2-4ghz-ch340-do-arduino-7241549772>, koszt 18,90 zł.

Czujnik DHT11: <https://allegro.pl/oferta/dht11-czujnik-temperatury-i-wilgotnosci-arduino-7487941486>, koszt 4,70 zł

Całkowity koszt w zależności od wybranej podstawki wynosi odpowiednio:

ESP8266: 47,20 zł

# Plan realizacji

Autor Adam Krizar

## Pierwszy punkt kontrolny [19.03]

Implementacja prototypowej wersji aplikacji na system Linux. Zaimplementowanie protokołu http po stronie aplikacji.

## Drugi punkt kontrolny [02.04]

Rozwój aplikacji na system Linux. Przygotowanie pierwszego urządzenia IoT i przetestowanie działania protokołu HTTP. Implementacja protokołu MQTT (bez testów).

## Trzeci punkt kontrolny [23.04]

Przeniesie aplikacji na system android. Przygotowanie drugiego urządzania IoT oraz przetestowanie protokołu MQTT.

## Instalacja [07.05]

## Testy użytkownika [21.05]

Wprowadzenie ewentualnych korekt w projekcie interfejsu użytkownika zgodnie z uwagami użytkownika końcowego.

## Oddanie projektu do użytku [04.06]

## Prezentacja naszych osiągnięć [10.06]

# Źródła

<https://store.arduino.cc/arduino-nano>

<https://www.qt.io/>

<https://store.arduino.cc/arduino-uno-rev3>

<https://learn.adafruit.com/dht>

<https://www.sparkfun.com/datasheets/Components/nRF24L01_prelim_prod_spec_1_2.pdf>

<https://en.wikipedia.org/wiki/ESP32>

<https://en.wikipedia.org/wiki/ESP8266>