1. Cele i opis projektu

Celem projektu było zaprojektowanie automatycznego przełącznika światła, sterowanego poprzez kliencką aplikację webową, udostępnianą przez serwer. Przełącznik został dodatkowo wyposażony w 2 dodatkowe funkcjonalności, tj. manualny przełącznik światła oraz czujnik temperatury i wilgotności. Dane uzyskane na podstawie odczytów przekazywane są do serwera głównego, a następnie przekazywane do aplikacji klienckiej.

2. Budowa systemu

System składa się z serwera zarządzającego urządzeniami wejścia/wyjścia, serwera głównego, który komunikuje się bezprzewodowo z serwerem głównym, który jest „pomostem" pomiędzy serwerem, a aplikacją kliencką. Komunikacja odbywa się w pomiędzy nimi w obie strony. Lista wykorzystanych urządzeń:

- ESP8266 + NodeMCU v3
- Raspberry PI 3B+
- Moduł przekaźnika 3,3VDC
- Czujnik DHT11
- Przycisk
- Dioda jako źródło światła

Dodatkowo do wykonania projektu wymagane było posiadanie:

- Płytka prototypowa
- Przewody GOLDPIN męsko-męskie oraz męsko-żeńskie
- Przewód USB typ A – microUSB typ B
- Karta SD min 8gb, z zainstalowanym systemem operacyjnym obsługującym środowisko Node.js (np. Raspbian)

Moduł ESP8266 pełni rolę serwera oraz kontrolera urządzeń wejścia wyjścia. Komunikuje się z serwerem głównym poprzez sieć bezprzewodową (WiFi).

RaspberryPi 3B+ pełni funkcje serwera głównego, którego zadaniem jest utrzymywanie komunikacji z pozostałymi serwerami, przetworzenie otrzymanych danych oraz utrzymanie komunikacji z klientami poprzez zbudowaną na nim aplikację kliencką.

Do zaprogramowania modułu NodeMCU wykorzystano środowisko Arduino IDE, w którym zainstalowano paczkę z modułami ESP8266, korzystając z Menadżera Płytek. (*Więcej informacji na temat konfiguracji i obsługi środowiska Arduino IDE pod adresem: https://www.instructables.com/id/Quick-Start-to-Nodemcu-ESP8266-on-Arduino-IDE/).*

Skrypt Node.js można stworzyć w dowolnie wybranym edytorze tekstowym (np. Visual Studio Code lub Notepad++).

Poniżej przedstawiono schemat ideowy zbudowanego systemu:

