**PRACA MAGISTERSKA**

**Zastosowanie Raspberry Pi w systemach IoT**

**Wykonał: Zbigniew Sroczyński**

Spis treści

[1. Wstęp 3](#_Toc134575959)

[1.1. Systemy IoT 3](#_Toc134575960)

[2. Cel projektu 5](#_Toc134575961)

[3. Rozważania projektowe 6](#_Toc134575962)

[3.1. Przegląd sieci bezprzewodowych - Sieć WiFi 6](#_Toc134575963)

[3.2. Protokół HTTP 8](#_Toc134575964)

[3.3. Protokół MQTT 10](#_Toc134575965)

[3.4. Przegląd dostępnych serwerów w systemach IoT 11](#_Toc134575966)

[3.5. Serwer Apache2 12](#_Toc134575967)

[3.6. Serwer Mosquitto 13](#_Toc134575968)

[3.7. Framework Django 14](#_Toc134575969)

[3.8. Magistrala 1-wire 14](#_Toc134575970)

[3.9. Raspberry Pi 4 14](#_Toc134575971)

[3.10. ESP8266 14](#_Toc134575972)

[4. Realizacja projektu 14](#_Toc134575973)

[4.1. Rzeczywisty układ 14](#_Toc134575974)

[4.2. Hardware projektu 16](#_Toc134575975)

[4.3. Software – Raspberry Pi 18](#_Toc134575976)

[4.3.1. Serwer HTTP – Django 18](#_Toc134575977)

[4.3.2. Baza danych MySql – dostęp phpMyAdmin 22](#_Toc134575978)

[4.3.3. Serwer MQTT – język Python 23](#_Toc134575979)

[4.4. Software ESP8266 – język Arduino 26](#_Toc134575980)

[5. Przeprowadzone testy 28](#_Toc134575981)

[6. Podsumowanie 28](#_Toc134575982)

[7. Zakończenie 28](#_Toc134575983)

# Wstęp

Internet Rzeczy (IoT) to koncepcja, która opiera się na połączeniu urządzeń ze sobą i z Internetem w celu umożliwienia komunikacji, zbierania danych i sterowania nimi na odległość. Zastosowania IoT są nieograniczone i obejmują wiele różnych sektorów, takich jak przemysł, rolnictwo, opieka zdrowotna, energetyka czy inteligentne domy.

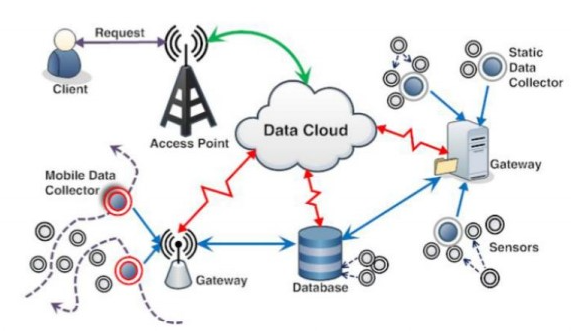
W przemyśle IoT jest wykorzystywane w celu automatyzacji procesów produkcyjnych i monitorowania stanu maszyn. Urządzenia IoT, takie jak czujniki, akcelerometry czy kamery, zbierają dane na temat temperatury, wilgotności, wibracji czy obciążenia maszyn, co umożliwia ich szybkie wykrycie i naprawę przed wystąpieniem awarii. W ten sposób można również zoptymalizować zużycie energii i surowców, co przekłada się na zmniejszenie kosztów produkcji.

W inteligentnych domach IoT umożliwia sterowanie urządzeniami domowymi, takimi jak oświetlenie, klimatyzacja czy zamki drzwi. Dzięki temu można zwiększyć bezpieczeństwo domu i zmniejszyć zużycie energii. W energetyce IoT umożliwia monitorowanie i zarządzanie siecią energetyczną na odległość. Dzięki temu można zoptymalizować zużycie energii, przewidywać awarie i szybko reagować na nie, co zwiększa niezawodność dostaw energii.

Systemy IoT do dziedzina technologii która prężnie się rozwija w różnych dziedzinach naszego życia. Takich systemów będzie coraz więcej, to za sprawą wykorzystania tych systemów w rozwiązaniach inteligentnych domów, inteligentnych miast, pojazdów itd.

## Systemy IoT

Systemy IoT to bardzo rozległe pojęcie które zawiera technologie z zakresu: informatyki, elektroniki, automatyki. Jest to koncepcja, która opiera się na połączeniu urządzeń ze sobą i z Internetem w celu umożliwienia komunikacji, zbierania danych i sterowania nimi na odległość. Nowoczesne systemy zawierają technologie typu sztuczna inteligencja, bazy danych czy umiejętność tworzenia stron internetowych. Wiedza na temat hardware także okaże się bardzo przydatna. To rozległe pojęcie (systemy IoT) wymaga tak naprawdę wielu specjalistów w różnych dziedzinach.



Rysunek 1. Poglądowy rysunek sieci IoT.

Omawiane systemy działają w sieciach bezprzewodowych, takich jak WiFi, Bluetooth czy Zigbee, umożliwiają bezprzewodową komunikację między urządzeniami. Dzięki temu możliwe jest zbieranie danych z różnych czujników i urządzeń oraz sterowanie nimi na odległość. Czujniki są kluczowymi elementami systemów IoT. Dzięki nim możliwe jest zbieranie danych na temat temperatury, wilgotności, wibracji czy obciążenia maszyn, co umożliwia ich szybkie wykrycie i naprawę przed wystąpieniem awarii.

Sztuczna inteligencja (AI) jest ważnym elementem systemów IoT, ponieważ umożliwia analizę dużych ilości danych i automatyczne podejmowanie decyzji. AI jest wykorzystywana w systemach monitoringu, zarządzania energią czy w inteligentnych domach, gdzie na podstawie analizy zachowań mieszkańców można dostosować ustawienia urządzeń domowych.

Automatyzacja to kolejna składowa technologii w systemach IoT. Systemy IoT umożliwiają automatyzację różnych procesów, co pozwala na zmniejszenie kosztów i zwiększenie efektywności. Przykładowo, w przemyśle IoT umożliwia automatyzację procesów produkcyjnych, co pozwala na zmniejszenie kosztów produkcji i zwiększenie szybkości produkcji.

Bazy danych to kolejna składowa technologii w systemach IoT. Dzięki bazom danych możliwe jest gromadzenie i przetwarzanie dużych ilości danych pochodzących z różnych źródeł. Bazy danych umożliwiają również analizę danych i podejmowanie na ich podstawie decyzji, co jest kluczowe w systemach IoT.

Wszystkie te składowe technologii w systemach IoT działają razem, umożliwiając efektywne zbieranie i przetwarzanie danych oraz sterowanie urządzeniami na odległość. Systemy IoT mają duży potencjał w poprawie efektywności i jakości życia, a ich wykorzystanie będzie coraz bardziej popularne w przyszłości.

# Cel projektu

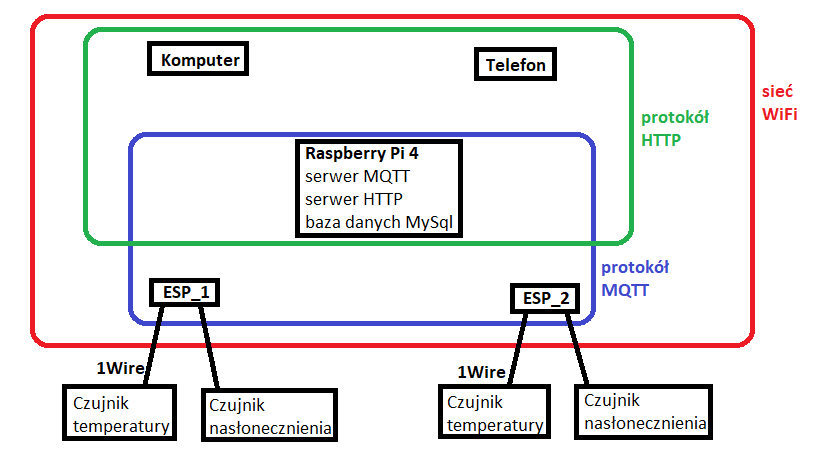
Celem pracy magisterskiej było utworzenie rozwiązania dla inteligentnego domu. Docelowo ma to być system sterowania roletami okiennymi w domu. Dane z czujników (czujnik temperatury, czujnik natężenia światła) będą wysyłane przez dwa urządzenia ESP8266 do Raspberry Pi 4. Jedno ESP na zewnątrz okna, drugie w środku pokoju. Komunikacja między urządzeniami przez sieć WiFi, protokół MQTT, broker Mosquitto.

