Politechnika
Wrocławska

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*

Spis treści

1. Organizacja pracy

1.1. Dokumentacja oraz zatwierdzania prac

Wykonawca: Adam Krizar
Sprawdzenie: Katarzyna Czajkowska

- Korekta rozdziałów (4, 2, 3, 5 – opis propozycji)

- Zatwierdzanie prac (Tylko Adam Krizar)

1.2. Czujniki oraz oprogramowanie urządzeń IoT

Wykonawca: Arkadiusz Cichy
Sprawdzenie: Szymon Cichy

- Zapoznanie się z obsługą, budową oraz możliwościami wybranego czujnika

- Stworzenie oprogramowania obsługującego wybrany czujnik na urządzeniu IoT

1.3. Komunikacja bezprzewodowa

Wykonawca: Mateusz Gurski
Sprawdzenie: Katarzyna Czajkowska

- Zrozumienie TCP/IP, HTTP, MQTT oraz TCP/IP

- Przygotowanie aplikacji w wersji na system android

1.4. Aplikacja desktopowa

Wykonawca: Katarzyna Czajkowska
Sprawdzenie: Arkadiusz Cichy

- Warstwa 3, 5,7 modelu OSI oraz projektowanie podsieci.
- Program do obsługi komunikacji po stronie systemu Linux

2. Opis zadania

Naszym zadaniem jest stworzenie aplikacji, która umożliwia komunikację z urządzeniami IoT zezwalając na zmianę protokołu komunikacji (elastyczność).

Stan początkowy określa jedynie platformy, które mamy wspierać oraz technologie do wykorzystania. Naszym zadaniem jest więc określenie następujących rzeczy:

- W jakiej wersji wykorzystać wymagane narzędzia
- Określić w jakim środowisku oraz w jaki sposób urządzenia IoT będą komunikować się z naszą aplikacją.
- Stworzenie przykładowego układu (Aplikacja, IoT) w celu zaprezentowania możliwości aplikacji.

3. Wymagania

Celem 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).

4. Założenia

Bazując na zgłoszonych wymaganiach opracowaliśmy następujące cele naszego projektu:

- Wymaganie wykorzystania biblioteki Qt: Wykorzystanie Qt w wersji 5.14 w wzwyż – Zapewnia wykorzystanie jak najdokładniejszych rozwiązań oraz gwarantuje dobre działanie na nowych systemach operacyjnych.

- Wymaganie obsługi dwóch platform: Wsparcie dla systemu Linux (ze względu na jego darmowość i łatwość instalacji na różnych urządzeniach) oraz dla systemu Android (obecnie najpopularniejsza platforma na urządzenia mobilne).

- Wymaganie elastycznej aplikacji: Możliwość wyboru używanego protokołu komunikacji oraz przygotowanie możliwości dodania obsługi nowych protokołów)

- Komunikacja z IoT: Aplikacja będzie realizować komunikacje poprzez sieć lokalną, która może odbywać się po kablu lub bezprzewodowo (Zależy od wykorzystanych urządzeń IoT i możliwości komputera na którym zainstalowana jest aplikacja)

- Możliwość obsługi wielu IoT: Projekt aplikacji przewiduje obsługę do 80 urządzeń. Ta liczba zależy od możliwości wybranego routera obsługującego połączenia.

- Przygotowanie dwóch urządzeń IoT (Wykorzystanie gotowych rozwiązań jak mikrokontrolery Arduino i im podobne) w celu prezentacji możliwości aplikacji.

5. Środowisko

5.1. 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 określenia na etapie projektowym. Zakładamy jednak, że każdy sprzęt, na którym może działać nowoczesny system operacyjny (np. Android 8+, dystrybucje Linux tj. Ubuntu, Manjaro) będzie wystarczający.

5.2. Programistyczne

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

Framework Qt zostanie wykorzystany w najnowszej stabilnej wersji (na dzień 12.03.2020 jest to 5.14.1). Jest to zestaw narzędzi które pozwolą na stworzenie różnych interfejsów użytkownika na osobnych platformach, które to interfejsy będą spójne wizualnie oraz będą mogły przystosowywać się do różnic w konkretnych urządzeniach, jak np. dopasowanie elementów do rozmiarów ekranu. Ponadto zastosowanie bibliotek Qt pozwoli przyspieszyć tempo prac poprzez modularność kodu – to znaczy, nie będzie potrzeby przepisywania całego kodu przy przejściu na nową platformę.