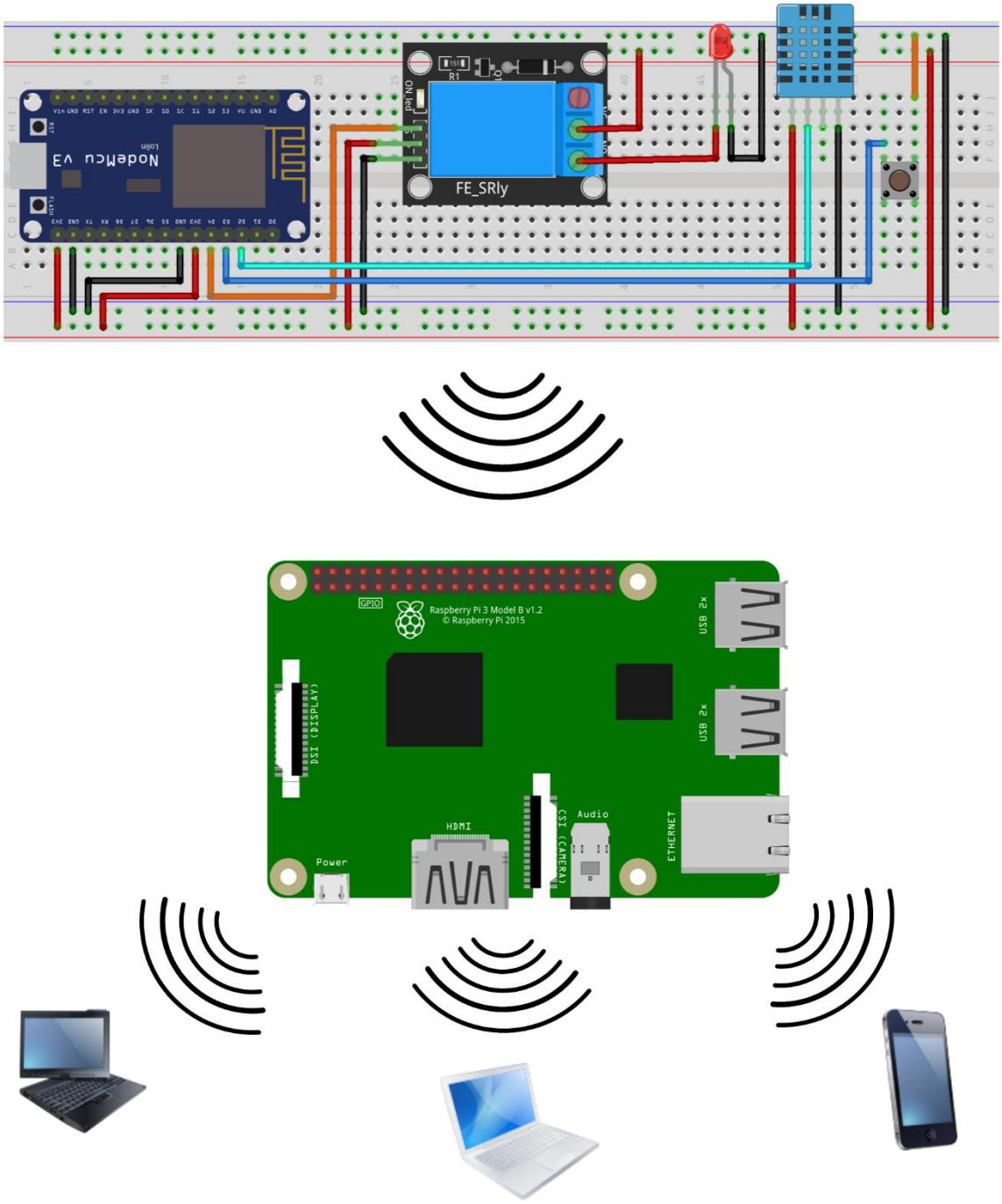


Tabela połączeń

NodeMCU	**Urządzenia I/O**
3,3V	Przekaźnik: VCC i VOUT, DHT11: VCC,
GND	Przekaźnik: GND, DHT11: GND, Dioda: GND, Przycisk: IN
D4	Przekaźnik: IN
D3	Przycisk: OUT
D2	DHT11: DOUT
	Przekaźnik: NO -> Dioda: VCC

3. Opis technologii

• ESP8266 - C/C++

- **WebServer** – biblioteka wykorzystana w celu stworzenia serwera uruchomionego na module ESP8266. W tym celu należy wykorzystać dołączoną do środowiska ArduinoIDE bibliotekę *ESP8266WebServer.h*, dodatkowo wspieraną przez bibliotekę *ESP8266WiFi.h*, która zapewnia obsługę komunikacji bezprzewodowej.
- **WebSocket** - jest technologią zapewniającą dwukierunkowy kanał komunikacji za pośrednictwem jednego gniazda TCP. Na urządzeniu ESP8266 znajduje się Serwer WebSocket, który wykorzystywany jest do dynamicznego przesyłania wartości odczytanych z czujnika DHT11, podczas gdy ulegną one zmianie. Do utworzenia serwera WebSocket, wymagane jest dodanie do projektu biblioteki *WebSocketsServer.h*.
- **REST/API** - jest usługą sieciową zaimplementowaną na bazie protokołu http. Wszystko jest zasobem (reprezentowanym przez jednoznaczny URL). Serwer nie tworzy sesji = łatwa skalowalność). Zasoby są abstrakcje, ale można się do nich dostać za pomocą zdefiniowanej reprezentacji (ten sam zasób może być udostępniany w wielu reprezentacjach, np. w formie XML lub JSON). Operacje ograniczają się do interfejsu http. W projekcie wykorzystywane są 3 z nich tj.: *GET* – pobiera reprezentację zasobu, jest bezpieczną operacją (nie zmienia żadnego obiektu) i może być cache'owane, POST – tworzy nowy zasób i w odpowiedzi w nagłówku Location zwraca URI nowego zasobu, *PUT* – aktualizuje zasób, operacja idempotenta.
- **JSON** - format wymiany danych komputerowych. Zapis i odczyt danych w tym formacie jest łatwy do opanowania przez ludzi. Jednocześnie, z łatwością odczytują go i generują komputery. Zbiór par nazwa/wartość. W różnych językach jest to implementowane jako obiekt, rekord, struktura, słownik, tabela hash, lista z kluczem, albo tabela asocjacyjna. Uporządkowana lista wartości. W większości języków implementuje się to za pomocą tabeli, wektora, listy, lub sekwencji. Do projektu dołączona została biblioteka *ArduinoJSON.h*, która pomaga kodować oraz dekodować obiekty w formacie JSON.
- Dodatkowe biblioteki: *Adafruit_Sensor.h* oraz *DHT.h* wymagane do obsługi czujnika DHT11.

Wszystkie wymienione biblioteki można znaleźć i zainstalować w Menedżerze Bibliotek środowiska Arduino IDE.