Na Raspberry Pi znajduje się serwer WWW Apache2 i Django, protokół HTTP. Na serwerze Apache2 znajduje się baza danych MySql. Na serwerze Django znajduje się serwis internetowy wykonany we frameworku Django. Dane z czujników odbierane są przez Raspberry Pi wykorzystując protokół MQTT, następnie wysyłane są do bazy danych znajdującej się na serwerze HTTP.

Utworzona jest strona internetowa na serwerze Django, która pobiera dane z bazy danych. Na stronie zamieszczone są wykresy temperatury i nasłonecznienia, przedstawia dane w przystępnej formie dla użytkownika. Z poziomu dowolnego urządzenia (komputer, telefon) połączony do lokalnej sieci WiFi w której znajduje się Raspberry Pi 4, mamy dostęp do strony internetowej.

**Składowe elementy projektu:**

* **Wykorzystane urządzenia:** Raspberry Pi 4, ESP8266
* **Wykorzystana sieć:** WiFi
* **Wykorzystane protokoły sieciowe:** MQTT, HTTP
* **Utworzone serwery:** Django, Apache2, Mosquitto
* **Wykorzystana baza danych:** MySql
* **Wykorzystane czujniki:** DS18B20, dzielnik napięcia z fotorezystorem
* **Wykorzystana magistrala szeregowa:** 1Wire
* **Wykorzystane języki programowania:** Python, Arduino, SQL, PHP, HTML, CSS, JS



Rysunek 2. Schemat poglądowy utworzonego projektu.

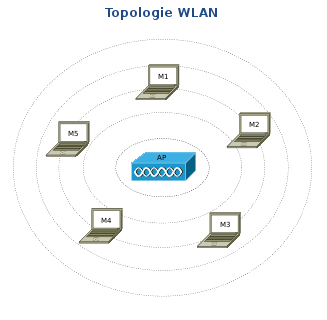
# Rozważania projektowe

Podczas realizacji projektu należało wybrać urządzenia które będą odpowiednie. Należało dokonać przeglądu istniejących rozwiązań na rynku. Dokonano wyboru sieci, protokołów, serwerów. W dalszej części rozdziału zostaną przedstawione rozważania projektowe.

## Przegląd sieci bezprzewodowych - Sieć WiFi

Początkowo dokonano przeglądu dostępnych rozwiązań bezprzewodowych. Istnieje kilka rodzajów sieci bezprzewodowej, z których najpopularniejsze to:

1. **Wi-Fi** - sieć bezprzewodowa wykorzystująca standard IEEE 802.11, która pozwala na przesyłanie danych w zakresie od kilku do kilkudziesięciu metrów od punktu dostępu do sieci (AP).
2. **Bluetooth** - technologia bezprzewodowa wykorzystywana głównie do połączenia urządzeń mobilnych, takich jak smartfony, słuchawki, głośniki czy też klawiatury, na krótkich dystansach, zazwyczaj do 10 metrów.
3. **ZigBee** - niskonapięciowa sieć bezprzewodowa wykorzystywana do komunikacji w systemach automatyki budynkowej, przemysłowej i kontrolno-pomiarowej.
4. **NFC (Near Field Communication)** - technologia bezprzewodowa stosowana do przesyłania danych na krótkich dystansach, zazwyczaj do 10 centymetrów, np. do przeprowadzania płatności zbliżeniowych.
5. **LTE (Long-Term Evolution)** - technologia bezprzewodowa wykorzystywana w sieciach telefonii komórkowej, która pozwala na szybką transmisję danych na duże odległości.
6. **WiMAX** - bezprzewodowa technologia transmisji danych na długie dystanse, wykorzystywana głównie do dostępu do internetu na obszarach wiejskich lub w miejscach, gdzie brakuje infrastruktury kablowej.
7. **LoRa (Long Range)** - technologia bezprzewodowa, która umożliwia transmisję danych na bardzo duże odległości, przy jednoczesnym niskim poborze energii, co jest przydatne np. w systemach IoT (Internet of Things).



Rysunek 3. Działanie sieci WiFi [7].

Sieć WiFi jest jednym z najpopularniejszych rozwiązań sieci bezprzewodowych, które znajduje zastosowanie w różnych projektach, w tym także w projektach związanych z Internetem rzeczy (IoT). W tym wypracowaniu przedstawię argumenty, które skłoniły do wyboru sieci WiFi z dostępnych rozwiązań w kontekście projektu IoT.

Kolejnym ważnym argumentem przemawiającym na korzyść sieci WiFi w kontekście projektu IoT jest jej łatwość w instalacji i konfiguracji. Wiele urządzeń IoT posiada już wbudowany moduł WiFi, co sprawia, że połączenie z siecią jest proste i wymaga jedynie podania hasła do sieci. Dzięki temu możliwe jest szybkie wdrożenie systemu IoT bez konieczności stosowania skomplikowanych konfiguracji i dodatkowych urządzeń.

Kolejnym argumentem na korzyść sieci WiFi w kontekście projektu IoT jest jej uniwersalność. Sieć WiFi obsługuje wiele protokołów i standardów, co umożliwia korzystanie z różnych urządzeń IoT bez konieczności zmiany sposobu połączenia. Dzięki temu, system IoT oparty na sieci WiFi może korzystać z różnych sensorów, modułów bezprzewodowych czy innych urządzeń, bez konieczności stosowania różnych sposobów połączenia.

**Podsumowując:** Na podstawie dostępnych rozwiązań, zdecydowano wybrać się sieć WiFi do mojego projektu. Sieć WiFi jest często i powszechnie wykorzystywana. Jest też uniwersalna, bo z powodzeniem będą działać na niej projekty IoT ale także serwisy internetowe lub bazodanowe serwery.

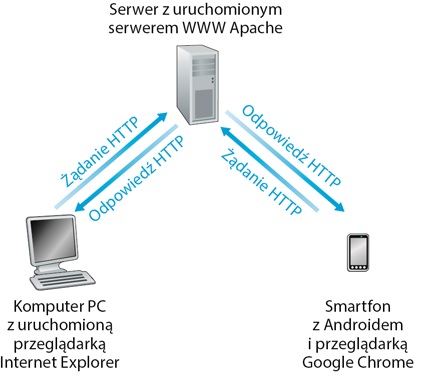
**Podstawowe parametry sieci WiFi:**

1. **Przepustowość**: Przepustowość określa maksymalną ilość danych, które mogą być przesłane przez sieć w ciągu jednostki czasu. Współczesne standardy sieci WiFi, takie jak 802.11ac i 802.11ax, oferują przepustowość do kilku gigabitów na sekundę.
2. **Częstotliwość**: Sieć WiFi działa na różnych częstotliwościach, zależnie od zastosowania. Najpopularniejsze częstotliwości to 2,4 GHz i 5 GHz. Częstotliwość 2,4 GHz jest popularniejsza ze względu na większy zasięg, ale oferuje niższą przepustowość niż częstotliwość 5 GHz.
3. **Zasięg**: Zasięg sieci WiFi zależy od wielu czynników, takich jak siła sygnału, przeszkody w środowisku, rodzaj anteny itp. Współczesne standardy sieci WiFi oferują zasięg do kilkudziesięciu metrów.
4. **Bezpieczeństwo**: Bezpieczeństwo sieci WiFi jest kluczowe ze względu na to, że jest ona bezprzewodowa i potencjalnie podatna na ataki. Współczesne standardy sieci WiFi, takie jak WPA2, oferują wysoki poziom bezpieczeństwa dzięki szyfrowaniu danych i autentykacji.
5. **Przepływność**: Przepływność sieci WiFi zależy od ilości użytkowników korzystających z sieci jednocześnie. Im więcej użytkowników, tym mniejsza przepływność. Przepływność może być również ograniczona przez sieć i urządzenia w niej działające.
6. **Latencja**: Latencja to czas opóźnienia między wysłaniem danych z jednego urządzenia a odbiorem ich przez drugie urządzenie. Latencja w sieci WiFi zależy od wielu czynników, takich jak przepustowość, zasięg i obciążenie sieci.
7. **Liczba** **kanałów**: Sieć WiFi działa na różnych kanałach, które dzielą dostępną przepustowość. Im więcej kanałów, tym więcej urządzeń może korzystać z sieci jednocześnie, co zwiększa przepływność.

