Tytuł pracy:
Stacja Pogodowa IoT

Streszczenie:
Praca obejmuje zbudowanie i zaprogramowanie urządzenia stacji pogodowej jako systemu wbudowanego w stylu „urządzenia internetu rzeczy”, wykorzystując nowoczesny mikrokontroler z modułem WiFi. Urządzenie wyświetla dane na ekranie LCD oraz udostępnia serwer HTTP z prostą stroną internetową do podglądu danych i stanu urządzenia. Serwer udostępnia też API typu REST do programistycznego uzyskiwania danych. Poza wykonaniem stacji pogodowej wykonano aplikację na komputery klasy PC do zmieniania ustawień urządzenia.

Słowa kluczowe:
stacja pogodowa, system wbudowany, internet rzeczy

Thesis title:
Weather Station IoT

Abstract:

The project involves building and programming a weather station in the style of an “internet of things” embedded system device, using a modern microcontroller with WiFi module. The device displays data on its LCD and shares a HTTP server with a simple website for displaying weather data and the device’s state. The server also presents a REST API for programmatic data acquisition. Besides building the weather station the project includes a PC application used for changing the device’s settings.

Keyword:
weather station, embedded system, internet of things

SPIS TREŚCI

1. Wstęp

Stacje pogodowe są bardzo przydatnymi urządzeniami używanymi w życiu codziennym przez wielu ludzi, o szerokim zakresie wykorzystywania. Z punktu widzenia konsumenta informacje dostarczane przez tego rodzaju urządzenia można wykorzystać aby na przykład: dobrać odpowiednio ubranie wierzchnie bez potrzeby wychodzenia najpierw na zewnątrz mieszkania. Z perspektywy automatyzacji zarządzania domami stacje pogodowe używane wewnątrz budynku mogą stanowić punkt odniesienia i stanowić o decyzji podniesienia czy obniżenia temperatury pieca ogrzewającego mieszkanie. Stacje pogodowe wykorzystywane są także przez stacje meteorologiczne do zbierania danych pogodowych, co pozwala na dokładniejsze prognozowanie pogody w dniach następujących.

Pomimo bardzo szerokiego zakresu wykorzystywania stacji pogodowych, ich konkretne implementacje są bardzo skupione na wykonywaniu jednego zadania. Produkty konsumenckie zazwyczaj pokazują tylko podstawowe informacje na ekranie urządzenia bez możliwości zapamiętywania odczytanych danych, czujniki temperatur stosowane do komunikacji z piecami udostępniają swoje dane tylko kompatybilnym piecom a rozwiązania kierowane do użycia w przemyśle są bardzo drogimi i skomplikowanymi produktami.

1.1. Cel projektu

Ze względu na wymienione informację tę pracę poświęcono na zbudowaniu prototypu stacji pogodowej, która będzie prostym i niedrogim urządzeniem typu „urządzenia Internetu Rzeczy”, możliwym do wykorzystania w domach rodzinnych do podglądania warunków pogodowych na zewnątrz budynku, do automatyzacji wszelakich systemów używając zapytań HTTP jako metody interakcji z urządzeniem czy do zbierania danych historycznych warunków pogodowych do celów bardziej profesjonalnych.

1.2. Motywacja

Motywacją do wykonania tego rodzaju urządzenia jest chęć wykorzystania i pogłębienia wiedzy nabytej w trakcie studiów nie tylko z przedmiotów związanych bezpośrednio z informatyką które dostarczyły umiejętności implementacji oprogramowania urządzenia, ale także z przedmiotów związanych z elektroniką, dzięki którym zdobyto umiejętności zaprojektowania systemu wbudowanego, który umożliwia współpracę różnych urządzeń i sensorów w świecie rzeczywistym.

1.3. Porównanie z produktami rynkowymi

Porównanie projektowanego urządzenia z produktami rynkowymi zostało wykonane tylko z urządzeniami kierowanymi do użytkowników domowych, na podstawie najpopularniejszego urządzenia [1], reklamowanego jako stacja pogodowa, oferowanego przez sklep „RTV Euro AGD" o nazwie: „*Reinston ESPOG02*". Popularność produktu wyznaczono na podstawie największej ilości opinii na stronie internetowej ww. sklepu. Projektowanego urządzenia nie porównywano do rozwiązań kierowanych do środowisk profesjonalnych gdyż różnice w funkcjonalności, specjalizacji urządzeń, jakości i klasie cenowej były zbyt duże aby dokonać jakiejkolwiek miarodajnej analizy porównawczej.

Tabela 1 - Porównanie projektowanego urządzenia z Reinston ESPOG02

Cecha	Projektowane urządzenie	Reinston ESPOG02
Ilość modułów urządzenia	• Jeden	• Dwa (pomiar docelowo wewnątrz i na zewnątrz)
Rodzaj czujników	• Czujnik temperatury • Czujnik wilgotności • Czujnik ciśnienia	• Czujnik temperatury • Czujnik wilgoci • Czujnik ciśnienia

Funkcja prognozy pogody	• Nie	• Tak
Zegar	• Tak (synchronizowany protokołem SNTP)	• Tak (synchronizowany sygnałem DCF)
Wymagane połączenie z internetem	• Tak (możliwa jest praca urządzenia bez internetu, ze zmniejszoną funkcjonalnością)	• Nie
Dostęp do danych	• Wyświetlacz LCD (część danych) • Strona internetowa • Karta SD (część danych)	• Wyświetlacz LCD
Interakcja z urządzeniem	• Za pomocą strony internetowej • Za pomocą karty SD	• Za pomocą przycisków na urządzeniu
Funkcje dodatkowe	• Zbieranie danych historycznych • Możliwość interakcji z urządzeniem za pomocą REST API	• Alarm mrozu • Budzik • Wyświetlanie faz księżyca

Porównując urządzenia na podstawie danych z tabeli 1 można zauważyć ich różnice wynikające z idei ich zaprojektowania. Urządzenie Reinston ESPOG02 jest typowym urządzeniem stacji pogodowej, gdzie głównym i zarazem jedynym sposobem interakcji z urządzeniem ma być moduł urządzenia z przyciskami na obudowie oraz wyświetlanie danych zarejestrowanych (i innych funkcjonalności) przez urządzenie odbywa się na wyświetlaczu LCD. W przeciwieństwie: projektowane urządzenie udostępnia tylko podstawowe, aktualne, dane na wyświetlaczu LCD urządzenia, a głównym sposobem interakcji z urządzeniem ma być interfejs strony internetowej oferowanej przez serwer HTTP urządzenia. Można więc porównać urządzenie Reinston ESPOG02 do klasycznej stacji pogodowej, a projektowane

urządzenie do klasy nowoczesnych urządzeń korzystających z technologii internetowych i komputerowych.

Dwiema głównymi różnicami tych urządzeń jako stacji pogodowych są:

1. Projektowane urządzenie wymaga połączenia internetowego aby użytkownik mógł uzyskać 100% funkcjonalności urządzenia, gdzie urządzenie Reinston ESPOG02 nie ma żadnych wymagań do poprawnej pracy,
2. Urządzenie Reinston ESPOG02 oferuje funkcjonalność prognozy pogody.

Pozostałe różnice także wynikają z filozofii przyjętych przez projektantów obu urządzeń. Produkt Reinston ESPOG02 oferuje dodatkowe funkcje uzupełniające pracę urządzenia jako stacji pogodowej (alarm mrozu, budzik, wyświetlanie faz księżyca), gdzie z kolei projektowane urządzenie skupia się na zbieraniu i udostępnianiu danych historycznych zarejestrowanych przez urządzenie, oraz dostęp do urządzenia nie tylko z poziomu graficznego (strony internetowej) ale także z poziomu programistycznego (interfejs REST API).

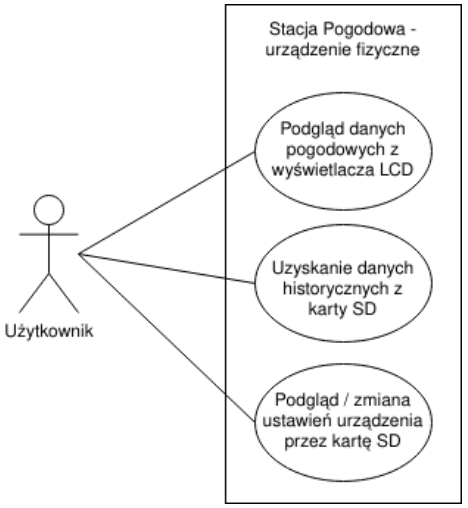
2. Założenia projektowe

Poniższy rozdział zaprezentuje założenia projektowe i wymagania stawiane projektowanemu urządzeniu, a następnie przedstawi listę urządzeń i technologii użytych do realizacji całego projektu.

Zakłada się zaprojektowanie i zbudowanie urządzenia o funkcjonalności stacji pogodowej. Z perspektywy urządzeń fizycznych zakłada się że główną jednostką logiczną i zarządzającą będzie mikrokontroler ESP32 firmy Espressif, który będzie zarządzał i korzystał z innych urządzeń cyfrowych wybranych na podstawie przedstawionych wymagań funkcjonalnych (rozdział 2.1). Urządzenie poza spełnianiem podstawowych funkcjonalności stacji pogodowej powinno spełniać też funkcjonalności będące charakterystycznymi dla urządzeń klasy „urządzeń Internetu Rzeczy", głównie: interfejs do interakcji z urządzeniem z poziomu komputera lub przeglądarki oraz możliwość interakcji z urządzeniem w sposób automatyczny – interakcja za pomocą jakiegoś interfejsu lub protokołu wymiany danych.

Dodatkowo zakłada się stworzenie aplikacji konfiguracyjnej na komputery klasy PC do konfiguracji podstawowych ustawień urządzenia stacji pogodowej, potrzebnych do jej prawidłowego funkcjonowania.

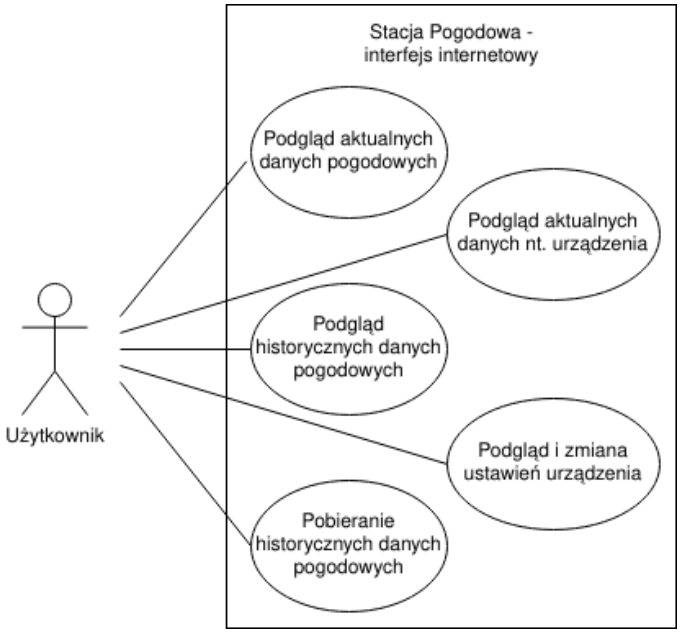
Korzystając z wyżej wymienionych założeń stworzono diagramy przypadków użycia stacji pogodowej i aplikacji konfiguracyjnej, przedstawione na rysunkach poniżej.



Rys. 1 - Diagram przypadków użycia - stacja pogodowa - fizycznie

Rysunek 1 przedstawia przypadki użycia stacji pogodowej jako urządzenia fizycznego. Zakłada się że użytkownik będzie mógł fizycznie spojrzeć na ekran LCD urządzenia i zobaczyć aktualne dane pogodowe, zarejestrowane przez urządzenie.

Dodatkowo poprzez fizyczną interakcję z kartą SD urządzenia (odczytywanie lub zapisywanie zawartości karty) użytkownik będzie w stanie uzyskać historyczne dane pogodowe zaobserwowane przez urządzenie i podejrzeć lub zmienić ustawienia urządzenia.