• RaspberryPI - Node.JS

Wieloplatformowe środowisko uruchomieniowe o otwartym kodzie do tworzenia aplikacji typu server-side napisanych w języku JavaScript. Dzięki Node.js możemy pisać aplikacje serwerowe, wykorzystując język JavaScript. Niebywałą zaletą platformy Node.js jest fakt, iż jej społeczność jest ogromna i cały czas się rozrasta. Wraz z rozwojem społeczności, pojawiają się nowe moduły, które wykorzystać możemy we własnych projektach dzięki npm.

- **NPM** - domyślny manager pakietów dla środowiska Node.js, może być także używany do zarządzania warstwą front-end aplikacji WWW. npm jest aplikacją wiersza poleceń, za pomocą której można instalować aplikacje dostępne w repozytorium npm. Strona domowa aplikacji zawiera wyszukiwarkę pakietów. Repozytorium jest publiczne i darmowe dla pakietów Open Source, ale istnieją także prywatne repozytoria dostępne za opłatą. npm jest standardowo dostępny, jeśli zainstalowane jest środowisko Node.js. Aby zainstalować pakiet wykonujemy poniższe polecenie:

  npm install moment

  Następnie, w aplikacji Node.js możemy użyć pakietu w następujący sposób

var moment = require('moment');

W ten sposób wykorzystano takie pakiety jak:

- **Request** – pakiet stworzony w celu ułatwienia wykorzystywania zapytań html (REST). Dokumentację modułu Request można znaleźć pod adresem: *https://www.npmjs.com/package/request*
- **Async** – pakiet stworzony w celu wykonywania i używania języka JavaScript w sposób asynchroniczny. W wypadku projektu moduł służy do wykonywania kilku zapytań Rest w jednej funkcji. Dokumentacja pakietu async: *https://www.npmjs.com/package/request*
- **WebSocket** – biblioteka pozwalająca na uruchomienie zarówno serwera jak i klienta websocket na serwerze stworzonym w Node.JS. Link do dokumentacji: *https://www.npmjs.com/package/websocket*

Spis wykorzystanych pakietów i dane na temat wymagań, które należy spełnić próbując uruchomić aplikację znajdują się w pliku package.json, który zainicjowano na początku procesu tworzenia wywołując w konsoli polecenie

npm init

Po przejściu przez wszystkie etapy kreatora, zostanie wygenerowany w/w plik. Więcej informacji na temat npm można znaleźć pod aresem: *https://www.sitepoint.com/beginners-guide-node-package-manager/*

- **REST/API**
- **JSON**

- **Aplikacja webowa**
  - **HTML**
  - **CSS**
  - **Java Script**
    - **WebSocket**
    - **REST/API**
    - **JSON**
    - **jQuery** - biblioteka programistyczna dla języka JavaScript, ułatwiająca korzystanie z JavaScriptu (w tym manipulację drzewem DOM). Kosztem niewielkiego spadku wydajności w stosunku do profesjonalnie napisanego kodu w niewspomaganym JavaScripcie pozwala osiągnąć interesujące efekty animacji, dodać dynamiczne zmiany strony, wykonać zapytania AJAX. Większość wtyczek i skryptów opartych na jQuery działa na stronach nie wymagając zmian w kodzie HTML. Aby móc korzystać z biblioteki jQuery, należy w kodzie aplikacji klienckiej dodać link do skryptu zawierającego bibliotekę:

      <script
      src='https://cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js'>
      </script>

4. Opis działania

Kody źródłowe wszystkich poniżej omawianych aplikacji znajdują się w archiwum Source dołączonego do dokumentacji.

Esp8266:

Opis najważniejszych funkcji aplikacji:

- InitWifi() – funkcja odpowiedzialna za utworzenie połączenia bezprzewodowego oraz jego konfiguracje. Określono tryb pracy sieci WiFi, ustawiono statyczne ip oraz połączono z siecią o SSID i haśle określonym w parametrach.
- GetIndex(), GetDTH(), …, GetRelay() – funkcje odpowiedzialne za przygotowanie danych w formacie JSON oraz odpowiedzi nimi na zapytanie HTTP_GET. Na początku każdej funkcji tworzony jest Bufor o objekt JSON, następnie każdy obiekt wypełniany jest danymi, w zależności od zapytania oraz przygotowywany do wysłania. W ostatnim kroku serwer wysyła odpowiedź.
- PostPutRealy() – funkcja odpowiedzialna za wykonywanie akcji na zapytania HTTP_POST oraz HTTP_PUT. Dane przesyłane do serwera są dekodowane i na ich podstawie odejmowane są akcje i wysyłane odpowiedzi.
- ConfigRouting() – funkcja, zawierająca tablicę routingu serwera. W zależności od otrzymanego zapytania API, oraz rodzaju zapytania, uruchamiana jest funkcja obsługująca zdarzenie.
- socketSendData() – funkcja kodująca dane w format JSON oraz przesyłająca je do wszystkich klientów WebSocket. Uruchamiana, gdy dane pochodzące z czujnika DHT11 ulegną zmianie.