## Protokół HTTP

Dokonano przeglądu dostępnych protokołów komunikacyjnych dostępnych w sieci WIFi:

1. **MQTT (Message Queuing Telemetry Transport):** Protokół ten jest często stosowany w urządzeniach IoT w sieciach WiFi, ze względu na swoją lekkość i niezawodność. MQTT jest protokołem opartym na subskrypcji i publikacji, który pozwala na szybkie i niezawodne przesyłanie danych między urządzeniami w sieciach WiFi.
2. **CoAP (Constrained Application Protocol):** Protokół ten jest opracowany z myślą o urządzeniach z ograniczoną mocą obliczeniową i niskim zużyciem energii, ale jest też stosowany w sieciach WiFi. CoAP wykorzystuje protokół UDP (User Datagram Protocol) i zapewnia niskie opóźnienia i niskie zużycie energii.
3. **HTTP (Hypertext Transfer Protocol):** Protokół ten jest dobrze znany i powszechnie stosowany w internecie, ale jest też wykorzystywany w sieciach WiFi. W sieciach IoT w sieciach WiFi HTTP może być wykorzystywany do komunikacji między urządzeniami a aplikacjami internetowymi.
4. **WebSockets:** Protokół ten umożliwia dwukierunkową komunikację między urządzeniami i aplikacjami w czasie rzeczywistym w sieciach WiFi. WebSockets wykorzystują protokół HTTP, ale umożliwiają trwałe połączenie między urządzeniem a serwerem.
5. **Wi-Fi Direct:** Protokół ten umożliwia bezpośrednie połączenie między dwoma urządzeniami w sieci WiFi bez potrzeby korzystania z punktu dostępowego. Wi-Fi Direct może być stosowany w urządzeniach IoT do bezpośredniej komunikacji między nimi.



Rysunek 4. Model protokołu HTTP [8]

HTTP, czyli Hypertext Transfer Protocol, to jeden z najczęściej używanych protokołów do przesyłania danych w Internecie. Jest on oparty na modelu klient-serwer, co oznacza, że dane są przesyłane między klientem (np. przeglądarką internetową) a serwerem (np. serwerem Apache2). W projekcie dotyczącym IoT, gdzie dane są przesyłane między urządzeniami w sieci lokalnej, protokół HTTP może być użyty do przesyłania danych z urządzeń IoT do serwera Apache2 i bazy danych MySql.

Przede wszystkim, protokół HTTP jest bardzo popularny i powszechnie stosowany w Internecie. Wiele aplikacji internetowych, takich jak strony internetowe, serwisy społecznościowe, sklepy internetowe i aplikacje mobilne, korzysta z protokołu HTTP. Dlatego wykorzystanie HTTP w projekcie IoT może ułatwić integrację z innymi systemami i narzędziami.

Kolejnym powodem, dla którego protokół HTTP jest dobrym wyborem w projekcie IoT, jest to, że jest on bardzo prosty w użyciu i łatwy do zrozumienia. Dzięki temu programiści mogą szybko nauczyć się korzystać z tego protokołu i łatwo wdrożyć go w swoich projektach.

Protokół HTTP jest również wysoce konfigurowalny i elastyczny, co oznacza, że można go dostosować do różnych wymagań i potrzeb. Na przykład, w projekcie IoT można skonfigurować serwer Apache2 tak, aby działał jako serwer HTTP i obsługiwał żądania HTTP przychodzące z urządzeń IoT.

W końcu, protokół HTTP oferuje wiele narzędzi i bibliotek, które ułatwiają jego stosowanie w projektach. Istnieją różne biblioteki klientów HTTP, takie jak Requests w języku Python, które ułatwiają wysyłanie żądań HTTP i odbieranie odpowiedzi. Istnieją również biblioteki serwerów HTTP, takie jak Flask i Django w języku Python, które ułatwiają tworzenie serwerów HTTP i obsługę żądań HTTP.

Podsumowując, protokół HTTP jest dobrym wyborem w projekcie IoT, w którym dane są przesyłane między urządzeniami IoT a serwerem Apache2 i bazą danych MySql. Jest popularny, prosty w użyciu, elastyczny i oferuje wiele narzędzi i bibliotek, które ułatwiają jego stosowanie w projektach.

**Podsumowując:** Na podstawie ww wymienionych rozwiązań, zdecydowano się na wykorzystanie protokołów HTTP i MQTT. Protokół HTTP będzie wykorzystywany do obsługi połączenia klient-serwer. Klientem będzie urządzenie dostępne w lokalnej sieci WiFi (telefon, komputer itp.). Serwerem będzie serwer Apache2 utworzony na Raspberry Pi. Na serwerze będą znajdować się strony internetowe oraz baza danych MySql.

**Parametry protokołu HTTP:**

1. Metoda (ang. method): określa rodzaj żądania, jakie zostanie przesłane do serwera. Najczęściej stosowanymi metodami są GET i POST.
2. Nagłówki (ang. headers): zawierają dodatkowe informacje na temat żądania, takie jak typ MIME, rozmiar pliku, dane uwierzytelniające itp.
3. Ciało (ang. body): opcjonalna część żądania, w której przesyłane są dane.
4. Kod odpowiedzi (ang. status code): informuje o tym, czy żądanie zostało pomyślnie wykonane czy też nie, i określa typ błędu, jeśli taki wystąpił.
5. Zawartość odpowiedzi (ang. response body): dane przesłane przez serwer w odpowiedzi na żądanie.
6. Wersja protokołu (ang. protocol version): wskazuje, jakiej wersji protokołu HTTP używa żądanie i odpowiedź.

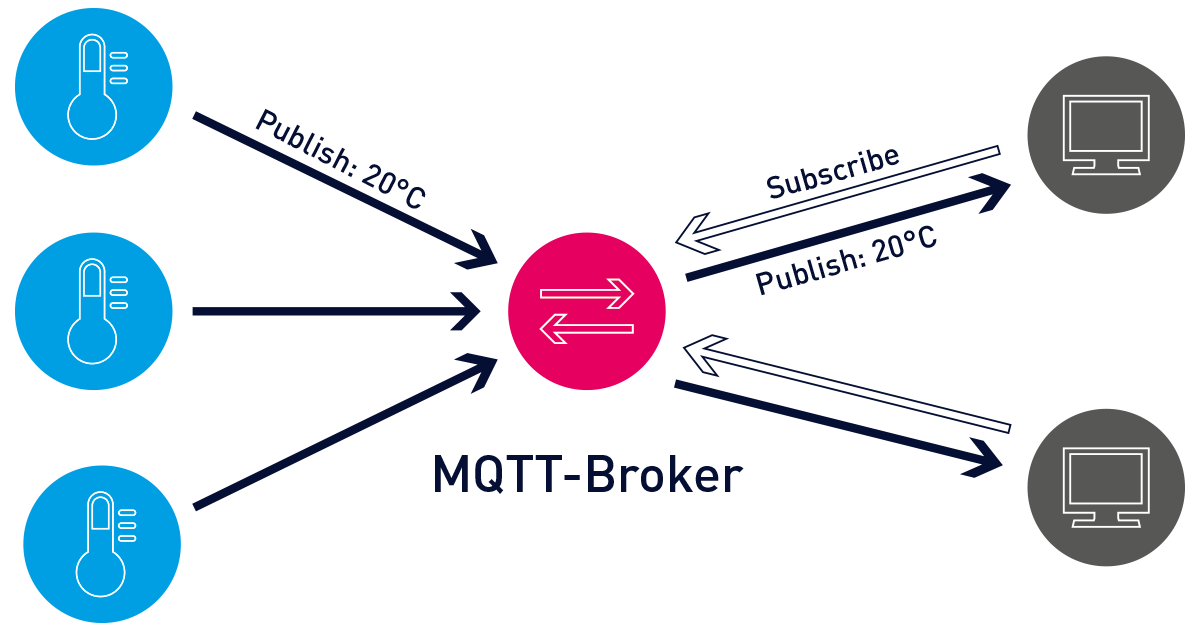
## Protokół MQTT

Protokół MQTT (Message Queue Telemetry Transport) to protokół komunikacyjny, który jest stosowany w sieciach IoT do przesyłania danych między urządzeniami. Główną zaletą protokołu MQTT jest jego lekkość i szybkość, co sprawia, że jest idealnym wyborem do przesyłania danych w czasie rzeczywistym.

W przypadku projektu IoT, w którym wykorzystywane są urządzenia ESP8266 i Raspberry Pi 4, protokół MQTT był dobrym wyborem ze względu na jego niskie wymagania dotyczące zasobów sprzętowych, które są szczególnie ważne w przypadku urządzeń mikroprocesorowych takich jak ESP8266. Protokół ten działa na niskim poziomie i nie wymaga dużych ilości pamięci i mocy obliczeniowej, co pozwala na szybsze i bardziej wydajne przesyłanie danych.