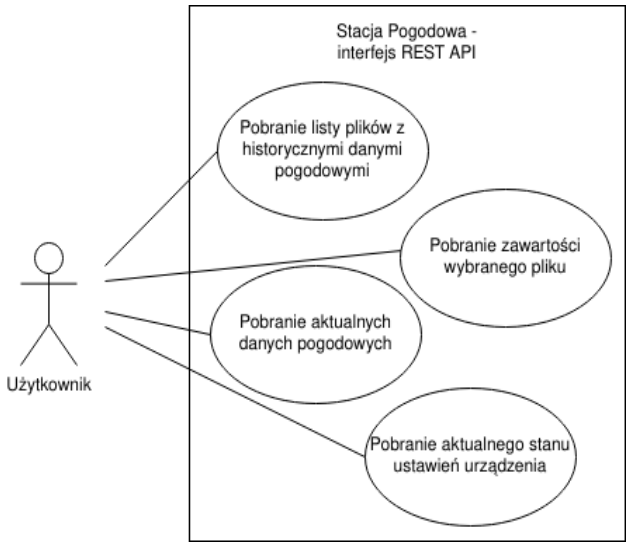


Rys. 2 - Diagram przypadków użycia - stacja pogodowa - interfejs internetowy

Diagram na rysunku 2 przedstawia przypadki użycia interfejsu internetowego urządzenia stacji pogodowej.

Podgląd aktualnych i historycznych danych pogodowych odbywać się będzie na tej samej stronie, gdzie aktualne lub wybrane historyczne dane pogodowe wyświetlane będą w postaci wykresów temperatury, wilgoci i ciśnienia. Dodatkowo wyświetlana będzie lista dostępnych do pobrania plików z historycznymi danymi pogodowymi.

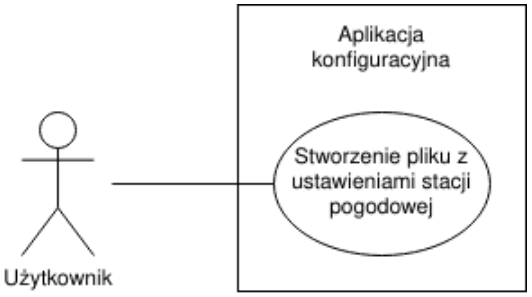
Podgląd aktualnych danych nt. urządzenia oraz podgląd i możliwość zmiany ustawień urządzenia odbywać się będzie na osobnych, dedykowanych do tego stronach.



Rys. 3 - Diagram przypadków użycia - stacja pogodowa - interfejs REST API

Rysunek 3 przedstawia przypadki użycia serwera HTTP urządzenia poprzez interfejs typu REST API do:

a) pobrania aktualnych danych nt. urządzenia, takich jak: aktualne dane pogodowe czy aktualne ustawienia urządzenia,
b) Pobrania listy plików z historycznymi danymi pogodowymi lub pobrania zawartości wybranego pliku.



Rys. 4 - Diagram przypadków użycia - aplikacja konfiguracyjna

Diagram przypadków użycia aplikacji konfiguracyjnej jest przedstawiony na rysunku 4. Aplikacja konfiguracyjna z założenia ma spełniać tylko jedną funkcjonalność, którą jest tworzenie pliku z ustawieniami, który docelowo ma zostać wgrywany na kartę SD stacji pogodowej.

2.1. Wymagania funkcjonalne i niefunkcjonalne

W tym rozdziale określone zostają wymagania funkcjonalne i niefunkcjonalne projektowanemu urządzeniu stacji pogodowej i aplikacji konfiguracyjnej, jako dopełnienie założeń projektowych i diagramów przypadków użycia z rozdziału 2, w celu jednoznacznego określenia wymagań. W celu zwięzłości i nie powtarzania informacji w tym rozdziale nie zostały podane wymagania które wynikają z diagramów przypadków użycia, np.: urządzenie będzie posiadać interfejs strony internetowej.

Tabela 2 - Wymagania funkcjonalne i niefunkcjonalne stacji pogodowej

Wymagania funkcjonalne	
Lp.	**Wymaganie**
1	Urządzenie powinno wyświetlać błędy w pracy na ekranie LCD oraz poprzez połączenie szeregowe mikrokontrolera z komputerem
2	Plik z ustawieniami urządzenia powinien być wykonany w znanym i popularnym formacie

3	Pliki z zapisanymi historycznymi danymi pogodowymi powinny być w formacie CSV i powinien przypadać jeden plik na jeden dzień
4	Urządzenie powinno dalej pracować po wyciągnięciu karty SD – bez funkcjonalności bezpośrednio związanej z kartą SD
5	Urządzenie powinno stosować protokół mDNS, w ramach dostępu do urządzenia na podstawie jego nazwy a nie adresu IP
6	Interfejs internetowy powinien udostępniać listę plików z historycznymi danymi pogodowymi z karty SD i pozwalać na wybór dowolnego pliku jako zawartości pokazywanej na wykresach danych
7	Interfejs internetowy nie powinien pozwalać na otwarcie pliku z ustawieniami jako pliku z historycznymi danymi pogodowymi do wyświetlenia na wykresach
8	Zmiana ustawień urządzenia poprzez interfejs internetowy powinna zapisywać ustawienia w pliku z ustawieniami na karcie SD
9	Lista plików z danymi udostępniana przez interfejs internetowy i REST API nie powinna zawierać plików lub katalogów niezwiązanych z pracą urządzenia.
10	Urządzenie powinno rejestrować: temperaturę, wilgoć otoczenia i ciśnienie

Wymagania niefunkcjonalne

Lp.	**Wymaganie**
1	Prosty i intuicyjny interfejs
2	Konsekwentny i spójny wygląd interfejsu
3	Jasny podział funkcjonalności na różne strony interfejsu internetowego

Tabela 3 - Wymagania funkcjonalne i niefunkcjonalne aplikacji konfiguracyjnej

Wymagania funkcjonalne	
Lp.	**Wymaganie**
1	Aplikacja ma tworzyć plik z ustawieniami urządzenia stacji pogodowej na podstawie wprowadzonych do aplikacji danych
2	Aplikacja ma być dostępna dla systemów operacyjnych: MacOS, Linux i Windows
Wymagania niefunkcjonalne	
Lp.	**Wymaganie**
1	Prosty i intuicyjny interfejs

2.2. Wykorzystane urządzenia

Projektowane urządzenie stacji pogodowej zbudowano przy pomocy komercyjnie dostępnych urządzeń. Poniżej w tabeli 4 przedstawiono listę użytych urządzeń i krótki ich komentarz.

Tabela 4 - Lista urządzeń i układów scalonych wykorzystanych w projektowanym urządzeniu

Nazwa	Opis
ESP32	● Mikrokontroler firmy Espressif ● Wersja ESP32-S2-DevKitC ● Służy jako główna jednostka logiczna urządzenia, i jako jednostka zarządzająca innymi urządzeniami
AM2320	● Producent: AOSONG ● Czujnik temperatury i wilgoci ● Komunikujący się za pomocą protokołu I2C
BMP280	● Producent: BOSCH ● Czujnik ciśnienia i temperatury ● Komunikujący się za pomocą protokołu I2C
Wyświetlacz LCD	● Wyświetlacz LCD o rozdzielczości 4x20 znaków ● Zbudowany na podstawie czipu HD44780 ● Używa czipu PCF8574 jako adaptera do protokołu I2C
PCF8574	● Ekspander wyprowadzeń o 8 wyjściach cyfrowych ● Kontrolowany za pomocą protokołu I2C ● Używany jako adapter wyświetlacza LCD do pracy z protokołem I2C
Czytnik kart SD	● Obwód scalony pracujący jako moduł czytnika kart SD ● Komunikujący się za pomocą interfejsu SPI

D24V5F3	● Producent: Pololu ● Przetwornica Step-Down ● Zamienia napięcie wejściowe 3.4 V - 36 V DC na napięcie wyjściowe 3.3 V DC ● Zastosowana jako wewnętrzne źródło napięcia 3.3 V w urządzeniu
Zasilacz sieciowy	● Zasilacz sieciowy impulsowy 5 V DC używany jako główne źródło zasilania urządzenia

W wyżej wymienionej tabeli 4:

- nie zaprezentowano producentów niektórych urządzeń gdyż sklep z którego zostały one zakupione nie oferował takich informacji,
- nie przedstawiono elementów montażowych, takich jak przewody, płytka do lutowania, itd.

Informacje co do implementacji, tj.: połączenia urządzeń z tabeli 4 w jeden spójny system, przedstawiono w rozdziale 3.

2.3. Wykorzystane technologie

Poniżej przedstawiono technologie, narzędzia i biblioteki wykorzystane w projekcie do budowy oprogramowania. Wszystkie poniżej wymienione pozycje są materiałami dostępnymi na podstawie otwartych licencji.

Do produkcji oprogramowania urządzenia stacji pogodowej wykorzystano język programowania C, oraz technologie HTML, CSS i JavaScript, jako technologie przy pomocy których zbudowano interfejs strony internetowej udostępnianej przez urządzenie. Narzędzia i biblioteki używane z ww. technologiami wykorzystane do implementacji oprogramowania mikrokontrolera ESP przedstawiono w tabeli 5.

Tabela 5 - Lista narzędzi i bibliotek użytych do implementacji oprogramowania urządzenia

Nazwa	Opis
Framework **ESP-IDF**	● Licencja Apache 2.0 ● Oficjalny framework przeznaczony do produkcji oprogramowania na urządzenia firmy Espressif ● Udostępnia oprogramowanie FreeRTOS ● Adres repozytorium: ○ https://github.com/espressif/esp-idf
Moduł **mDNS**	● Licencja Apache 2.0 ● Moduł frameworku ESP-IDF dostępny w suplementarnym repozytorium ● Moduł użyty aby uzyskać funkcjonalność protokołu mDNS ● Adres repozytorium: ○ https://github.com/espressif/esp-protocols/tree/master/components/mdns
Moduł **am2320**	● Licencja BSD-3-Clause ● Moduł jest częścią większej biblioteki nazwanej esp-idf-lib, która **nie** jest oficjalną częścią biblioteki firmy Espressif ● Moduł użyty jako sterownik do czujnika AM2320 ● Adres repozytorium: ○ https://github.com/UncleRus/esp-idf-lib/tree/master/components/am2320
Moduł **bmp280**	● Licencja MIT ● Moduł jest częścią większej biblioteki nazwanej esp-idf-lib, która **nie** jest oficjalną częścią biblioteki firmy Espressif ● Moduł użyty jako sterownik do czujnika BMP280 ● Adres repozytorium: ○ https://github.com/UncleRus/esp-idf-lib/tree/master/components/bmp280

Moduł **esp_idf_lib_helpers**	● Licencja ISC ● Moduł jest częścią większej biblioteki nazwanej esp-idf-lib, która **nie** jest oficjalną częścią biblioteki firmy Espressif ● Moduł użyty jako biblioteka pomocnicza, wymagana przez inne moduły biblioteki esp-idf-lib ● Adres repozytorium: ○ https://github.com/UncleRus/esp-idf-lib/tree/master/components/esp_idf_lib_helpers
Moduł **hdd44780**	● Licencja BSD-3-Clause ● Moduł jest częścią większej biblioteki nazwanej esp-idf-lib, która **nie** jest oficjalną częścią biblioteki firmy Espressif ● Moduł użyty jako sterownik do wyświetlacza LCD zbudowanego na podstawie czipu HD44780 ● Adres repozytorium: ○ https://github.com/UncleRus/esp-idf-lib/tree/master/components/hd44780
Moduł **i2cdev**	● Licencja MIT ● Moduł jest częścią większej biblioteki nazwanej esp-idf-lib, która **nie** jest oficjalną częścią biblioteki firmy Espressif ● Moduł użyty jako sterownik do pracy z protokołem I2C. Moduł wymagany przez inne używane moduły biblioteki esp-idf-lib ● Adres repozytorium: ○ https://github.com/UncleRus/esp-idf-lib/tree/master/components/i2cdev

Moduł **pcf8574**	● Licencja MIT ● Moduł jest częścią większej biblioteki nazwanej esp-idf-lib, która **nie** jest oficjalną częścią biblioteki firmy Espressif ● Moduł użyty jako sterownik do urządzenia PCF8574, będącego adapterem I2C dla wyświetlacza LCD ● Adres repozytorium: ○ https://github.com/UncleRus/esp-idf-lib/tree/master/components/pcf8574
Biblioteka **cJSON**	● Licencja MIT ● Biblioteka użyta do pracy z formatem tekstu JSON ● Adres repozytorium: ○ https://github.com/DaveGamble/cJSON