Do tworzenia aplikacji desktopowej użyte zostaną narzędzia Qt Creator oraz QT Designer. Użycie ich usprawni utrzymanie aplikacji oraz wprowadzanie zmian w przyszłości. Wykorzystanie tych specjalnych środowisk poprawi jakość oraz obniży czas wykonania aplikacji, ponadto może skutkować niższymi kosztami obsługi w wypadku konieczności wprowadzenia zmian w interfejsie użytkownika.

Kończąc, użycie bibliotek Qt pozwoli na stworzenie kodu aplikacji który w spójny sposób obsługuje nie tylko interfejs użytkownika, lecz także obsługę protokołów komunikacji z urządzeniami.

Po stronie urządzeń IoT kod będzie napisany w języku C++ lub być może, w zależności od bieżących potrzeb, w innym języku jak np. skrypt Lua.

Wybór innych narzędzi programistycznych może nastąpić w trakcie wykonywania projektu i ich lista może zostać uzupełniona w późniejszej dacie.

W celu ułatwienia pracy w grupie wykorzystany zostanie system kontroli wersji. Repozytorium zostanie utworzone na platformie Github. Jest to sposób na centralizację zasobów w projekcie i ułatwi śledzenie zmian i postępu przez nie tylko programistów, lecz także zleceniodawców.

6. Sprzęt

Podstawowe wymagania jakie wybrany przez na sprzęt musi spełniać:

- Możliwość programowania bezpośrednio z komputera.

- Obsługa wybranych przez nas protokołów: HTTP, MQTT

- Obsługa czujnika DHT11

6.1. Jednostki centralne (Mikrokontrolery)

- **ESP8266**

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: 10

- Koszt: 18,90 zł

Ponadto urządzenie wspiera wybrane przez nas standardy komunikacji.

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

- **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

- **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 flash

- 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

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 potrzeb. Oba warianty Arduino pomimo łatwości programowania na tej platformie zostały odrzucone ze względu na wysoki koszt zakupu oryginalnych urządzeń. Ponadto wymagają one dodatkowego modułu Wi-Fi co podnosi 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 w związku z czym nie ma sensu żebyśmy płacili więcej za funkcjonalność, która nie jest nam potrzebna.

6.2. 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.