W projekcie IoT, w którym przesyłane są dane między urządzeniami, ważne jest również zapewnienie bezpieczeństwa przesyłanych informacji. Protokół MQTT oferuje kilka możliwości w zakresie zabezpieczeń, w tym szyfrowanie SSL/TLS, uwierzytelnienie użytkownika i hasła oraz kontrole dostępu na poziomie użytkownika. To sprawia, że protokół ten jest idealnym wyborem do zastosowań wymagających wysokiego poziomu bezpieczeństwa, takich jak w przypadku IoT.

Oprócz tego, protokół MQTT oferuje wiele funkcjonalności, które są przydatne w przypadku przesyłania danych w IoT, takich jak kontrola jakości usługi (QoS), co zapewnia dostarczenie danych w określonym czasie i z określoną jakością, a także możliwość publikowania i subskrybowania tematów. To wszystko umożliwia bardziej elastyczne zarządzanie przesyłanymi danymi, co jest niezbędne w projektach IoT.



Rysunek 5. Model protokołu MQTT [9].

**Podsumowując:** wybór protokołu MQTT jako rozwiązania do przesyłania danych między urządzeniami ESP8266 i Raspberry Pi 4 w projekcie IoT jest uzasadniony ze względu na jego lekkość, szybkość i niskie wymagania dotyczące zasobów sprzętowych. Protokół ten oferuje również wysoki poziom bezpieczeństwa oraz wiele funkcjonalności, które są przydatne w przypadku przesyłania danych w IoT.

## Przegląd dostępnych serwerów w systemach IoT

Wśród najczęściej wykorzystywanych serwerach w systemach IoT, możemy wymienić:

1. **Apache2** - to popularny serwer HTTP, który umożliwia udostępnianie treści internetowych, takich jak strony internetowe i pliki. W projektach IoT Apache2 może być wykorzystywany jako serwer internetowy do wyświetlania interfejsów użytkownika lub do udostępniania plików konfiguracyjnych dla urządzeń.
2. **Mosquitto** - to broker MQTT, który umożliwia komunikację między urządzeniami IoT. MQTT jest lekkim i efektywnym protokołem komunikacyjnym, który pozwala na przesyłanie danych w czasie rzeczywistym między urządzeniami IoT. Mosquitto może być wykorzystywany jako broker MQTT do przesyłania danych między urządzeniami.
3. **Node-RED** - to oprogramowanie, które umożliwia łatwe tworzenie aplikacji IoT poprzez wizualny interfejs graficzny. Node-RED można wykorzystać do tworzenia przepływów danych między urządzeniami, integracji z różnymi serwerami i bazami danych, jak również do przetwarzania i analizy danych.
4. **InfluxDB** - to baza danych specjalnie zaprojektowana do przechowywania i analizowania danych związanych z IoT. InfluxDB oferuje efektywne przechowywanie danych czasowych, co jest często wymagane w projektach IoT, które generują dużą ilość danych.
5. **Grafana** - to oprogramowanie, które umożliwia wizualizację danych zebranych z różnych źródeł, w tym z bazy danych InfluxDB. Grafana pozwala na tworzenie interaktywnych wykresów i paneli, które umożliwiają łatwe śledzenie i analizowanie danych związanych z IoT.

## Serwer Apache2



Rysunek 6. Główne logo serweru Apache2 [10].

Wykorzystanie serwera Apache2 w projekcie IoT jest jednym z najlepszych rozwiązań ze względu na jego uniwersalność i wszechstronność. Apache2 to popularny serwer HTTP, który jest darmowy i open-source. Jest on dostępny na wielu platformach, takich jak Linux, Unix, Windows i inne. Jego popularność wynika z tego, że jest łatwy w instalacji i konfiguracji, a także ze względu na jego wydajność, skalowalność i stabilność.

Serwer Apache2 jest często stosowany w projektach IoT do zbierania danych w bazie danych MySQL oraz wyświetlania informacji na stronach internetowych. Jest to możliwe dzięki modułom, które można zainstalować na serwerze Apache2, takim jak mod\_php czy mod\_perl. Te moduły pozwalają na uruchomienie skryptów PHP i Perl, które są często wykorzystywane w projektach IoT.

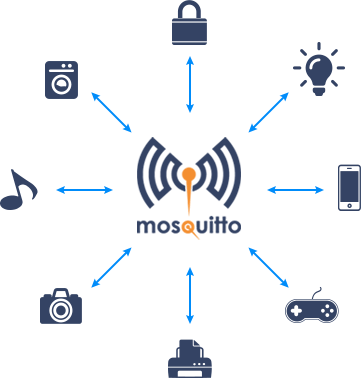
Ponadto, serwer Apache2 oferuje wiele funkcjonalności, takich jak wirtualne hosty, uwierzytelnianie, obsługę protokołu SSL i wiele innych. Pozwala to na dostosowanie serwera do specyficznych potrzeb projektu IoT.

Ważnym atutem Apache2 jest również bogata dokumentacja, która ułatwia korzystanie z serwera dla początkujących i zaawansowanych użytkowników. Ponadto, istnieje wiele społeczności i forów, gdzie użytkownicy mogą uzyskać pomoc i porady w razie problemów z serwerem.

Wydajność serwera Apache2 jest również znana ze względu na to, że może obsługiwać wiele jednoczesnych połączeń. Dzięki temu serwer jest w stanie obsłużyć wiele urządzeń IoT, które przesyłają dane jednocześnie.

**Podsumowując:** wykorzystanie serwera Apache2 w projekcie IoT jest jednym z najlepszych rozwiązań ze względu na jego uniwersalność, łatwość konfiguracji, wydajność i skalowalność. Ponadto, bogata dokumentacja i społeczność użytkowników Apache2 czynią go łatwym w użyciu nawet dla początkujących użytkowników.

## Serwer Mosquitto



Rysunek 7. Główne logo serweru Mosquitto [11].

Wykorzystanie serwera Mosquitto w projekcie IoT związane z zbieraniem danych z urządzeń ESP8266 jest jednym z najlepszych rozwiązań ze względu na jego specyficzne cechy i zalety. Mosquitto jest niewielkim i wydajnym serwerem, który obsługuje protokół MQTT, co czyni go idealnym rozwiązaniem dla projektów IoT. Poniżej przedstawiam argumenty za wykorzystaniem serwera Mosquitto.

Po pierwsze, Mosquitto jest łatwy w instalacji i konfiguracji, dzięki czemu jego uruchomienie nie zajmuje wiele czasu. Ponadto, jest to serwer open source, co oznacza, że kod jest dostępny publicznie, a jego modyfikacja i rozwój są możliwe dla każdego.

Po drugie, Mosquitto jest niskowydajnym serwerem, co oznacza, że zużywa niewiele zasobów systemowych, co jest szczególnie ważne dla projektów IoT, gdzie każdy bit mocy procesora i pamięci jest cenny.

Po trzecie, Mosquitto obsługuje protokół MQTT, który jest uznawany za jeden z najlepszych protokołów dla IoT. MQTT jest protokołem komunikacyjnym, który zapewnia efektywną wymianę informacji między urządzeniami IoT i serwerami. Dzięki temu Mosquitto może pośredniczyć w wymianie danych między urządzeniami ESP8266 a serwerem Apache2 i MySQL.

Wreszcie, Mosquitto oferuje zaawansowane funkcje, takie jak szyfrowanie SSL / TLS, uwierzytelnienie użytkowników i kontroli dostępu, które są ważne w przypadku projektów IoT, w których bezpieczeństwo i prywatność są kluczowe.

**Podsumowując:** Mosquitto jest najlepszym rozwiązaniem dla projektów IoT związanych z zbieraniem danych z urządzeń ESP8266 i wymianą ich z serwerem Apache2 i MySQL. Jego zalety, takie jak łatwość instalacji i konfiguracji, niskie wymagania dotyczące zasobów, obsługa protokołu MQTT oraz zaawansowane funkcje bezpieczeństwa, czynią go idealnym wyborem dla projektów IoT.

## Framework Django

Django to popularny open-sourceowy framework dla języka Python, służący do tworzenia aplikacji webowych. Django pozwala na szybkie i łatwe tworzenie skalowalnych aplikacji webowych dzięki wykorzystaniu wzorca projektowego Model-View-Controller (MVC).

Django dostarcza wiele wbudowanych funkcjonalności, takich jak autoryzacja użytkowników, zarządzanie sesjami, obsługa plików statycznych i wiele innych, dzięki czemu programiści mogą skupić się na implementacji logiki biznesowej aplikacji.