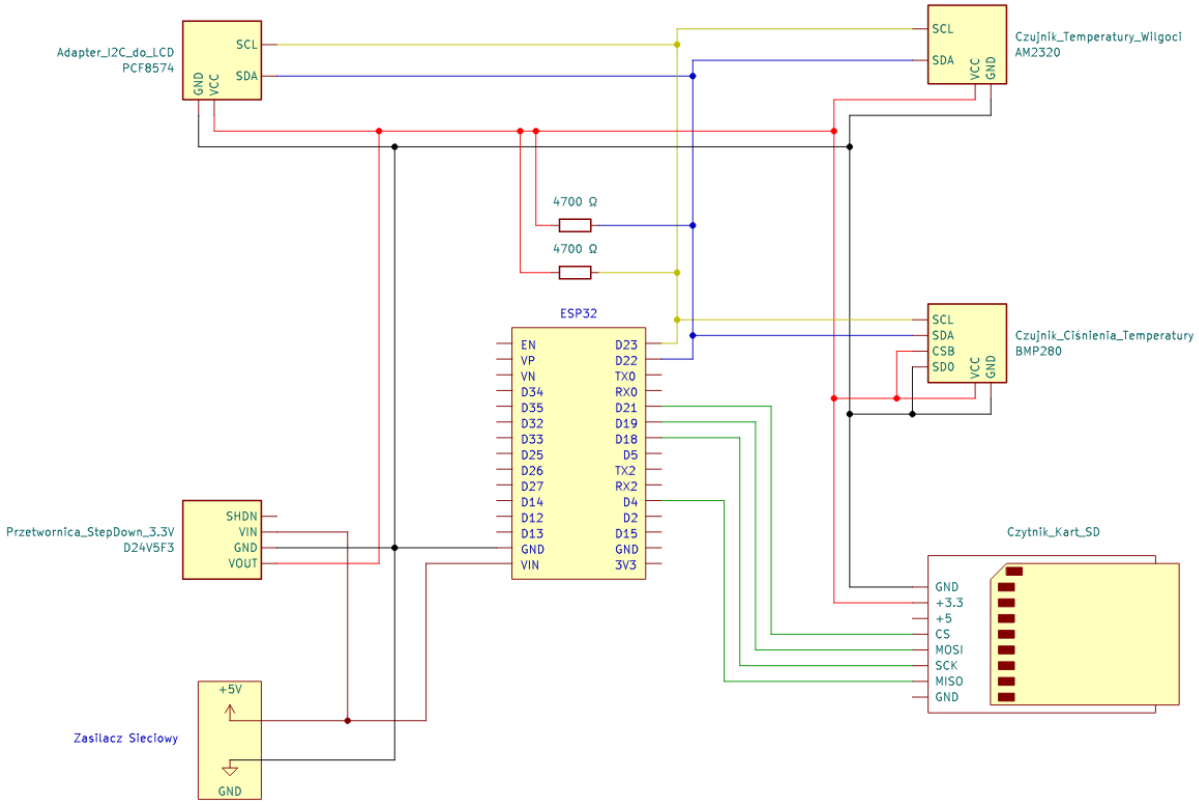
Do tworzenia oprogramowania aplikacji konfiguracyjnej na komputery klasy PC użyto głównie technologii HTML, CSS oraz JavaScript, jako że tą aplikację napisano przy użyciu frameworku ***Electron***. Pełną listę użytych narzędzi i bibliotek do tworzenia ww. oprogramowania przedstawiono w tabeli 6.

Tabela 6 - Lista narzędzi użytych do implementacji aplikacji konfiguracyjnej

Nazwa	Opis
Electron	● Licencja MIT ● Służy do budowania aplikacji na komputery przy użyciu technologii internetowych (jak HTML czy JavaScript) ● Adres repozytorium: ○ https://github.com/electron/electron
Biblioteka **fs-jetpack**	● Licencja MIT ● Moduł używany do pracy z plikami przez środowiska Node.js ● Adres repozytorium: ○ https://github.com/szwacz/fs-jetpack

3. Implementacja sprzętowa

W ramach pracy zaprojektowano i wykonano fizycznie urządzenie stacji pogodowej. Ten rozdział przedstawia (na rysunku 5, poniżej) i omawia schemat elektryczny będący podstawą implementacji sprzętowej projektowanego urządzenia. Poza omówieniem schematu ten rozdział przedstawia też zdjęcia prezentujące (podrozdział 3.1) wykonany prototyp urządzenia.



Rys. 5 - Schemat elektryczny urządzenia stacji pogodowej

Korzystając z listy urządzeń (przedstawionej w rozdziale 2.2) wykonano projekt połączeń elektrycznych urządzenia stacji pogodowej, przedstawiony na schemacie widocznym powyżej.

Urządzenie potrzebowało dwóch różnych poziomów napięcia zasilania:

- 5 V - dla zasilania mikrokontrolera ESP32,
- 3.3 V - dla zasilania wszystkich innych użytych urządzeń,

i jako że uznano zasilanie innych urządzeń poprzez wyjście cyfrowe mikrokontrolera za zły pomysł, należało zapewnić w urządzeniu dwa różne źródła napięcia. Zasilanie napięciem 5 V uzyskane jest przez bezpośrednie podłączenie przewodów do pozytywnego terminalu gniazda DC φ5.5 x 2.1 mm. Z kolei zasilanie napięciem 3.3 V uzyskano poprzez wykorzystanie urządzenia D24V5F3 (przetwornicy step-down) przetwarzającej napięcie zasilacza (5 V) na napięcie 3.3 V.

Aby urządzenie posiadało możliwość pracy z kartami SD użyto modułu czytnika kart SD w urządzeniu (Czytnik_Kart_SD na schemacie). Czytnik kart SD mógł być zasilany napięciem 3.3 V lub napięciem 5 V, ale jako że karty SD standardowo pracują z napięciem 3.3 V [2] zdecydowano się na użycie właśnie tej opcji poziomu napięcia zasilania urządzenia. Czytnik kart SD używał interfejsu SPI do komunikacji z urządzeniami nadrzędnymi, a więc połączono wyjścia interfejsu SPI czytnika do wybranych wyjść cyfrowych mikrokontrolera.

Wszystkie pozostałe urządzenia, tj.:

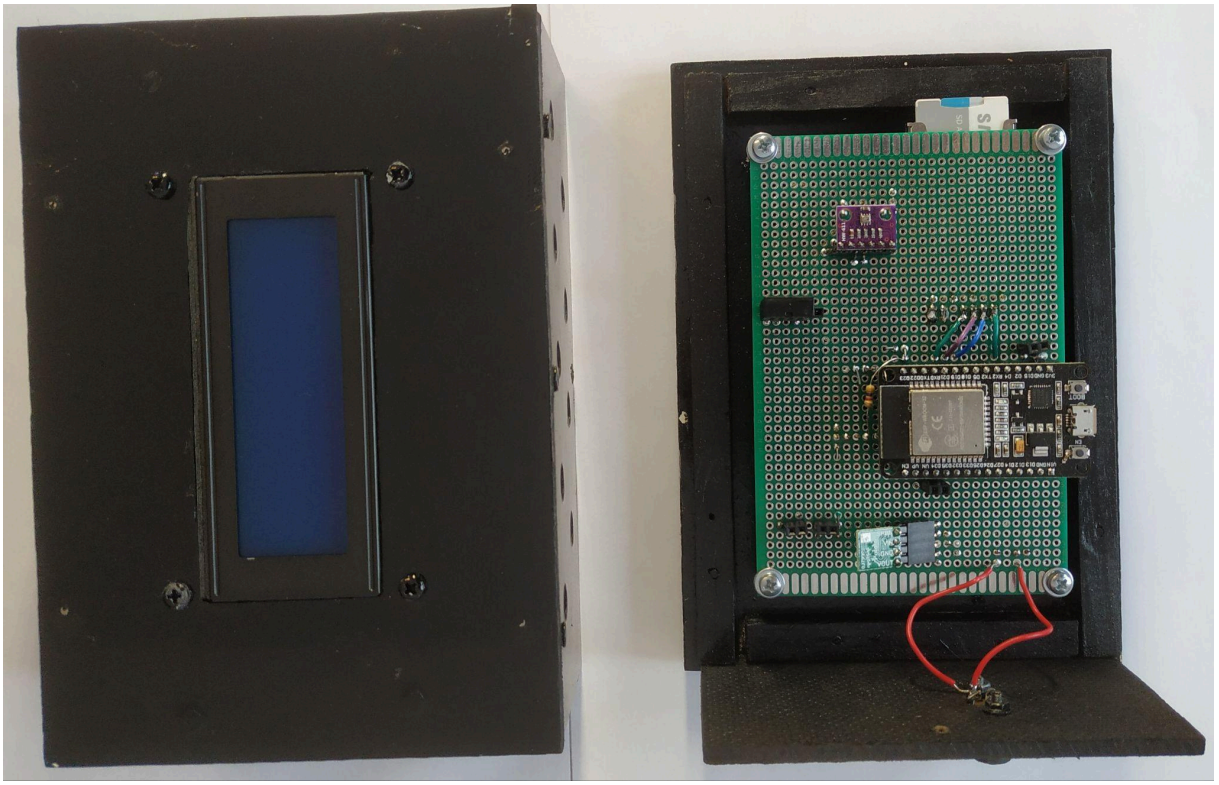
- AM2320 (czujnik temperatury i wilgoci)
- BMP280 (czujnik ciśnienia i temperatury)
- PCF8574 (adapter wyświetlacza LCD na interfejs I2C)

korzystały z protokołu I2C do komunikacji z urządzeniem nadrzędnym, a więc wszystkie zostały podłączone do tego samego interfejsu I2C ze względu na to że protokół I2C pozwala na podłączenie wielu urządzeń podrzędnych do jednego interfejsu [3]. Aby zapewnić prawidłowy interfejs I2C wspólne przewody (znane w protokole I2C jako SDA i SCL) należało podłączyć rezystorami, znanymi z języka angielskiego jako *pull-up resistors*, do napięcia zasilania (w tym przypadku 3.3 V), a więc użyto rezystorów o rezystancji 4700 Ω. Wszystkie urządzenia korzystające z protokołu I2C do komunikacji podłączono do zasilania o napięciu 3.3 V.

Aby zapewnić poprawne relatywne poziomy napięć w urządzeniu, wszystkie połączenia urządzeń z ziemią GND (ang. *ground*) dokonano poprzez zwarcie wszystkich przewodów GND.

3.1. Zdjęcia prototypu urządzenia

W tym rozdziale przedstawiono, i krótko opisano, zdjęcia wykonanego prototypu urządzenia.



Rys. 6 - Zdjęcie prototypu urządzenia - urządzenie otwarte

Na rysunku 6 przedstawiono zdjęcie prototypu urządzenia, gdy urządzenie jest otwarte. Ze względu na fakt że urządzenie korzysta z kart SD na której przechowywane są ustawienia oraz pliki z danymi możliwymi do wykorzystana przez użytkownika, urządzenie powinno być możliwe do otwarcia aby uzyskać dostęp do ww. karty SD. Urządzenie składa się z części górnej i dolnej.

Część górna urządzenia jest górną pokrywą urządzenia wraz z umiejscowionym w niej wyświetlaczem LCD.

Dolna część urządzenia jest platformą na której urządzenie jest postawione, do której przykręcona jest płytka prototypowa oraz w której umiejscowione jest gniazdo na wyjście zasilacza.

Prototyp urządzenia wykonano poprzez zlutowanie urządzeń na płytce prototypowej, gdzie urządzenia (z wyjątkiem czytnika karty SD) umiejscowiono na górnej części płytki prototypowej, a przewody połączające umiejscowiono na dolnej części płytki.

Rys. 7 - Zdjęcie prototypu urządzenia - urządzenie w trakcie pracy

Na rysunku 7 przedstawiono zdjęcie urządzenia pracującego. Na wyświetlaczu LCD widać aktualne dane zarejestrowane przez czujniki urządzenia.

Aby urządzenie stacji pogodowej odczytywało dane z atmosfery otaczającej urządzenie, a nie z atmosfery panującej w środku urządzenia, na bocznych ściankach urządzenia wywiercono dziury pozwalające na przepływ powietrza przez urządzenie.

4. Implementacja oprogramowania

Oprogramowanie było głównym elementem nad którym się skupiono w tym projekcie. Zdecydowaną większość pracy poświęcono implementacji oprogramowania urządzenia stacji pogodowej, co opisano w rozdziale 4.1, lecz oprócz tej części wykonano także aplikację konfiguracyjną której implementację opisano w rozdziale 4.2.