7. 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

8. Plan realizacji

8.1. Pierwszy punkt kontrolny [19.03]

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

8.2. Drugi punkt kontrolny [02.04]

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

8.3. Trzeci punkt kontrolny [23.04]

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

8.4. Instalacja [07.05]

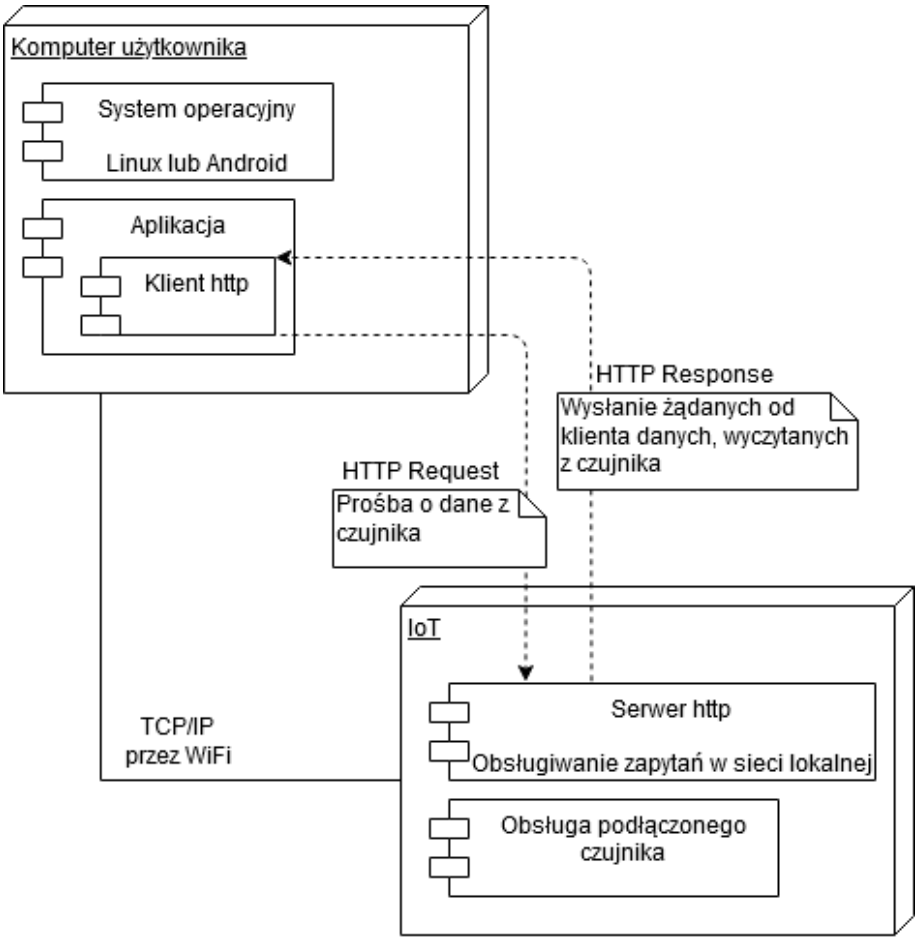
8.5. Testy użytkownika [21.05]

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

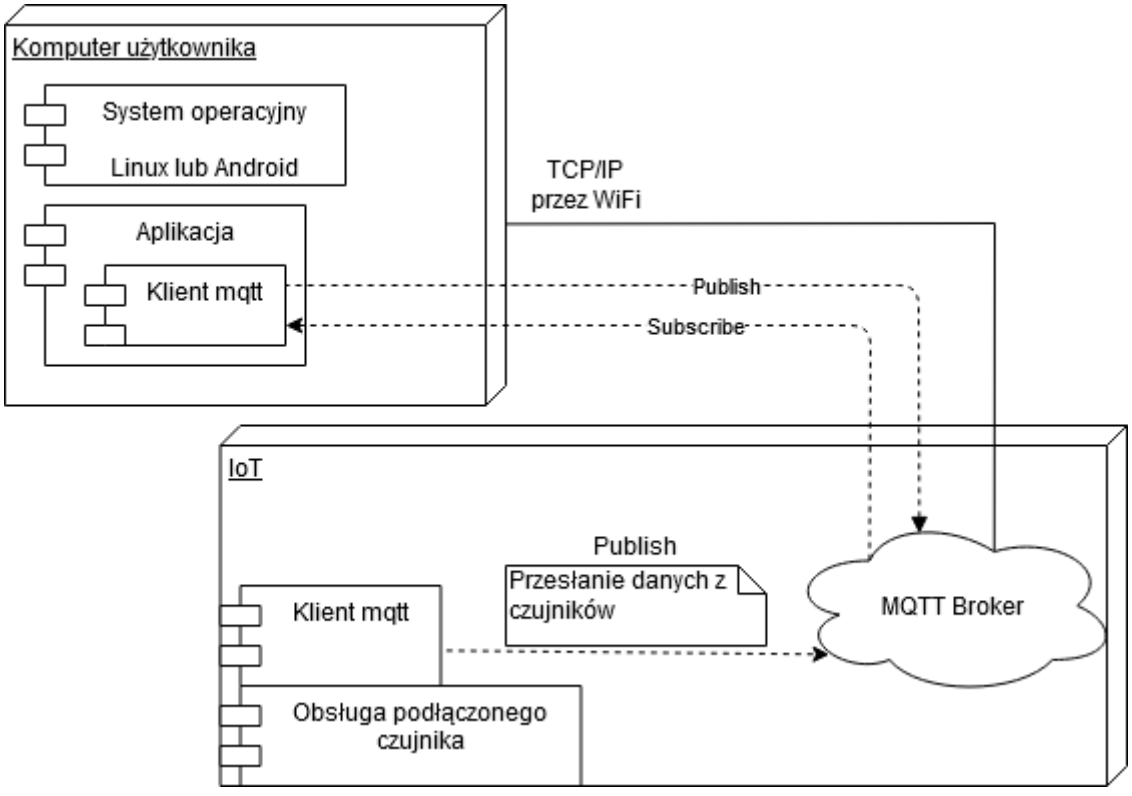
8.6. Oddanie projektu do użytku [04.06]