Wraz z frameworkiem Django dostarczany jest wbudowany serwer WWW, który umożliwia szybkie uruchomienie aplikacji na lokalnym komputerze w czasie rozwoju. Serwer ten może być wykorzystany jako środowisko deweloperskie. Jednak do hostowania aplikacji w środowisku produkcyjnym, zalecane jest wykorzystanie dedykowanych serwerów WWW, takich jak Apache, Nginx lub Gunicorn.

Django może być również wykorzystywany w sieciach IoT, na przykład do tworzenia aplikacji monitorujących lub sterujących urządzeniami. Dzięki integracji z popularnymi bibliotekami Pythona takimi jak NumPy, Pandas czy Scikit-learn, Django umożliwia analizę danych i wizualizację wyników, co może być bardzo przydatne w przypadku aplikacji IoT.

## Magistrala 1-wire

Ds.

## Raspberry Pi 4

Ds.

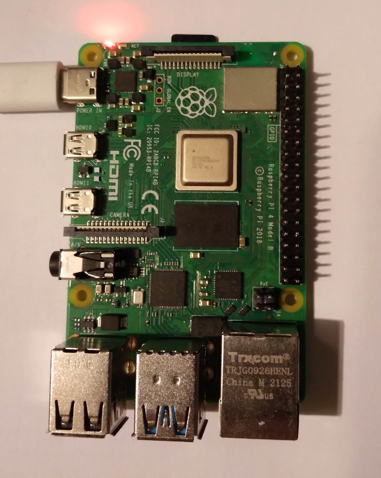
## ESP8266

fd

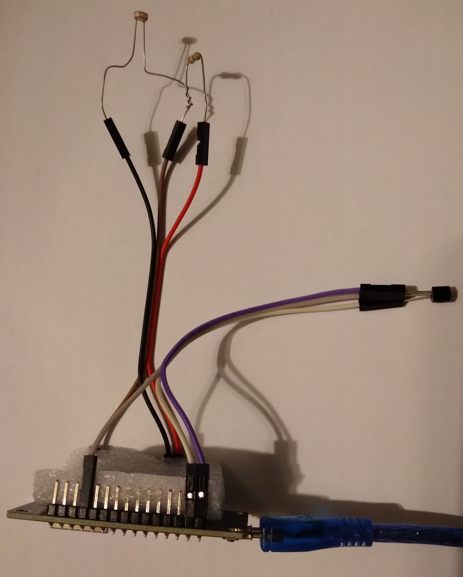
# Realizacja projektu

## Rzeczywisty układ

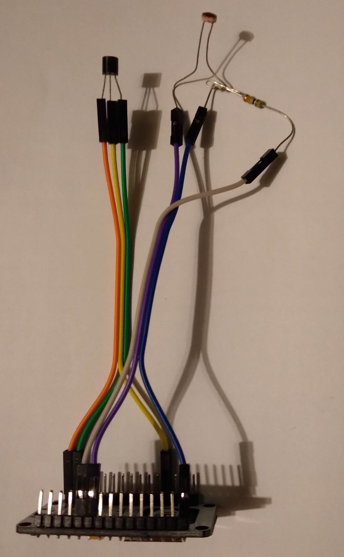
Poniżej zamieszczone zostały fotografie przedstawiające wykorzystane urządzenia w projekcie. Było to Raspberry Pi 4 znajdujące się w pokoju, dwa ESP8266 jedno w pokoju, drugie za oknem.



Rysunek 8. Raspberry Pi 4.



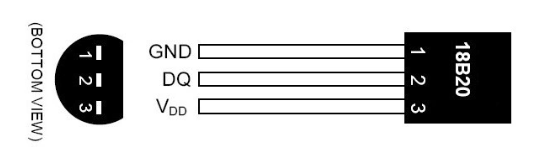
Rysunek 9. ESP8266 nr 1 oraz czujniki.



Rysunek 10. ESP8266 nr 2 oraz czujniki.

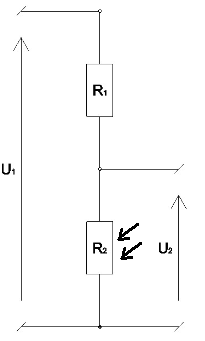
## Hardware projektu

Wykorzystane czujniki to termometr cyfrowy DS18B20 oraz fotorezystor w układzie dzielnika napięcia.



Rysunek 11. Termometr cyfrowy.

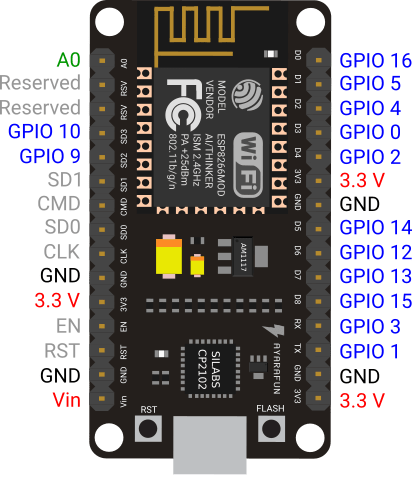
Powyższy czujnik połączono z układem ESP8266. GND podłączono do masy, VDD podłączono do zasilania 3,3V. Wyprowadzenie DQ połączono z pinem GPIO0 układu ESP. Do komunikacji wykorzystano jeden przewód.



Rysunek 12. Fotorezystor w układzie dzielnika napięcia.

W powyższym układzie zastosowano rezystor R1 o wartości 560kΩ. R2 to fotorezystor o zmniejszającej się rezystancji. Z powyższego układu wynika że wraz ze wzrostem natężenia światła maleje wartość napięcia U2. Napięcie z dzielnika podawane jest na pin A0 układu ESP. Jest to przetwornik analogowo-cyfrowy.

Wykorzystane platformy sprzętowe to ESP8266 i Raspberry Pi 4.



Rysunek 13. ESP8266 pinout.

W przypadku ESP wykorzystano dwa piny do czujnika temperatury i światła. Wykorzystany został wbudowany moduł WiFi który jest wbudowany w układ SoC.

Dla Raspberry nie wykorzystano żadnych pinów. Został użyty moduł WiFi wbudowany w układ SoC znajdujący się na płytce.

## Software – Raspberry Pi

Aby móc wykorzystać Raspberry Pi 4 w projekcie należało zainstalować system operacyjny na karcie microSD. Wykorzystany system operacyjny to:

**Raspberry Pi OS Lite (32-bit) – port of Debian Bullseye with no desktop enviroment.**

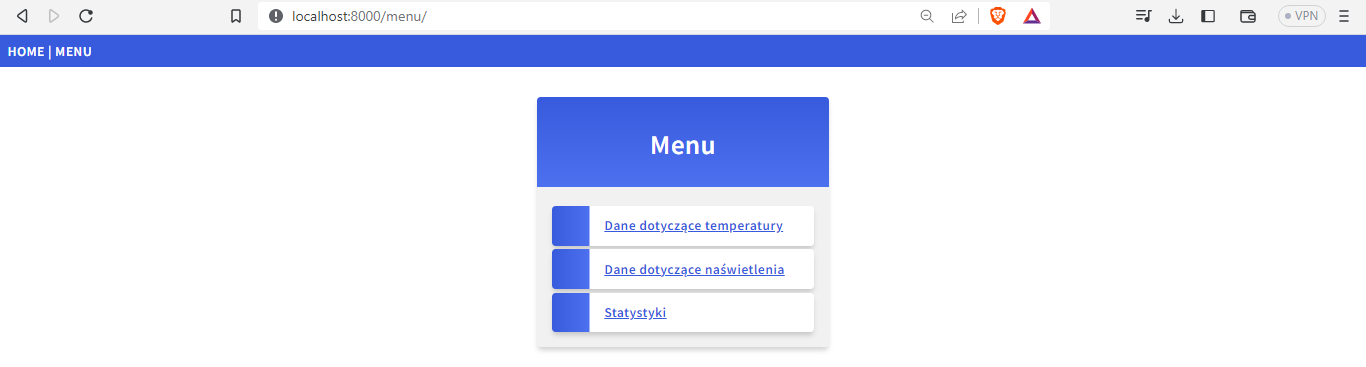
Kolejnym krokiem było ustawienie statycznego adresu IP urządzenia, tak aby można było połączyć się zdalnie przez protokół SSH. Kolejnych konfiguracji systemu dokonano z poziomu konsoli. Dostęp do konsoli był zdalny, wykorzystano program Putty. Do kopiowania plików między komputerem a Raspberry wykorzystano program WinSCP. Do pisania programów wykorzystano VisualStudio Code wraz z dodatkiem SSH-remote.