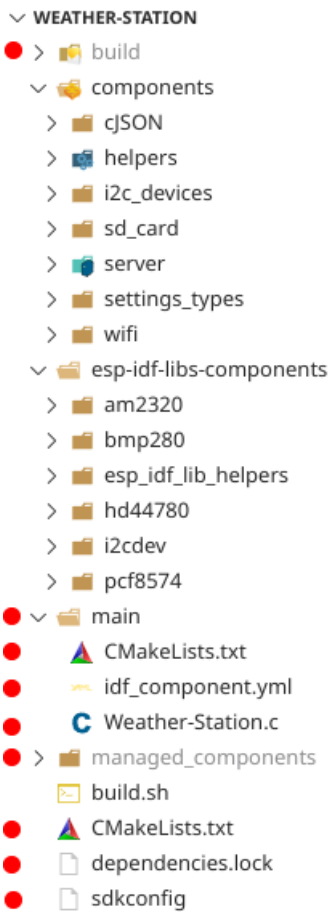
4.1. Implementacja oprogramowania urządzenia

Oprogramowanie urządzenia zostało wykonane używając mikrokontrolera ESP32 jako jednostki nadrzędnej, kontrolującej resztę systemu. Firma Espressif udostępnia framework *ESP-IDF* jako zbiór narzędzi do produkcji oprogramowania na ich urządzenia (mikrokontroler ESP32 jest produkowany przez firmę Espressif). Całość oprogramowania w tym projekcie które ma trafić na mikrokontroler ESP32 zostało wykonane właśnie w oparciu o ten ww. framework ESP-IDF. W trakcie prac korzystano często z dokumentacji [4] opisującej funkcjonalność i dostępne API frameworku ESP-IDF.

Projekt oprogramowania urządzenia zdecydowano się zaimplementować skupiając się na tworzeniu modułów, tzn.: dla każdej logicznie spójnej funkcjonalności tworzyło się moduł (opis modułów w rozdziale 4.1.3) który był odpowiedzialny za daną funkcjonalność. Przykładowo moduł *sd_card* był modułem odpowiedzialnym za implementację funkcjonalności związanej z pracą z kartą SD, a wszystkie funkcje udostępniane przez ten moduł miały przedrostek *sd_card_**. Takie podejście modułowe można uznać za pewnego rodzaju emulację programowania obiektowego w języku C.

4.1.1 Struktura plików projektu

Framework ESP-IDF został użyty do inicjalizacji projektu, co stworzyło główną strukturę projektu wraz z podstawowymi plikami. W trakcie prac naturalnie dodano dodatkowe katalogi i pliki źródłowe, gdzie finalną wersję struktury projektu pokazano na rysunku 8.



Rys. 8 - Struktura plików projektu

Pliki i katalogi zaznaczone czerwoną kropką po lewej stronie rysunku są plikami / katalogami wygenerowanymi automatycznie, a reszta została stworzona manualnie.

Katalog *build* jak sama nazwa wskazuje był katalogiem w którym przechowywane były pliki „budownicze" projektu generowane przez komendę *idf.py build*.

Katalog *components* był katalogiem stworzonym do przechowywania modułów (ang. *components*) stworzonych na potrzeby projektu. Dalsze omówienie modułów zostało wykonane w rozdziale 4.1.3.

Katalog *esp-idf-lib-components,* podobnie jak katalog *components*, przechowywał moduły wykorzystane w projekcie, aczkolwiek wszystkie moduły tego katalogu były modułami pozyskanymi z biblioteki *esp-idf-lib* (na podstawie otwartych licencji, zob. rozdział 2.3).

Katalog *main* był katalogiem zawierającym główny plik źródłowy projektu *Weather-Station.c*, w którym znajdowałą się funkcja wejściowa *main()*.

Katalog *managed_components* był katalogiem, ze źródłami, automatycznie stworzonym po dodaniu oficjalnego modułu ESP-IDF: *mDNS*.

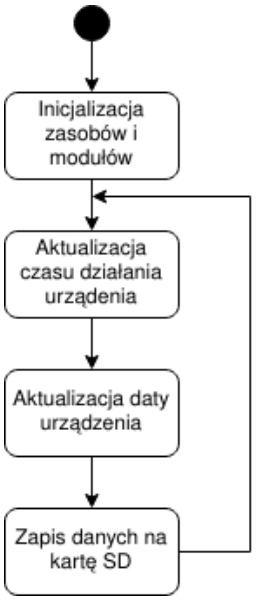
4.1.2 Główny plik źródłowy projektu: Weather-Station.c

Plik *Weather-Station.c* służył jako plik główny projektu, tj.: plik z funkcją wejściową programu *main()*. Ponieważ projekt oprogramowania urządzenia wykonano skupiając się głównie na modułach funkcjonalności, plik główny *Weather-Station.c* składa się z dwóch części: inicjalizacyjnej i zapisywania danych, oraz dwóch funkcji pomocniczych.

Część inicjalizacji zasobów i modułów jest pierwszą częścią programu urządzenia, i polega na inicjalizacji następujących rzeczy:

- Zasobów i funkcjonalności systemowych (Non-Volatile Storage, interfejsu sieciowego, itd.),
- Modułów przedstawionych w katalogu *components* na rysunku 8.

Moduł zarządzający urządzeniami I2C i w szczególności wyświetlaczem LCD jest inicjalizowany najwcześniej w programie jak tylko jest to możliwe, ponieważ każda inicjalizacja zwraca kod błędu i na podstawie kodu błędu (lub jego braku) możliwe jest wyświetlenie na LCD napotkania błędu podczas włączania urządzenia.



Rys. 9 - Diagram sekwencji działania pliku źródłowego Weather-Station.c

Po poprawnie zakończonej inicjalizacji urządzenia wątek główny przechodzi do funkcjonalności dotyczącej aktualizacji czasu działania i daty urządzenia oraz zapisywania danych na karcie SD. Więcej o wątkach w programie jest napisane w rozdziale 4.1.4.

4.1.3 Opis modułów projektu

Projekt oprogramowania urządzenia skupia się na wykorzystaniu modułów, nazwanych w dokumentacji ESP-IDF jako *component*, do podziału funkcjonalności. Każdy moduł zaprojektowany jest zgodnie z wymaganiami co do modułów przedstawionymi przez dokumentację ESP-IDF [5], które są następujące:

- plik CMakeLists.txt,
- plik nagłówkowy *.h*,
- plik źródłowy *.c*.

Moduły w ten sposób zadeklarowane zostaną skompilowane i zalinkowane w procesie budowy programu jako biblioteki statyczne projektu. Takie wykorzystanie modułów nie tylko pozwala na podzielenie projektu na pliki, ale też podzielenie danych plików źródłowych na osobne moduły będące odpowiedzialne za jedną funkcjonalność, dzięki czemu ułatwia się proces pracy nad oprogramowaniem - gdy szuka się danej funkcjonalności lub chce się ją zmodyfikować – wiadomo gdzie szukać odpowiedniej deklaracji funkcji lub jej implementacji. Poniższa tabela 7 przedstawia stworzone moduły w projekcie i ich funkcjonalność.

Tabela 7 - Opis modułów oprogramowania urządzenia

Nazwa modułu	Opis
cJSON	Moduł z kodem źródłowym biblioteki cJSON, pozyskanej na podstawie otwartej licencji MIT.
helpers	Moduł pomocniczy z funkcjami niepasującymi do żadnego innego modułu.
i2c_devices	Moduł odpowiedzialny za funkcjonalność I2C, w tym: ● inicjalizację urządzeń i zasobów I2C, ● pracę z wyświetlaczem LCD, ● wątek odczytujący dane urządzeń I2C i aktualizujący LCD.
sd_card	Moduł odpowiadający za pracę z kartą SD.

server	Moduł odpowiadający za funkcjonalność serwera HTTP.
settings_types	Moduł do pracy ze strukturą przedstawiającą ustawienia urządzenia.
wifi	Moduł odpowiadający za funkcjonalność WiFi i wątku monitorującego stan połączenia WiFi.

Wyżej przedstawione moduły (poza biblioteką *cJSON* i modułem *helpers*) oferują podobną konwencję co do ich funkcjonalności:

- moduł udostępnia funkcję *<nazwa_modułu>_init()* do inicjalizacji modułu i zasobów modułu,
- moduł udostępnia funkcje zgodne z zakładaną funkcjonalnością modułu, gdzie każda nazwa funkcji posiada nazwę według formatu *<nazwa_modułu><nazwa_funkcji>()*.

4.1.4 Opis sposobu działania programu

Cały program urządzenia polega na wykorzystaniu funkcjonalności FreeRTOS (biblioteki systemu operacyjnego czasu rzeczywistego) do implementacji programu z wieloma wątkami wykonującymi przypisanym im zadania, zamiast tworzenia oprogramowania z tzw. super-pętlą (ang. *super loop*). Poprzez wykorzystanie wielu wątków w programie jest się w stanie kontrolować priorytetami zadań, podzieleniu programu na logiczne wątki z odrębnymi funkcjonalnościami czy łatwiejszą kontrolą zarządzania okresem wykonania wybranych zadań / wątków.

Listę wątków, ich priorytetów i opisów umieszczono w tabeli 8.

Tabela 8 - Lista wątków w programie urządzenia

Wątek	Priorytet	Opis
Główny	1	Wątek do aktualizacji czasu działania i lokalnego czasu urządzenia oraz do okresowego zapisywania danych na karcie SD.
I2C	3	Wątek tworzony do okresowego odczytywania danych z sensorów

		i do okresowego odświeżania informacji na wyświetlaczu LCD.
WiFi	2	Wątek tworzony do monitorowania siły połączenia WiFi (RSSI).
Serwer HTTP	-	Wątek tworzony przez moduł HTTP Server frameworku ESP-IDF. Nie znaleziono informacji o priorytecie tego wątku.
mDNS	1	Watek tworzony przez moduł mDNS frameworku ESP-IDF

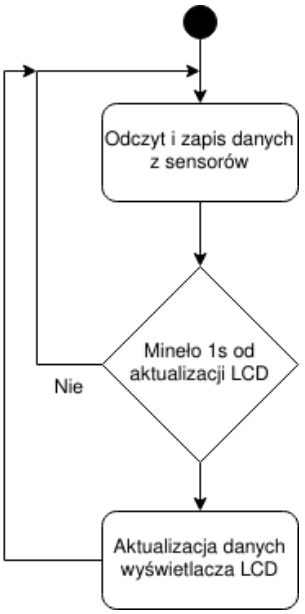
Zadania i diagram sekwencji działania wątku głównego przedstawia rysunek 9.

Diagram sekwencji działania wątku I2C przedstawia rysunek 10, przedstawiony obok.

Wątek I2C odczytuje dane z obu (AM2320 i BMP280) sensorów a następnie w zależności od ustawień urządzenia wybiera dane jednego z nich (lub ich średnią) do zapisu jako najnowsze odczytane dane. Po odczytaniu i zapisie danych sensorów wątek kontroluje czy należy odświeżyć wyświetlacz LCD – korzystając z „czasomierza" programowego, który działa na podstawie inkrementacji zmiennej mówiącej ile razy wykonało się zbieranie danych z sensorów i jeżeli ta wartość będzie większa od wartości obliczonej z wzoru:

1000 [ms] / okres zbierania danych z sensorów [ms]

to wątek przechodzi do aktualizacji wyświetlacza LCD. Na koniec każdej pętli wątek przechodzi w stan uśpienia na czas okresu zbierania danych.



Rys. 10 - Diagram sekwencji działania wątku I2C

Wątek WiFi polega tylko na testowaniu siły połączenia WiFi i zapisywaniu odczytanych danych a następnie do przechodzenia w stan uśpienia równy czasowi ustalonemu w ustawieniach urządzenia: okresu testowania siły połączenia WiFi.

Wątki Serwera HTTP i *mDNS* są tworzone przez moduły ESP-IDF i nie ma wglądu wewnątrz tych wątków.

4.1.5 Informacje dot. kodu źródłowego stron internetowych

W projekcie oprogramowania urządzenia moduł serwera HTTP wykorzystuje strony internetowe jako interfejs graficzny urządzenia. Poniżej opisano sposób implementacji tych stron internetowych w kodzie C, gdyż nie jest to język do końca przystosowany do technologii internetowych i ich implementacja nie jest intuicyjna.