RaspberryPi:

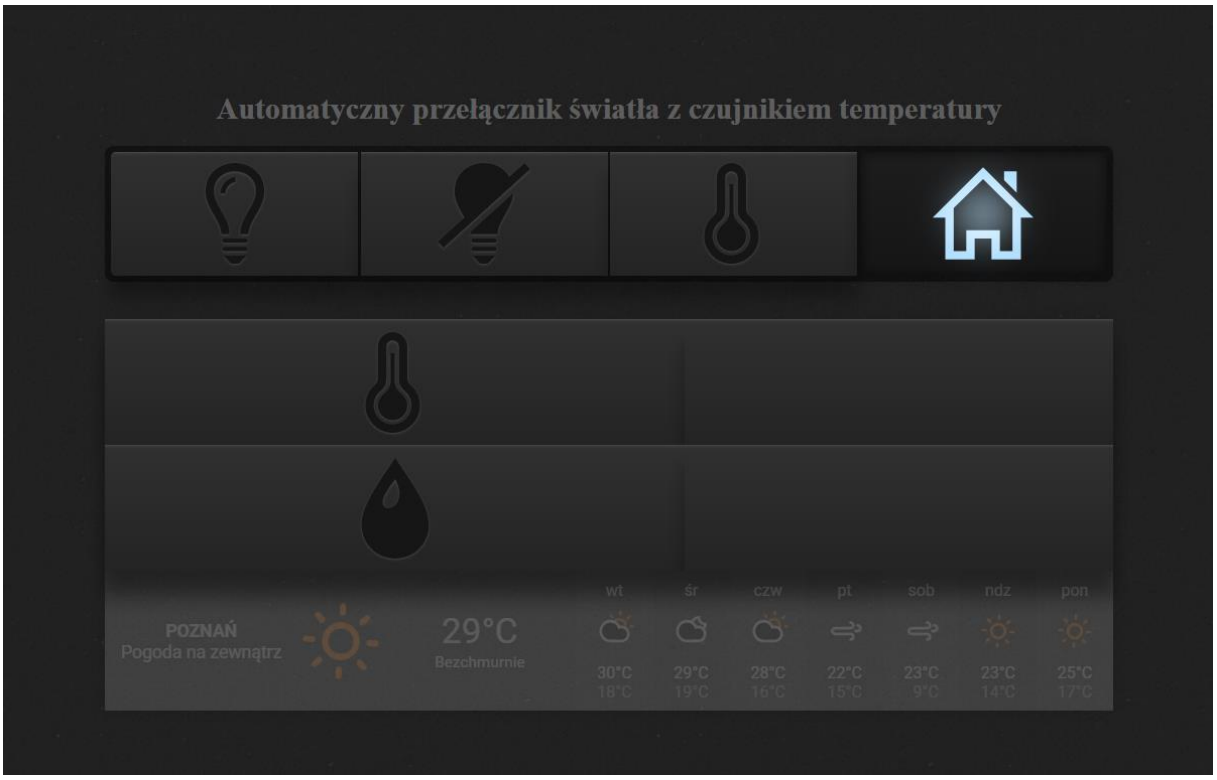
Opis najważniejszych funkcji:

- http.createServer(function (req, res) – funkcja będąca tablicą routingu serwera, wykonuje akcje w odpowiedzi na zapytania REST klienta.
- broadcast(DHT) – wysyła dane otrzymane z serwera pracującego na NodeMCU do każdego klienta za pomocą serwera WebSocket.
- wsServer.on(), wsClient.on() – wykonywanie akcji w odpowiedzi na zdarzenia pochodzące od klientów WebSocket, oraz serwera WebSocket na ESP8266.

Aplikacja klienckai:

- Socket.onopen, Socket.onmessage – podejmowanie akcji w odpowiedzi na zdarzenia pochodzące z serwera WebSocket.
- SendingButtonState() – wysyłanie statusu wciśnięcia przycisków za pomocą WebSocket, do innych klientów.

Gotowa aplikacja kliencka prezentuje się w następujący sposób:


Automatyczny przełącznik światła z czujnikiem temperatury
wt śr czw pt sob ndz pon
POZNAŃ
Pogoda na zewnątrz
29°C
Bezchmurnie
30°C 29°C 28°C 22°C 23°C 23°C 25°C
18°C 19°C 16°C 15°C 9°C 14°C 17°C


Automatyczny przełącznik światła z czujnikiem temperatury
28°C
37%
wt śr czw pt sob ndz pon
POZNAŃ
Pogoda na zewnątrz
29°C
Bezchmurnie
30°C 29°C 28°C 22°C 23°C 23°C 25°C
18°C 19°C 16°C 15°C 9°C 14°C 17°C