### Serwer HTTP – Django

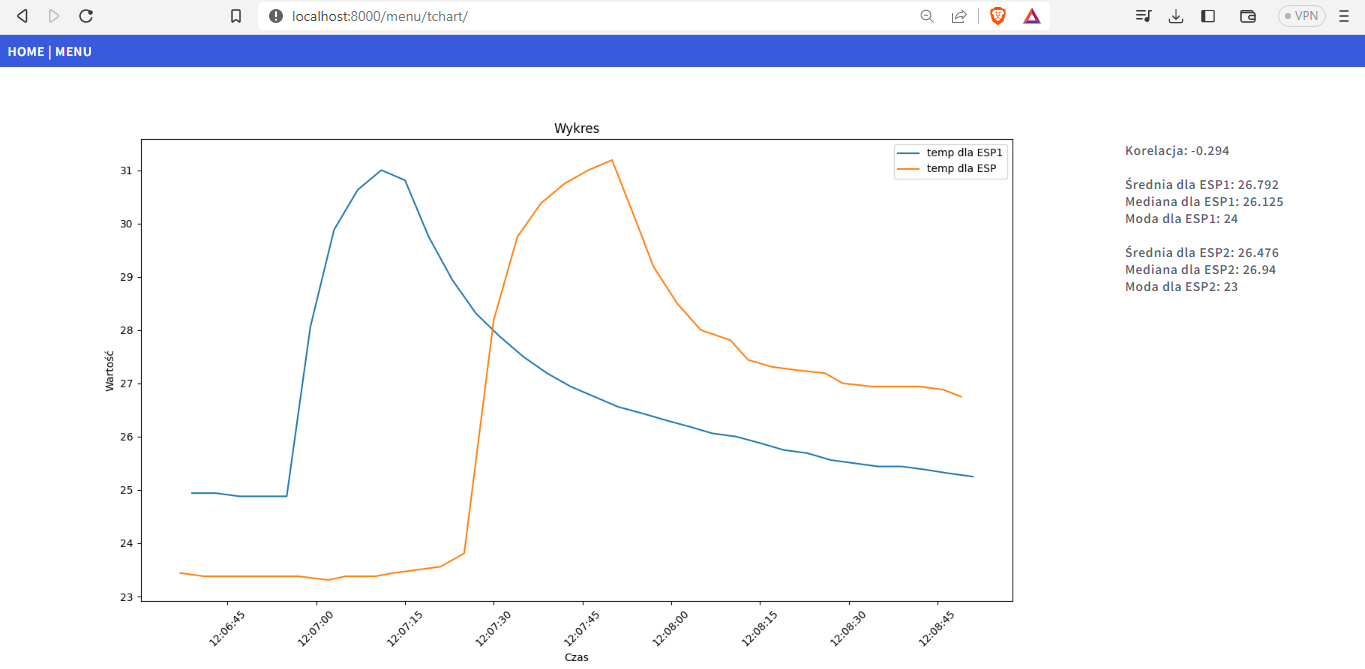
Dla stron internetowych skorzystano z serwera Django. Cała aplikacja została napisana w frameworku o takiej samej nazwie. Framework umożliwia pisanie aplikacji internetowych o architekturze plików MVC (Model-View-Controller). Utworzone strony internetowe:



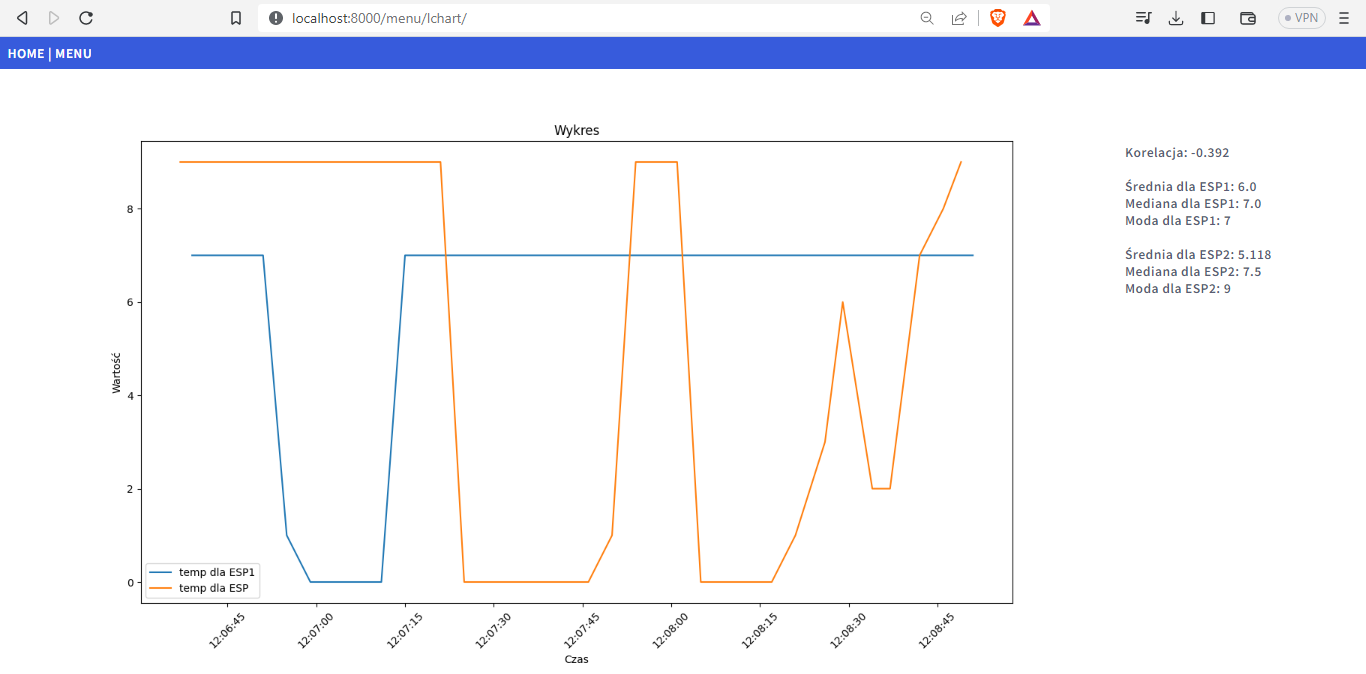
Rysunek 14. Strona startowa aplikacji.



Rysunek 15. Menu aplikacji.



Rysunek 16. Wykresy temperatury + statystyki.



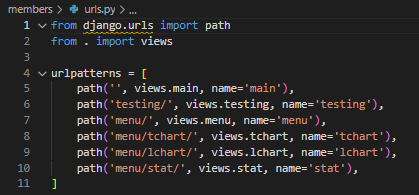
Rysunek 17. Wykresy naświetlenia + statystyki.

**Zasada działania programu**

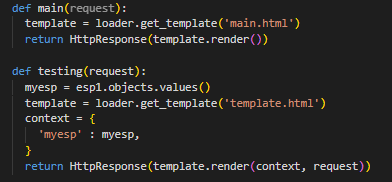
Dane są pobierane z bazy danych MySQL i wyświetlane na wykresach. Do wykresów i analityki danych wykorzystano matplotlib i pandas.

Z punktu widzenia struktury MVC, najważniejsze dla nas pliki projektu to:

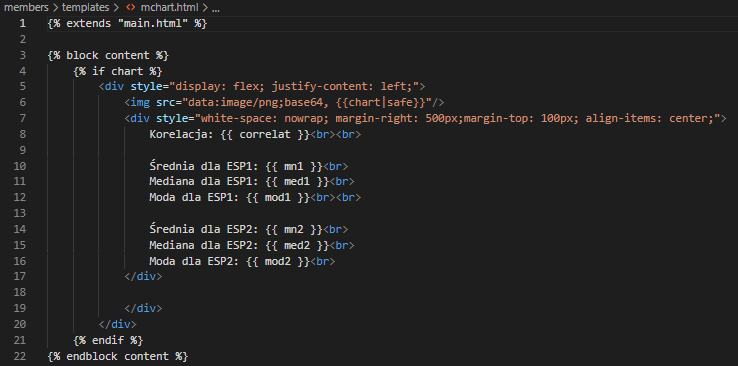
* urls.py - odpowiada za kontrolę adresów URL i przekierowania między widokami, których zadaniem jest generowanie odpowiedzi dla danego żądania HTTP.
* views.py - to plik, w którym znajdują się widoki aplikacji, czyli funkcje, które wykonują logikę biznesową aplikacji i generują odpowiedzi na żądania HTTP.
* Katalog templates - zawiera pliki HTML, które odpowiadają za prezentację danych generowanych przez widoki.



Rysunek 18. Kontroler aplikacji.