Gdy zostanie wykonane zapytanie HTTP typu GET na odpowiedni adres URL serwer uruchamia funkcję która została do tego adresu przypisana, która ma za zadanie zwrócić informacje nt. strony (jej kod źródłowy i ewentualnie metadane protokołu HTTP). Serwer oferuje następujące adresy URI:

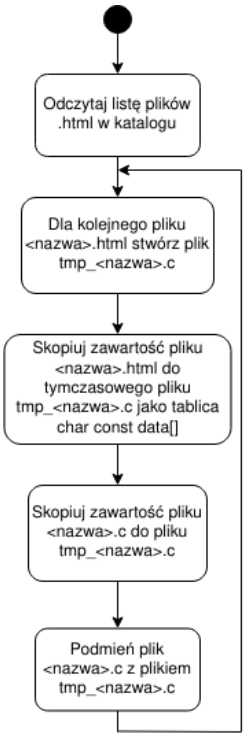
- "/" - strona główna interfejsu urządzenia,
- "/device_info" - strona z informacjami nt. urządzenia,
- "/settings" - strona z ustawieniami urządzenia.

Serwer oferuje też adresy URI do interfejsu REST API, które zostały przedstawione w rozdziale 5.1. Każda funkcja odpowiedzialna za obsługę żądania HTTP składa się z dwóch głównych części:

- wysyłania części statycznej strony,
- wysyłania części dynamicznej strony.

Wysyłanie części statycznej strony polega na wysłaniu do klienta danych strony internetowej które są statyczne - nie zmieniają się, i są to dane HTML i CSS (w niektórych przypadkach też predefiniowane funkcje JavaScript) które określają architekturę strony internetowej. Początkowo te strony internetowe budowane są w plikach *.html* (z wbudowanymi stylami CSS i skryptami JavaScript w plik HTML) dla wygody testowania wyglądu strony, ale dane tych plików muszą być dostępne z poziomu skompilowanego programu języka C. Do uzyskania tego efektu napisano prosty skrypt w języku Python który kopiuje wszystkie dane wybranego pliku HTML i na ich podstawie otwiera odpowiedni (wybrany na podstawie nazwy) plik źródłowy języka C w którym istnieje funkcja która ma zwrócić te dane, i wpisuje do tego pliku źródłowego dane HTML w postaci tablicy znaków (`char const data[]`). Gdy funkcja obsługująca żądania na dany adres URI ma dostęp do danych strony internetowych w postaci tablicy znaków może je wysłać do klienta jako jeden kawałek (ang. *chunk*) danych protokołu HTTP v1.1. Należy tutaj wspomnieć że funkcją kluczową tego

mechanizmu jest wykorzystanie słowa kluczowego `const` w deklaracji tej tablicy znaków. Użycie tego słowa kluczowego zapewnia że te dane zostają umiejscowione w pamięci trwałej flash urządzenia, a gdyby tego słowa kluczowego nie użyto te dane zostałyby wczytywane do pamięci RAM urządzenia, co bardzo szybko wykorzystywałoby dostępną pamięć urządzenia, gdyż na jedną strone internetową przypada około od +/- 4 kB do 16 kB danych. Obrano konwencję: na jedną stronę internetową przysługuje jeden plik HTML i jeden plik źródłowy języka C. Diagram sekwencji działania skryptu kopiującego dane HTML do plików źródłowych C przedstawiono na rysunku 11.



Rys. 11 - Diagram sekwencji działania skryptu *parser.py*

Ważnym elementem skryptu parser.py przedstawionego na rysunku 11 jest 3 krok w pętli: kopiowanie zawartości pliku <nazwa>.c do nowego pliku z danymi HTML. Jest to ważny krok gdyż bez niego za każdym razem gdy skrypt generuje plik źródłowy C z danymi HTML tracony byłby kod źródłowy funkcji odpowiadających za obsługę żądań dot. tej strony internetowej.

Przykład wysyłania danych statycznych (tablicy znaków z danymi HTML: `home_data`) głównej strony internetowej przedstawia instrukcja:

```
httpd_resp_send_chunk(req, home_data, HTTPD_RESP_USE_STRLEN);
```

Ponieważ dane źródłowe HTML stron internetowych zostały zadeklarowane jako stałe (`const`) to nie można ich manipulować z poziomu programu w czasie wykonania a więc dane dynamiczne, jak przykładowo: dane pogodowe, muszą być wysłane w inny sposób. Do wysyłania danych dynamicznych (dostępnych tylko w czasie wykonania programu) i zapełnienia nimi stron internetowych wykorzystano funkcjonalność JavaScript. W czasie wykonania - w momencie działania funkcji obsługującej żądanie HTTP odczytywane są dane dynamiczne (przykładowo dane pogodowe) a następnie wysyłany jest kawałek (chunk) danych skryptu strony który po wykonaniu zmieni zawartość tekstową wybranego elementu HTML. Przykład przedstawiono poniżej:

```
char str[256] = {0};
sprintf(
    str,
    "document.getElementById('temp').textContent = '%.1f °C';\n"
    "document.getElementById('humd').textContent = '%.1f %%';\n"
    "document.getElementById('pres').textContent = '%d hPa';\n"
    ,
    home_page_device_data→i2c_daq_data→temperature_C,
    home_page_device_data→i2c_daq_data→humidity_P,
    (int)(home_page_device_data→i2c_daq_data→pressure_Pa) / 100
);
result = httpd_resp_send_chunk(req, str, HTTPD_RESP_USE_STRLEN);
```

Do pomocniczej tablicy znaków `str` zapisane zostają funkcję Javascript które zmieniają zawartość elementów HTML strony internetowych przedstawiajacych temperaturę, wilgotność i ciśnienie, a następnie ta tablica pomocnicza zostaje wysłana jako jeden kawałek danych strony internetowych. Gdy te instrukcje JavaScript zostaną wpisane bezpośrednio do pola `<script>` bez bycia osadzonymi w żadnej funkcji to zostaną one wykonane jak tylko strona internetowe je odczyta / dostanie od serwera, a więc za każdym razem. Wszystkie dane dynamiczne dostępne tylko w czasie wykonania programu zostają wysłane w podobny sposób: wykorzystuje się elementy HTML z tymczasowymi - zastępczymi danymi, których zawartość zostaje zamieniona przy pomocy instrukcji JavaScript.

4.1.6 System budowy projektu

Przedstawiany projekt oprogramowania urządzenia, jak i sam framework ESP-IDF, korzysta z systemu automatyzacji budowania CMake. Każdy moduł w projekcie (zarówno te w katalogu *components* jak i moduł główny *main*) potrzebuje plik CMakeLists.txt w którym deklaruje się moduł korzystając z funkcji *idf_component_register()* w której deklaruje się pliki źródłowe modułu, lokalizację pliku nagłówkowego modułu oraz modułów (zależności) od których deklarowany moduł zależy. Ta ostatnia pozycja jest szczególnie ważna gdyż jawne zadeklarowanie zależności w procesie budowania zapewnia że budowany moduł poprawnie zostanie zalinkowany w procesie kompilacji, gdyż moduły budowane są do bibliotek statycznych które i tak muszą być zalinkowane, a ich źródła nie są bezpośrednio kompilowane razem z modułem który na nich zależy. Dodatkowo można wspomnieć że framework ESP-IDF implementuje wiele różnych dodatkowych funkcjonalności w systemie CMake, które nie są częścią standardowego zestawu funkcjonalności CMake. Funkcja *idf_component_register()* jest przykładem takiej funkcjonalności.

4.1.7 Inne ważne informacje nt. projektu oprogramowania

W projekcie przyjęto konwencję deklaracji funkcji zgodnie z konwencją przyjętą we frameworku ESP-IDF, przedstawioną poniżej przy pomocy funkcji która ma dodawać dwie liczby:

```
esp_err_t add(int a, int b, int * out_result)
```

Funkcja przyjmuje argumenty normalnie jak, w zależności od wymogów stawionych przez programistę ale wynik nie jest zwracany jako wartość zwracana przez funkcję, tylko jest zapisywany pod adres podany do funkcji przez wskaźnik jako adres zmiennej gdzie ten wynik ma zostać zapisany. Takiego sposobu podejście do tworzenia funkcji pozostawia nieużywaną wartość zwracaną przez funkcję co z kolei zostaje wykorzystane do zwracania kodów błędu. Każda funkcja zwraca wartość typu *esp_err_t* co jest enumeracją języka C przedstawiającą różnego rodzaju kody błędów, lub kod powodzenia (enumeracja ESP_OK), a więc użytkownik zawsze jest w stanie jawnie się dowiedzieć czy wywołana przez niego funkcja odbyła się poprawnie lub nie.

W projekcie jako że występują różne wątki pracujące na tych samych danych należało wykonać jakiś mechanizm wymiany informacji między wątkami. Zaimplementowanym rozwiązaniem były struktury języka C chronione przez mutex'y (a dokładniej binarne semafory) biblioteki FreeRTOS. Zostanie to wyjaśnione na przykładzie modułu *wifi*. Przy inicjalizacji modułu *wifi* do funkcji inicjalizującej *wifi_init* należy podać wskaźnik na strukturę *wifi_state_t* do której zapisywane zostaną aktualne dane dotyczące stanu połączenia wifi. Strukturę tę przedstawiono poniżej:

```
typedef struct {

    bool     is_connected;
    bool     got_ip;
    int32_t  wifi_rssi;
    char     ip[16];
    SemaphoreHandle_t mutex;
} wifi_state_t;
```

Struktura składa się z pól danych oraz z „uchwytu" (ang. *handle*) do mutexu który ma chronić dane w tej strukturze przed wystąpieniem wyścigu danych (ang. *data race*). Gdy funkcja *wifi_init* zainicjalizuję wątek odpowiadający za okresowe sprawdzanie stanu połączenia wifi, dwa różne wątki mogą jednocześnie do tej struktury próbować pisać lub odczytywać dane, ale poprzez wykorzystanie mutex'ów jako mechanizmów wzajemnego wykluczania ten problem zostaje rozwiązany gdyż tylko jeden wątek który przejął ten mutex może korzystać z tych danych.

Tego typu struktury wykorzystuje się w programie do przesyłania informacji o:

- stanie połączenia wifi,
- stanie ustawień urządzenia,
- czasie pracy i lokalnym czasie urządzenia,
- aktualnych danych odczytanych z sensorów.

Oczywiście takiego typu rozwiązanie może wprowadzić program w tzw.: zakleszczenie (ang. *deadlock*) i żeby zapewnić że zakleszczenie w programie się nie pojawi przyjęto następującą strategię: jeżeli dana funkcja posiada własność jednego mutex'a to musi go zwolnić zanim spróbuje zawłaszczyć innego mutex'a.

W programie w wątku odczytującym dane z sensorów i zapisującym je do struktury `i2c_devices_daq_task_config_t` następuje komunikacja szeregowa z sensorami

zewnętrznymi z punktu widzenia mikrokontrolera. Ponieważ w programie występuje kilka wątków to mogłaby się stać sytuacja że inny wątek dostałby czas procesora w trakcie gdy wątek komunikujący się z sensorami był w trakcie komunikacji, co przerwałoby tą komunikację na przynajmniej jeden takt planisty (ang. *scheduler*) FreeRTOS, aczkolwiek ponieważ wykorzystywane są w tym oprogramowaniu tylko protokoły synchroniczne (I2C i SPI) do komunikacji z sensorami zewnętrznymi taka sytuacja nie stanowi zagrożenia poprawnego działania systemu.

4.2. Implementacja oprogramowania aplikacji konfiguracyjnej PC

Aplikacja konfiguracyjna na komputery klasy PC została stworzona na podstawie frameworku Electron. Dwoma głównymi powodami dlaczego wybrano takie podejście było:

- framework Electron pozwala na szybkie i łatwe tworzenie aplikacji desktopowych na różne platformy,
- użycie frameworku Electron pozwala na wykorzystanie wcześniej stworzonej strony ustawień interfejsu internetowego urządzenia, co pozwala na zaoszczędzenie pracy.

Projekt aplikacji stworzony przy pomocy frameworku Electron tworzy automatycznie wiele różnych plików potrzebnych do pracy frameworku, ale czterema wartymi uwagi plikami źródłowymi które zostały stworzone manualnie są:

- app_start.js,
- main.css,
- main.html,
- main.js.

Plik źródłowy *„app_start.js”* jest skryptem w języku JavaScript i jest punktem wejściowym, wywoływanym przez framework Electron po uruchomieniu aplikacji. Ten skrypt odpowiada za dwie podstawowe funkcjonalności:

- tworzy okno aplikacji na podstawie pliku main.html,
- odpowiada za zapisanie danych przekazanych skryptowi przez skrypt main.js, do pliku settings.txt.