8.7. Prezentacja naszych osiągnięć [10.06]

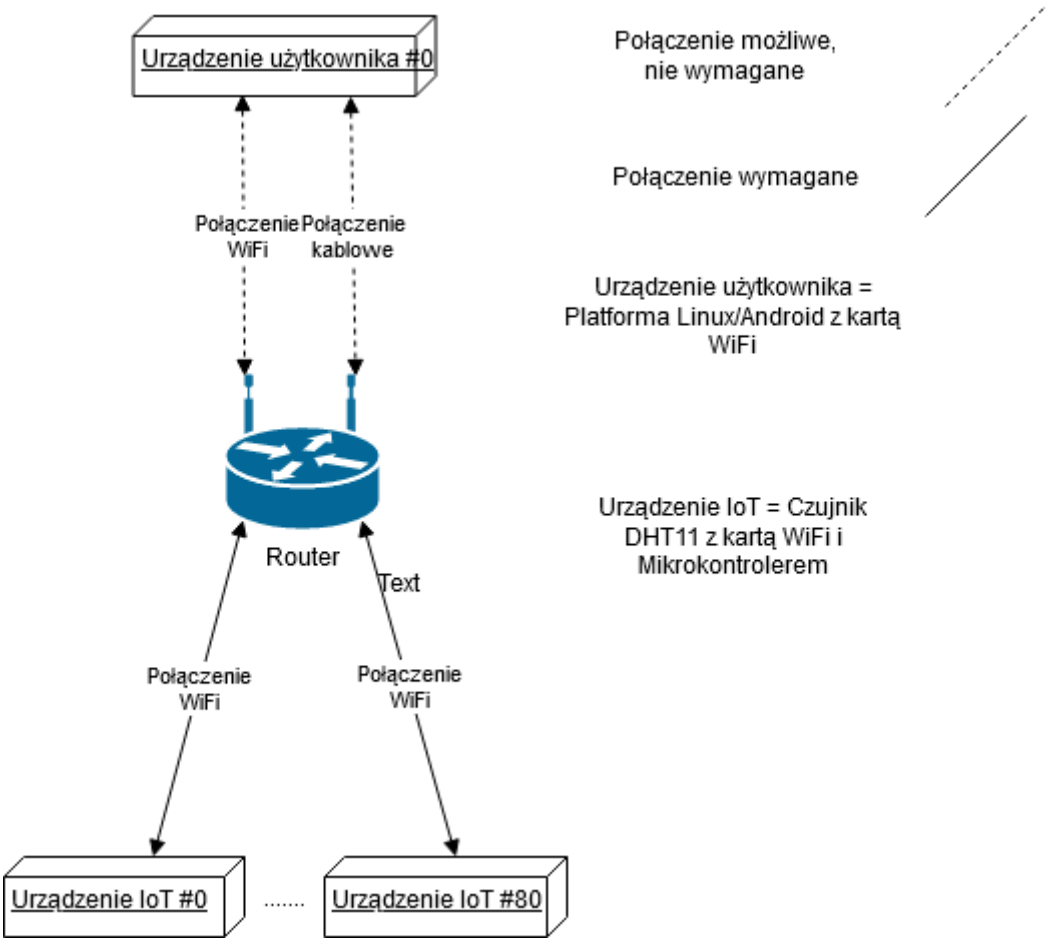
9. Propozycja rozwoju systemu



Rysunek 1. Obsługa protokołu HTTP



Rysunek 1. Obsługa protokołu MQTT



Rysunek 2. Ogólna propozycja użycia aplikacji

10. 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