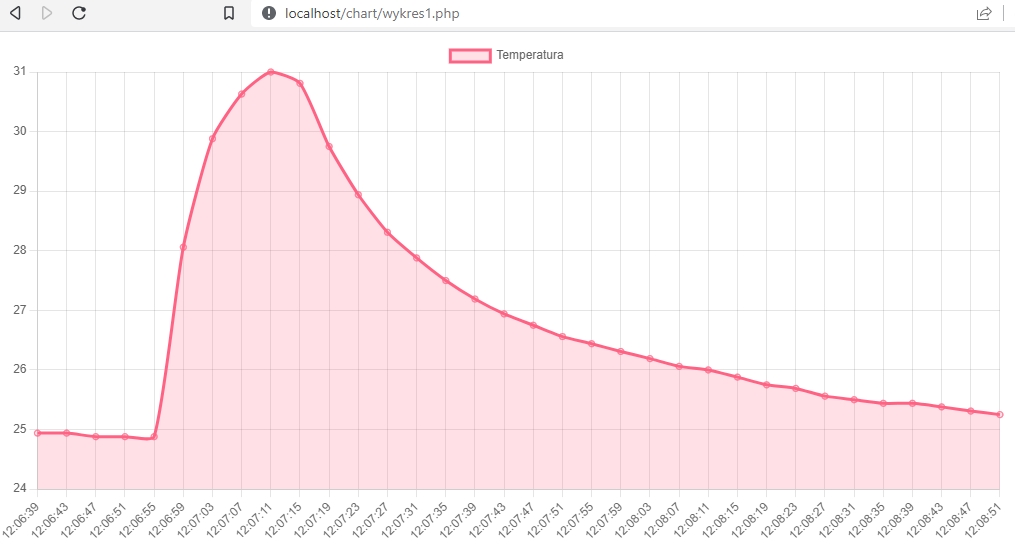
Rysunek 19. Fragment pliku odpowiedzialnego za widoki aplikacji.



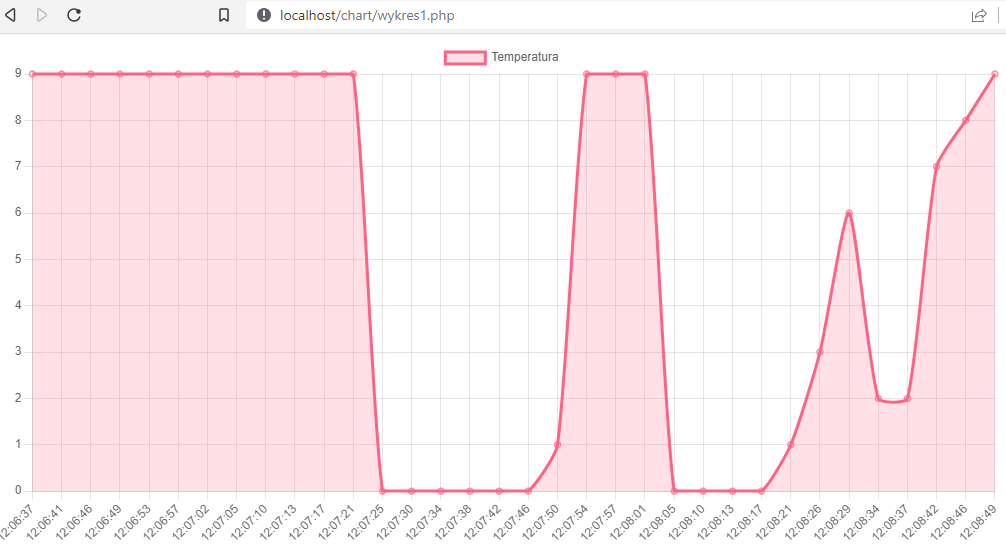
Rysunek 20. Model - przykładowa strona odpowiedzialna za wyświetlenie wykresu i statystyk.

W powyższym przypadku wykresy są generowane przez funkcje matplotlib i printowane na obraz png który jest umieszczany na stronie.

Poniżej przedstawiono koncepcje tworzenia interaktywnych wykresów w JavaScript. Ta część projektu jest jeszcze rozwijana.

**Utworzone interaktywne wykresy w JS…**

Rysunek 21. Wykres temperaturowy.

****

Rysunek 22. Wykres naświetlenia.

### Baza danych MySql – dostęp phpMyAdmin

Należało utworzyć serwer www na Raspberry Pi. Tworzenie i konfiguracja serwera sprowadzała się do podania następujących po sobie komend. Komendy były wprowadzane w powłoce Bash systemu Raspbian.

Instalacja LAMP Apache2:

**sudo apt install apache2 –y**

Instalacja interpretera PHP:

**sudo apt install php –y**

Instalacja bazy danych MySql:

**sudo apt install mysql-server php-mysql –y**

Instalacja phpMyAdmin:

**sudo apt install phpmyadmin**

Powyższe kroki powodują instalacje podstawowych elementów związanych z serwerem Apache2 i bazą danych MySQL.

Dostęp do bazy danych odbywał się z poziomu przeglądarki internetowej. Należało przejść do strony phpMyAdmin znajdującej się na serwerze localhost Apache2 na Raspberry Pi. Dokonano utworzenie nowej bazy danych oraz tabeli.

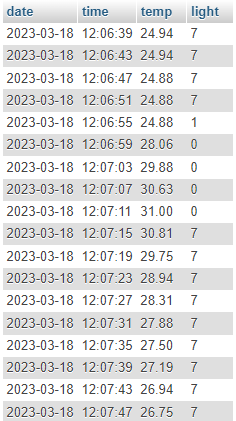


Rysunek 23. Utworzona baza danych oraz tabele.

W tych tabelach były gromadzone dane z czujników. W każdej tabeli zostały utworzone następujące kolumny:

* date – aktualna data dokonania pomiaru
* time, - aktualna godzina, minuta i sekunda dokonania pomiaru
* temp – odczytana wartość temperatury w st. C
* light – odczytana wartość naświetlenia, wartości 0-10 (gdzie: 0 – bardzo jasno, 10 – bardzo ciemno)

Dokonano odczytu ww wartości co 4 sekundy, wynik wypełnionej tabeli esp1:



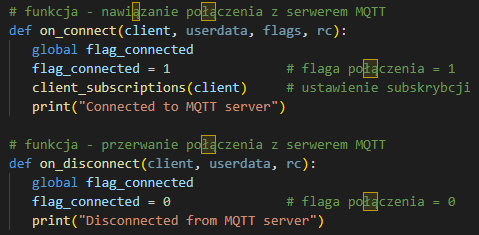
Rysunek 24. Wypełniona tabela esp1.

### Serwer MQTT – język Python

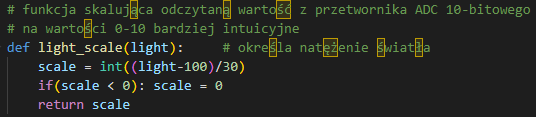
Poniżej zostaną omówione najważniejsze części programu wykonanego w języku Python. Wykonano dwa programy, odpowiednio dla subskrybcji i publikowania tematów. Poszczególne programy i ich funkcje:

* **rasp\_sub.py**
  + nawiązanie połączenia z serwerem MQTT
  + ustawienie subskrybcji tematów od ESP8266
  + wysyłanie odepranych danych do bazy danych
* **rasp\_pub.py**
  + nawiązanie połączenia z serwerem MQTT
  + ustawienie publikowanie tematów od Raspberry Pi
  + przykład użycia programu – wysyłanie inkrementowanej zmiennej

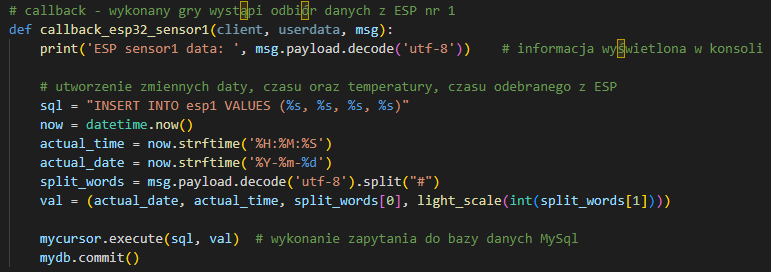
Najważniejsze fragmenty programu **rasp\_sub.py**:



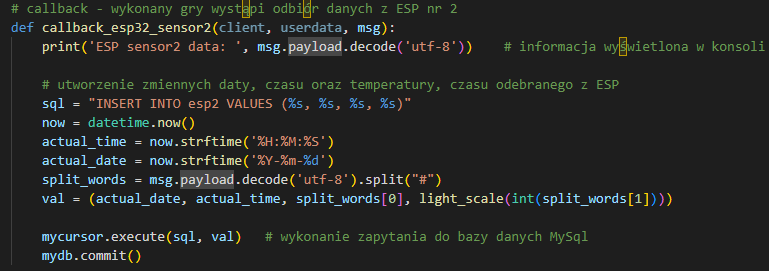
Fragment kodu 1. Funkcje wykonane przy nawiązaniu i zerwaniu połączenia z serwerem.