Plik main.html podobnie jak w tworzeniu stron internetowych odpowiada za deklarację struktury strony będącej interfejsem aplikacji, a plik main.css, zawierany przez plik main.html, odpowiada za styl / formatowanie elementów zadeklarowanych w pliku main.html.

Plik main.js, również zawierany przez plik main.html, jest głównym plikiem skryptowym, odpowiadającym za logikę interfejsu aplikacji. W tym pliku zaimplementowane zostały 3 różne funkcje:

- anonimowa część skryptu:
    - odpowiada za ustalanie kolorów tła aplikacji,
    - polega na zmienianiu tła aplikacji co 100 ms używając funkcji CSS *linear-gradient* do stworzenia gradientu, na podstawie wcześniej zadeklarowanej tablicy kolorów,
- accordion_toggle():
    - funkcja która jest wywoływana przez kliknięcie na przyciski INSTRUCTIONS, BASIC SETTINGS lub ADVANCED SETTINGS, która odpowiada za implementację logiki która pozwala ww. przyciskom na zachowanie się jak sekcje akordeonowe - pokazujące lub chowające podrzędną im zawartość,
- create_settings_file():
    - funkcja która zbiera informacje wprowadzone do pól ustawień, tworzy z tych informacji plik w formacie JSON i przesyła ten plik do skryptu *app_start.js* aby został on zapisany w miejscu podanym przez użytkownika.

5. Przykład użycia i instrukcja obsługi

Ten rozdział poświęcono przedstawieniu przykładowego użycia urządzenia i aplikacji konfiguracyjnej, które przedstawi wykorzystanie każdej oferowanej przez urządzenie funkcjonalności i jednocześnie przedstawi oferowany przez urządzenie i aplikację konfiguracyjną interfejs graficzny. Przedstawiony w tym rozdziale przykład użycia ma za zadanie też służyć jako instrukcja obsługi urządzenia, jego interfejsu graficznego i aplikacji konfiguracyjnej.

Ze względu na to że interfejs strony internetowej urządzenia przystosowany jest do monitorów komputerów klasy PC oraz do wyświetlaczy telefonów, rysunki przedstawiające interfejs zaprezentowane w tym rozdziale przedstawiają interfejs graficzny strony internetowej w wersji mobilnej (dostępnej z przeglądarki internetowej na telefonach), gdyż rysunki przedstawiające interfejs widoczny na komputerach klasy PC byłyby zbyt duże aby mogły być zaprezentowane w tym rozdziale, ale zostały one dołączone do pracy i przedstawione w rozdziale 8 (rysunki 17 do 19).

5.1. Przykład użycia

Pierwszą czynnością aby użyć urządzenie należy stworzyć plik z ustawieniami które zostaną odczytane przez urządzenie. Do tworzenia pliku z ustawieniami należy użyć aplikacji „*weather-station-config-app*” dostępnej na platformy:

- Darwin (MacOS),
- Linux,
- Windows.

Po uruchomieniu aplikacji pokaże się interfejs zaprezentowany poniżej na rysunku 11.

Rys. 12 - Interfejs aplikacji konfiguracyjnej

Aplikacja posiada 4 przyciski:

- **INSTRUCTIONS** - przycisk otwiera krótki opis aplikacji i instrukcje,
- **BASIC SETTINGS** - przycisk otwiera podstawowe ustawienia,
- **ADVANCED SETTINGS** - przycisk otwiera zaawansowane ustawienia,
- **Create Settings File** - przycisk odczytuje wprowadzone dane i tworzy na ich podstawie plik z ustawieniami.

Część **BASIC SETTINGS** pozwala na:

- ustawienie nazwy (WiFi SSID) i hasła (WiFi Password) sieci WiFi do której urządzenie ma się podłączyć po uruchomieniu,
- ustawienie strefy czasowej w której urządzenie będzie się znajdować.

Część **ADVANCED SETTINGS** pozwala na:

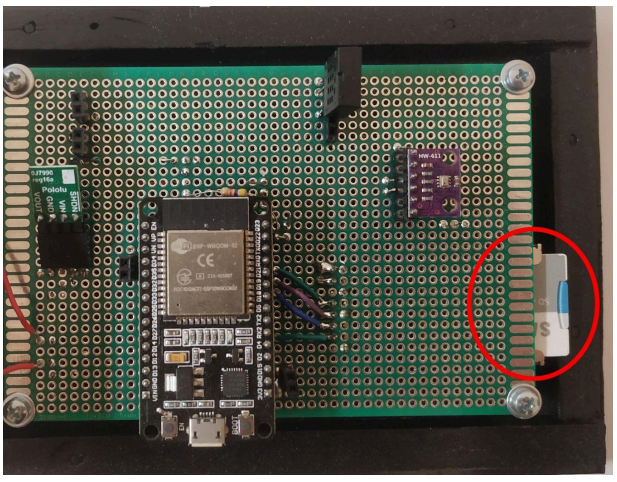
- ustawienie nazwy urządzenia (mDNS Hostname), które pozwala na dostęp do urządzenia poprzez wykorzystanie technologii mDNS, przykładowo: dla nazwy *„esp”* można uzyskać dostęp do strony internetowej urządzenia poprzez wprowadzenie

adresu *esp.local* zamiast adresu IP urządzenia, który może być trudny do znalezienia, domyślnie nazwa jest ustawiona na „*weatherstation*”,

- ustawienie poziomu logowania komunikatów (Serial Logging Verbosity) przy połączeniu seryjnym (przez kabel USB) komputer – mikrokontroler, domyślnie ustawione jest na poziom „*Debug*”,
- ustawienie okresu testowania siły połączenia WiFi (WiFi RSSI test period), domyślnie wartość 60 sekund,
- ustawienie okresu pozyskiwania danych z czujników, domyślnie 1 sekunda,
- ustawienie wyboru którego czujnika (Device’s temperature sensor select) należy użyć do pozyskiwania informacji o temperaturze, domyślnie wybrano opcję średniej z dwóch urządzeń (odczytana temperatura jest średnią odczytanej temperatury z dwóch czujników: AM2320 i BMP280),
- ustawienie okresu zapisywania danych z czujników (Data Write Interval) na kartę SD.

Po wprowadzeniu ustawień i kliknięciu przycisku *„Create Settings File*” wyświetli się okno dialogowe pozwalające wybrać lokację gdzie zapisać plik z ustawieniami, o nazwie *„settings.txt”*.

Po utworzeniu pliku z ustawieniami należy przenieść ten plik na kartę SD, która będzie wkładana do czytnika kart SD w urządzeniu oraz otworzyć urządzenie. Lokalizację czytnika kart SD przedstawiono na rysunku 12.



Rys. 13 - Miejsce włożenia karty SD do urządzenia zaznaczone na czerwono

Po otwarciu urządzenia i włożeniu karty SD do czytnika należy zamknąć urządzenie. Następnie powinno się włożyć końcówkę wyjścia zasilacza do gniazda zasilania urządzenia oraz podłączyć zasilacz do gniazdka sieci co załączy urządzenie.

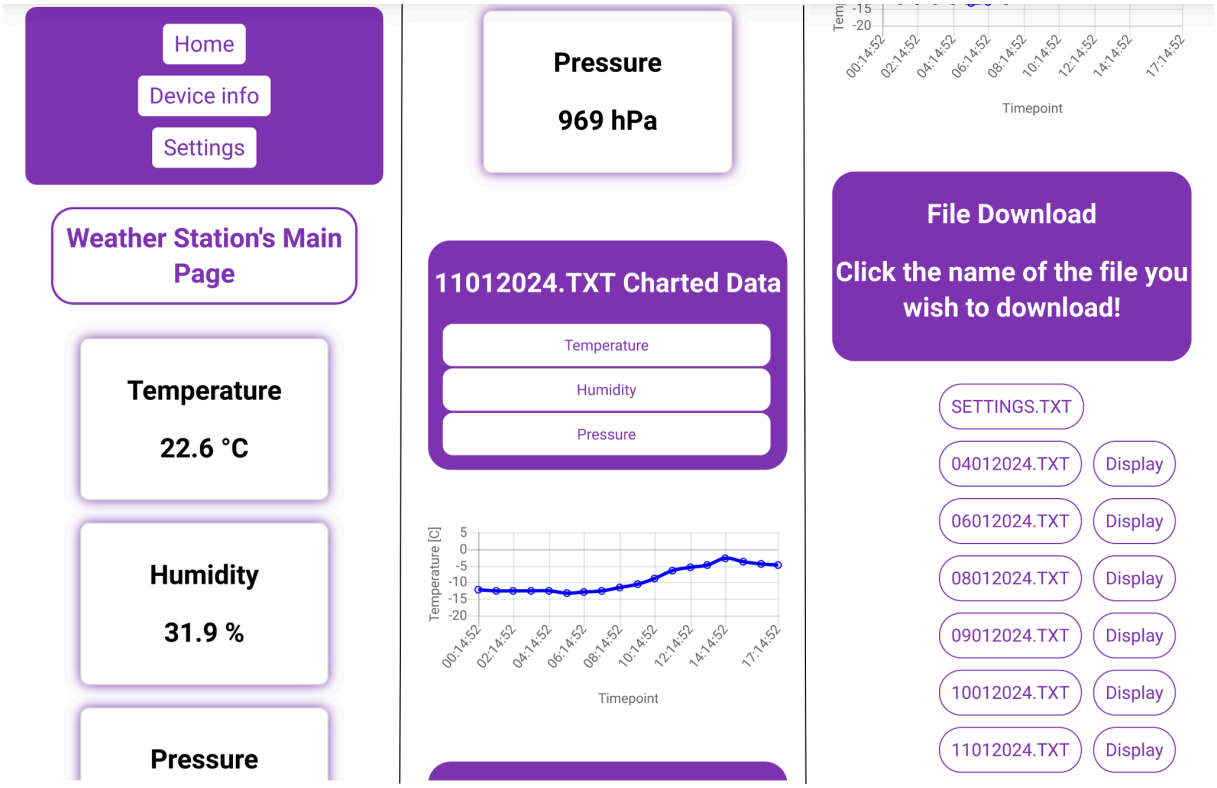
Po załączeniu się urządzenia na ekranie LCD wyświetli się jedna z dwóch możliwych informacji:

- ekran główny urządzenia (widoczny na rysunku 7), lub
- ekran wyświetlający błąd (przykładowa wiadomość o błędzie przedstawiona na rysunku 14). Informacje na temat błędów zostały podane w rozdziale 5.2.

Rys. 14 - Wyświetlacz LCD z wiadomością o błędzie

Informację o tym czy urządzeniu udało się połączyć z siecią WiFi jest wyświetlenie (lub jego brak) aktualnej daty i czasu w pierwszej linii wyświetlacza LCD, gdyż aktualny czas i data urządzenia synchronizowane jest poprzez internetowy protokół SNTP. Gdy aktualny czas i data nie zostaną wyświetlone na ekranie LCD urządzenia przez czas dłuższy niż kilka minut, można to uznać za niepowodzenie w połączeniu się z siecią WiFi.

Gdy zweryfikuje się że urządzenie poprawnie się uruchomiło i połączyło z siecią WiFi i/lub internetem, można przejść do uzyskiwania dostępu do interfejsu urządzenia przez stronę internetową oferowaną przez urządzenie. Aby uzyskać dostęp do ww. strony należy w przeglądarce wpisać adres <*nazwa urządzenia*>.local jeżeli dana przeglądarka lub platforma posiada funkcjonalność mDNS, lub adres ip urządzenia.



Rys. 15 - Interfejs mobilnej strony internetowej głównej urządzenia

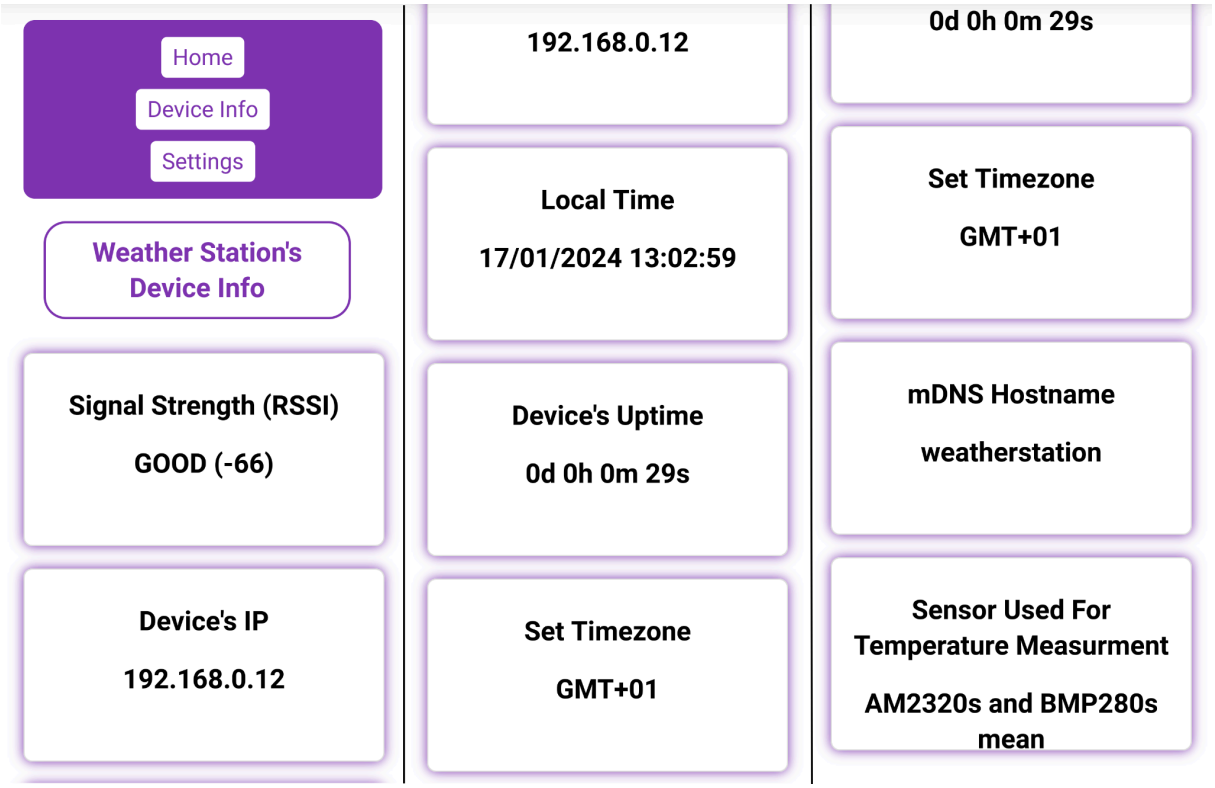
Po uzyskaniu dostępu do strony głównej, widocznej na rysunku 15, można zobaczyć:

- panel nawigacyjny (dostępny z poziomu każdej strony),
- aktualny stan czujników urządzenia,
- wykres (do wyboru, jeden wykres naraz): temperatury, wilgoci lub ciśnienia,
- listę plików, gdzie:
    - naciśnięcie na nazwę pliku powoduje jego pobranie,
    - Naciśnięcie na przycisk “Display” dostępny przy nazwie danego pliku powoduje pokazanie zawartości danych tego pliku na wykresach.

Należy pamiętać że każda nazwa pliku (z wyjątkiem pliku settings.txt) jest nazwą w formacie:

„<dzień><miesiąc><rok>.txt”

a więc przykładowy plik o nazwie „26062024.*txt*” jest plikiem którego zawartość to dane pogodowe (w formacie CSV) odczytane w trakcie dnia 26 lipca 2024r.

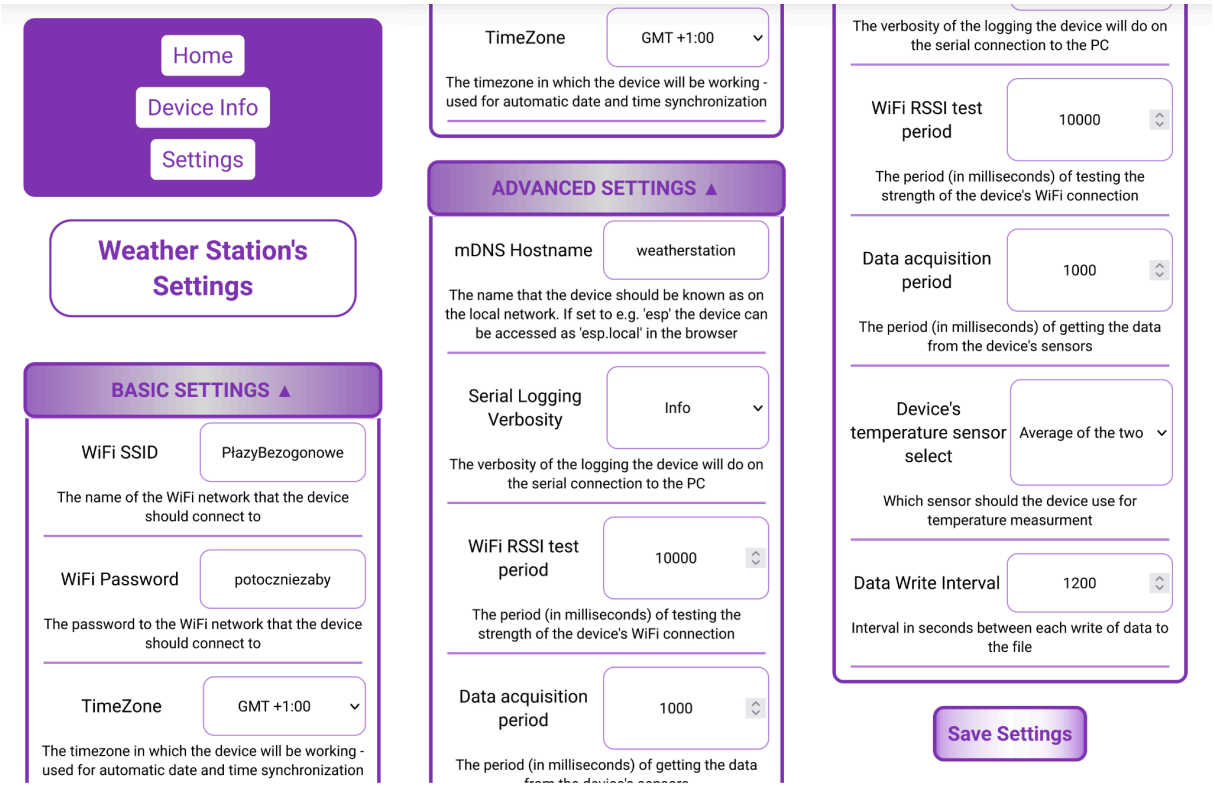


Rys. 16 - Interfejs mobilnej strony internetowej urządzenia z informacjami o urządzeniu

Po przejściu na stronę z informacjami o urządzeniu (Device info) można zauważyć niektóre dane dot. urządzenia:

- siła połączenia WiFi (Signal Strength (RSSI)),
- adres IP (Device's IP) urządzenia,
- lokalna data urządzenia (Local Time),
- czas działania urządzenia (Device's Uptime),
- ustawiona strefa czasowa (Set Timezone),
- nazwa urządzenia (mDNS Hostname),
- wybór czujnika temperatury (Sensor Used for Temperature Measurement) wybranego do pomiaru temperatury.

Dane te aktualizują się tylko przy ponownym wczytaniu strony.



Rys. 17 - Interfejs mobilnej strony internetowej urządzenia z ustawieniami

Po przejściu na stronę ustawień (Settings) można zobaczyć interfejs przedstawiony na rysunku 16. Interfejs i cel tej strony jest taki sam jak interfejs i cel aplikacji konfiguracyjnej, z tą różnicą że po wprowadzeniu / zmianie ustawień na tej stronie i kliknięciu przycisku „*Save Settings*" następuje automatyczne ponowne uruchomienie urządzenia, o czym informuje alert w przeglądarce.

Aby skorzystać z interfejsu REST API urządzenia należy wykonać zapytanie HTTP (funkcja GET) na jeden z następujących URI:

- <adres>/api/data/current_sensors – zapytanie zwraca plik w formacie JSON z aktualnymi danymi pogodowymi urządzenia,
- <adres>/api/data/settings – zapytanie zwraca plik w formacie JSON z aktualnymi ustawieniami urządzenia,
- <adres>/api/files/list – zapytanie zwraca plik w formacie JSON z listą nazw plików z zawartością: historycznych danych pogodowych, dostępnymi do pobrania,

- <adres>/api/files/download?filename=<*nazwa_pliku*> - zapytanie zwraca zawartość wybranego pliku (wybór pliku następuje poprzez wpisanie nazwy pliku w pole <*nazwa_pliku*>) w formacie CSV.

5.2. Lista błędów wyświetlanych na LCD

Poniżej zostanie zaprezentowana lista błędów, wraz z ich wyjaśnieniami i potencjalnymi rozwiązaniami, które mogą zostać wyświetlone na ekranie LCD. Wszystkie przedstawione w tabeli 9 błędy poza byciem wyświetlonym na ekranie LCD logowane są także na połączeniu szeregowym mikrokontroler – komputer, używając interfejsu USB mikrokontrolera.

Tabela 9 - Lista błędów wyświetlanych na ekranie LCD urządzenia

Kod błędu	Opis błędu	Potencjalne rozwiązania
Software 1	Nie udało się zainicjalizować mutex'u struktury basic_settings_t przechowującej ustawienia urządzenia	● Należy zresetować urządzenie
Software 2	Nie udało się zainicjalizować mutex'u struktury uptime_t przechowującej dane o czasie	● Należy zresetować urządzenie

Software 3	Nie udało się wczytać / przeczytać pliku z ustawieniami settings.txt	● Należy sprawdzić czy plik settings.txt istnieje na karcie SD i ● Należy sprawdzić czy format pliku settings.txt (format JSON) jest poprawny lub ● należy stworzyć plik z ustawieniami settings.txt na nowo
Software 4	Nie udało się uruchomić procesu odpowiadającego za odczytywanie danych z sensorów	● Należy zresetować urządzenie
Software 5	Nie udało się ustawić nazwy urządzenia (mDNS hostname)	● Należy zresetować urządzenie
Software 6	Nie udało się zainicjalizować mutex'u struktury wifi_state_t przechowującej informacje o stanie połączenia wifi	● Należy zresetować urządzenie
Software 7	Nie udało się zainicjalizować modułu wifi	● Należy zresetować urządzenie
Software 8	Nie udało się zainicjalizować modułu SNTP, odpowiedzialnego za synchronizację daty	● Należy zresetować urządzenie lub ● Należy zweryfikować kod strefy czasowej (tz_code w pliku z ustawieniami)
Software 9	Nie udało się zainicjalizować serwera HTTP	● Należy zresetować urządzenie

Hardware 1	Nie udało się zainicjalizować modułu karty SD – problemy z wirtualnym systemem plików	● Należy się upewnić że karta SD znajduje się w urządzeniu lub ● Należy zresetować urządzenie
Hardware 2	Nie udało się zainicjalizować modułu karty SD – problemy z komunikacja SPI	● Należy zresetować urządzenie
Hardware 3	Wyciągnięto kartę SD w trakcie działania urządzenia	● Należy włożyć kartę SD i zresetować urządzenie

Wyżej przedstawiona lista błędów jest podstawowym wskaźnikiem że urządzenie nie działa poprawnie, ale dużo bardziej celnym rodzajem uzyskiwania informacji o błędach jest połączenie szeregowe komputera z mikrokontrolerem ESP32 przy użyciu kabla USB-A do micro-USB, przy pomocy protokołu UART o szybkości transmisji (baud rate) 115200.

6. Podsumowanie

Przedstawiony w tej pracy projekt urządzenia stacji pogodowej był projektem inżynierskim, wykonanym na kierunku Informatyka. W trakcie wykonywania tego projektu nie tylko przypomniano sobie i wykorzystano wiedzę zdobytą w trakcie studiów, ale także nauczono się wielu nowych rzeczy, przykładowo: pracy z systemami operacyjnymi czasu rzeczywistego.

Porównując końcowy efekt pracy można powiedzieć że pomyślnie zrealizowano wszystkie założone wymagania projektowe, i nauczono się jak dużo pracy wchodzi w nawet tak prosty projekt jak stworzenie prototypu stacji pogodowej korzystając z mikrokontrolera dla którego został przygotowany framework znacznie ułatwiający produkcję oprogramowania.