Fragment kodu 2. Funkcja skalująca wartość natężenia światła.



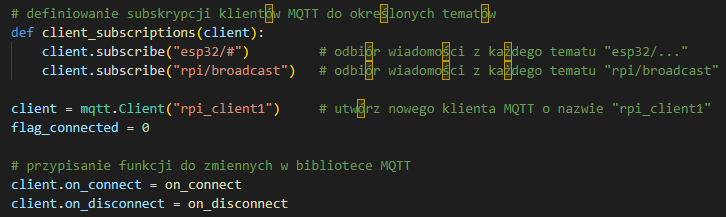
Fragment kodu 3. Callback ESP nr 1 - odbiór danych, wysłanie do bazy danych.



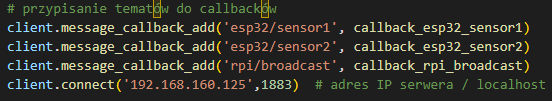
Fragment kodu 4. Callback ESP nr 2 - odbiór danych, wysłanie do bazy danych.



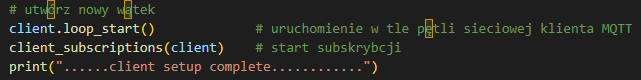
Fragment kodu 5. Callback Raspberry Pi - odbiór danych, wyświetlenie w konsoli



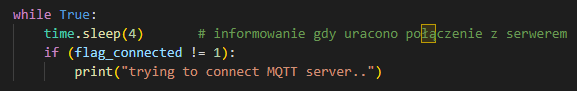
Fragment kodu 6. Zdefiniowanie subskrybcji tematów, tworzenie klienta sieci.



Fragment kodu 7. Przypisanie callbacków.

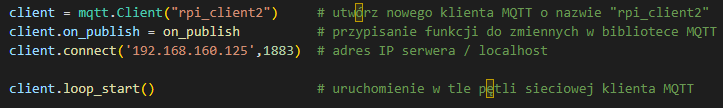


Fragment kodu 8. Subskrybcja odbywa się w nowym wątku.

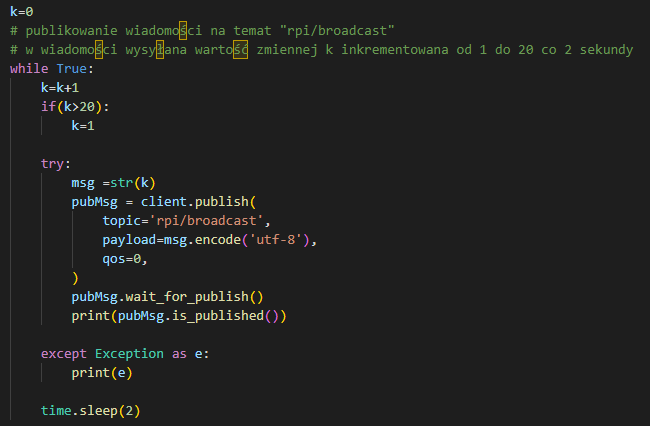


Fragment kodu 9. Pętla główna programu.

Najważniejsze fragmenty programu **rasp\_pub.py**:



Fragment kodu 10. Utworzenie nowego klienta sieci, nawiązanie połączenia z serwerem, przeniesienie procesów do nowego wątku.



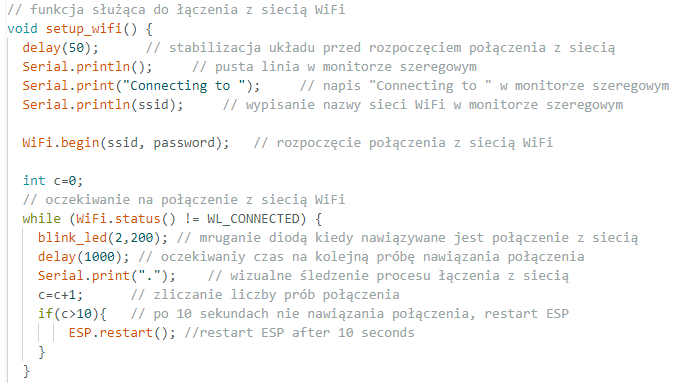
Fragment kodu 11. Pętla główna programu.

.

## Software ESP8266 – język Arduino

Poniżej zostaną omówione najważniejsze części programu wykonanego w języku Arduino. Wykonany program odpowiedzialny jest m.in:

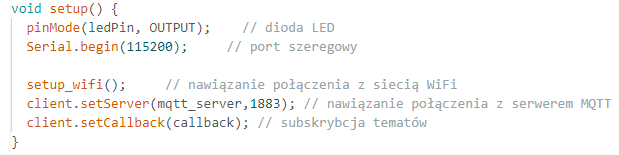
* Nawiązanie połączenie się z siecią WiFi,
* Nawiązanie połączenie z serwerem MQTT,
* odbiór danych z czujnika temperatury i światła,
* subskrybowanie i publikowanie tematów przez protokół MQTT.



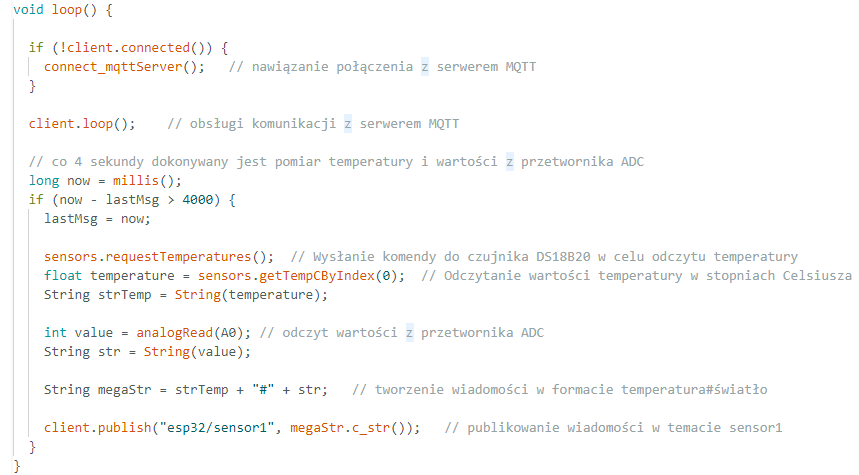
Fragment kodu 12. Nawiązanie połączenie się z siecią WiFi.



Fragment kodu 13. Nawiązanie połączenie z serwerem MQTT.



Fragment kodu 14. Konfiguracja portu szeregowego, łączenie z siecią WiFi - serwerem MQTT, subskrybcja tematu.



Fragment kodu 15. Odbiór danych z czujników, publikowanie tematu.

Program dla drugiego ESP3266 został napisany w analogiczny sposób.

# Przeprowadzone testy

# Podsumowanie

# Zakończenie

[1] - <https://mikrokontroler.pl/2021/06/07/inteligentne-systemy-parkingowe-strumien-przestrzenny-czujnikow-iot/>

[2] - <https://botland.com.pl/cyfrowe-czujniki-temperatury/165-czujnik-temperatury-ds18b20-cyfrowy-1-wire-tht-5904422366513.html>

[3] - <https://www.egsystem.pl/przykladowe-rozwiazanie-sygnalizacji-obecnosci-napiecia-w-rozdzielnicy-sredniego-napiecia>

[4] - <https://randomnerdtutorials.com/esp8266-pinout-reference-gpios/>

<https://mechatronicsblog.com/esp8266-nodemcu-pinout-for-arduino-ide/>

[5] - <https://roboticsbackend.com/raspberry-pi-3-pins/>

[6] - <https://forbot.pl/blog/kurs-raspberry-pi-projekty-prosty-serwer-www-id27882>

[7] - https://sieci.infopl.info/images/l\_rtopologiewlan.png

[8] - <https://strefainzyniera.pl/images/artykuly/6859b3907.jpg>

[9] - <https://hlassets.paessler.com/common/files/infographics/mqtt-architecture.png>

[10] - <https://netvel.sk/wp-content/uploads/2017/10/Install-Apache-Web-Server-on-Ubuntu.png>

[11] - https://pomoc.unicloud.pl/wp-content/uploads/2017/09/building-m2m-iot-projects.png