Aczkolwiek niektóre rozwiązania zostały zaimplementowane w gorszej wersji niż się spodziewano, na przykład: funkcjonalność interfejsu graficznego nie była tak bardzo rozbudowana jak sobie wyobrażano przed przystąpieniem do pracy, ze względu na trudności połączenia programowania mikrokontrolerów, programowania w języku C i programowania responsywnych stron internetowych, oraz fakt że moduł serwera HTTP frameworku ESP-IDF oferuje małą ilość (8 sztuk) różnych możliwych do zarejestrowania adresów URI. Innym dużym problemem w trakcie realizacji projektu było debugowanie wszelkiego rodzaju błędów działania programu. Ze względu na fakt że program działał na mikrokontrolerze a nie lokalnie na komputerze użycie konwencjonalnych narzędzi do debugowania było niemożliwe.

Projekt jest też przystosowany do możliwie dalszego rozwoju. Wszystkie moduły oprogramowania, odpowiadające za różne funkcjonalności, są stworzone według takiego samego schematu i każdy moduł odpowiada za tylko jedną logicznie spójną funkcjonalność, a więc odnalezienie się w projekcie osobie trzeciej nie powinno przysporzyć większego problemu. Co do funkcjonalności o które można by wzbogacić ten projekt to na pewno najważniejszą z nich byłoby jakieś zabezpieczenie lub uwierzytelnienie użytkownika, zanim ten zdobędzie dostęp do informacji urządzenia lub możliwości zmiany jego urządzeń.

7. Bibliografia

[1] https://www.euro.com.pl/stacje-pogody/reinston-espog02.bhtml (dostęp: 13.01.2024)

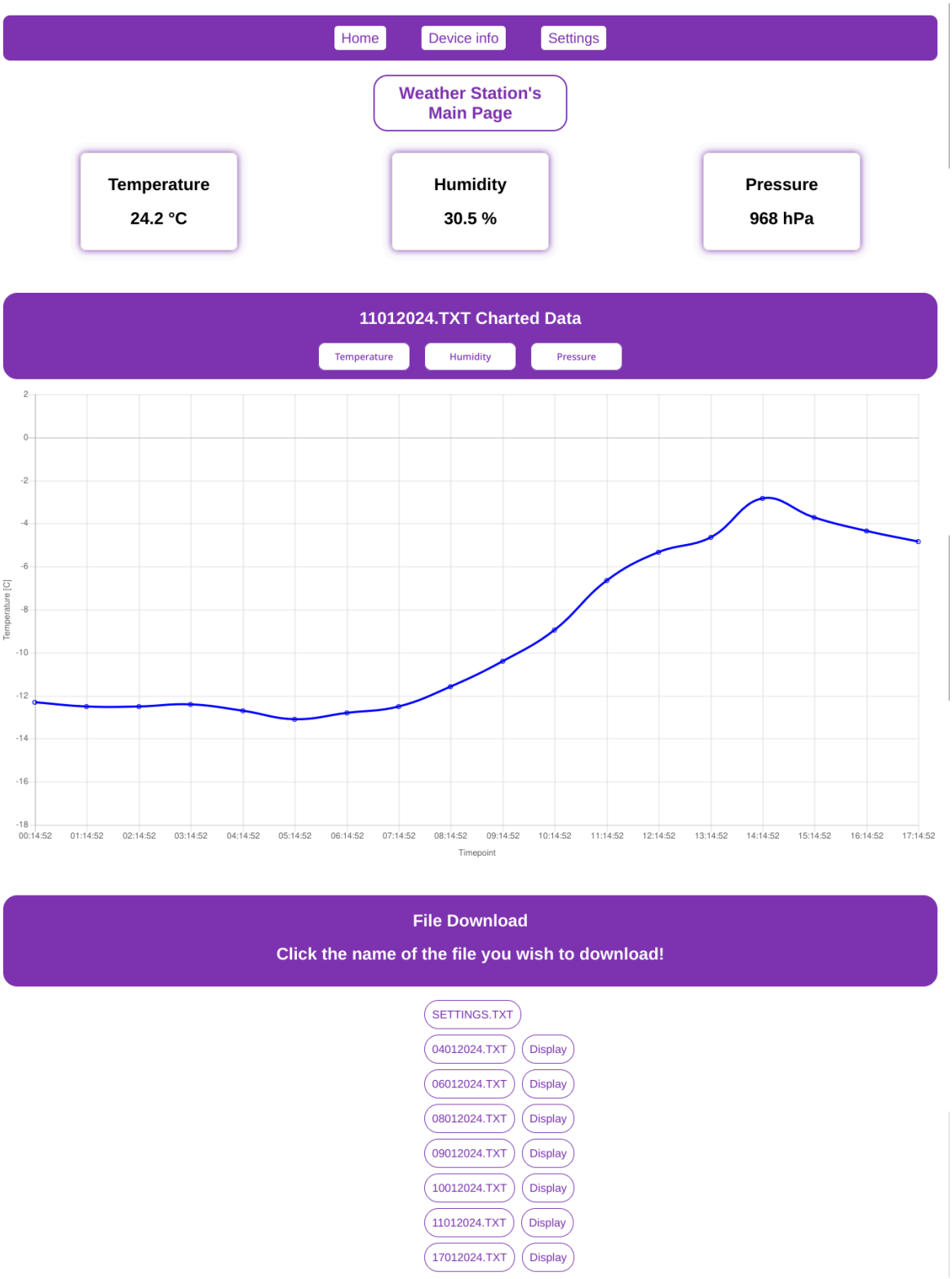
[2] https://www.sdcard.org/developers/sd-standard-overview/ (dostęp: 16.01.2024)

[3] https://web.archive.org/web/20210813122132/https://www.nxp.com/docs/en/user-guide/UM10204.pdf (dostęp: 16.01.2024)

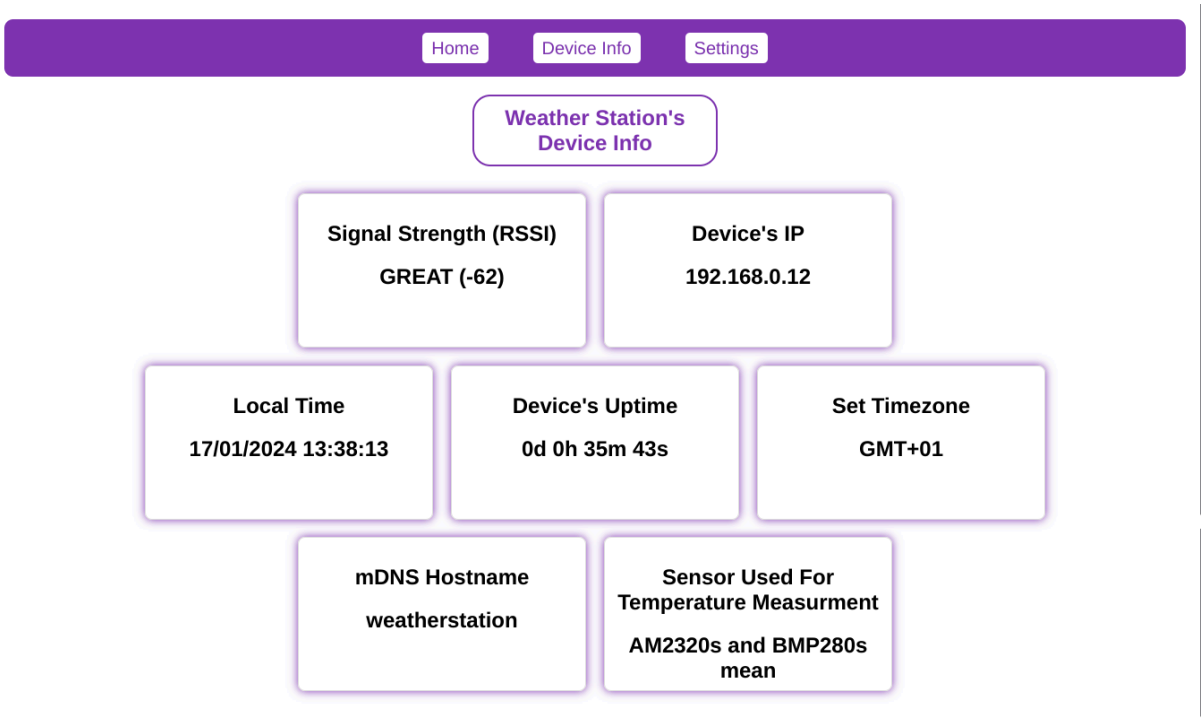
[4] https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/index.html (dostęp: 17.01.2024)

[5] https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/build-system.html#component-requirements (dostęp: 17.01.2024)

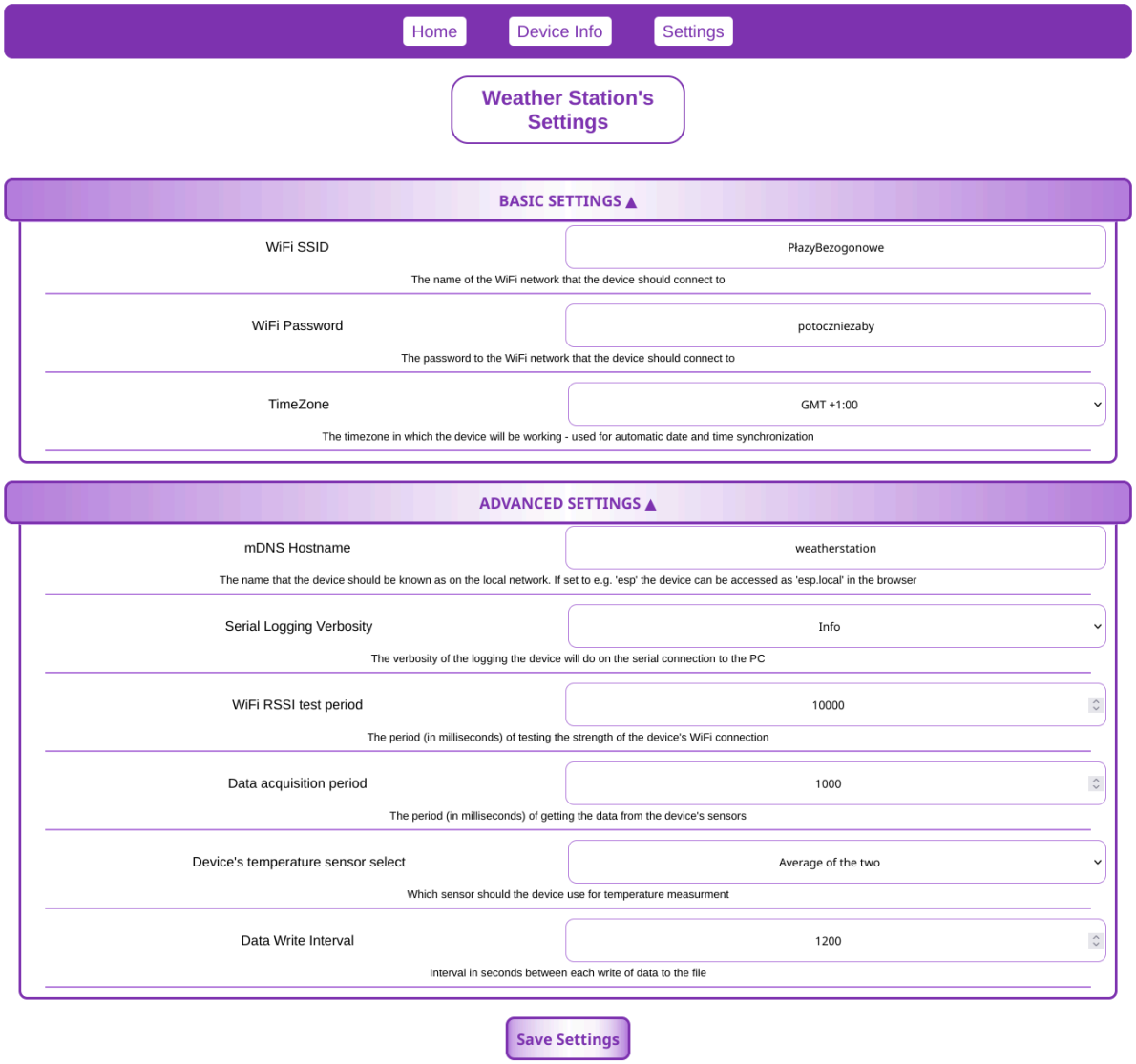
8. Źródła



Rys. 18 - Interfejs głównej strony internetowej (PC) urządzenia



Rys. 19 - Interfejs strony internetowej (PC) urządzenia z informacjami o urządzeniu



Rys. 20 - Interfejs strony internetowej (PC) urządzenia z ustawieniami

9. Spis rysunków

10. Spis tabel