

# AGH

## **AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE**

Wydział Informatyki, Elektroniki i Telekomunikacji

Systemy wbudowane 2019/2020

Moduł monitorujący stan powietrza i gleby

Dokumentacja

*Autorzy:*

Krzysztof Bieniasz

Arkadiusz Jurczak

*Opiekun:*

dr inż. Robert Brzoza-Woch

# Spis treści

<b>Spis treści</b>	<b>1</b>
<b>Wstęp</b>	<b>2</b>
<b>Informacje o ESP32-DevKitC</b>	<b>2</b>
<b>Dokumentacja poszczególnych części systemu</b>	<b>5</b>
Wykorzystanie dwóch rdzeni	5
Semafore systemu FreeRTOS	7
Mechanizm przerwań	7
Pomiar temperatury:	9
Pomiar wilgotności powietrza:	9
Pomiar wilgotności gleby:	10
System sterowania pilotem oraz ostrzegania	11
Udostępnianiu wyników pomiarów przez sieć	12
<b>Opis funkcjonalności dla użytkownika końcowego</b>	<b>16</b>
<b>Przeprowadzone prace, napotkane problemy oraz dalsze możliwości rozwoju</b>	<b>21</b>
Czynności wstępne	21
System FreeRTOS	21
Mechanizm przerwań oraz obsługa pilota	22
Czujniki wartości	22
WiFi	22
Wersja zaawansowana urządzenia	23
<b>Repozytorium</b>	<b>23</b>
<b>Bibliografia</b>	<b>24</b>

# Wstęp

Wybrany przez nas tematem projektu było stworzenie stacji pogodowej z wykorzystaniem płytki ESP32-DevKitC. Jeden z nas dysponował zestawem startowym Arduino Uno posiadającym dużą ilość interesujących sensorów, które moglibyśmy wykorzystać w naszym projekcie. W trakcie konsultacji z prowadzącymi dowiedzieliśmy się o znacznie większych możliwościach jakie oferują układy z rodziny ESP32. Dodatkowo części z zestawu startowego byłyby kompatybilne z urządzeniami danego typu. Zachęciło nas to do tego, aby nasz projekt rozwijać właśnie na tej platformie. Nasze pierwotne założenia obejmowały stworzenie urządzenia, które za pomocą czujników będzie mierzyło wartość temperatury, wilgotności powietrza i gleby oraz przesyłało zmierzone wartości z wykorzystaniem Wi-fi na inne urządzenie. Z uwagi na prostotę i popularność zdecydowaliśmy się na rozwijanie oprogramowania w środowisku Arduino IDE.

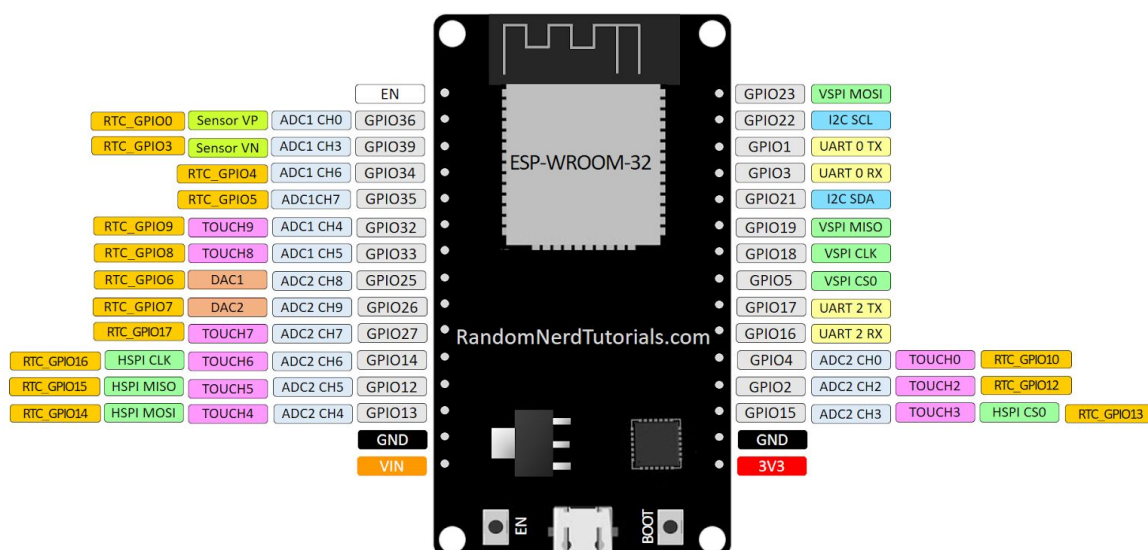
## Informacje o ESP32-DevKitC

Za podstawę projektu służy płytka ESP32-DevKitC z wbudowanym modulem ESP-WROOM-32. Jest to urządzenie o wymiarach 50 x 26 x 8 mm i napięciu zasilania 5V. Pobór prądu szacowany jest na około 80mA. Wiele pozostałych informacji możemy znaleźć w rozbudowanej oficjalnej dokumentacji [1]. Ważnymi elementami z punktu widzenia celu naszego projektu jest używany standard Wi-fi zakres temperatur, w których urządzenie poprawnie pracuje - od -40 °C do 125 °C. Istotna jest łatwość połączenia poprzez wbudowane złącze microUSB oraz dwa przyciski enable oraz boot. Zakupiona przez nas wersja składa się z 30 wejść-wyjść ogólnego przeznaczenia [2]. Warto zwrócić uwagę na to, iż na rynku oferowanych jest wiele różnych wersji zakupionego przez nas urządzenia i niektóre mają różną liczbę pinów.



# ESP32 DEVKIT V1 – DOIT

version with 30 GPIOs

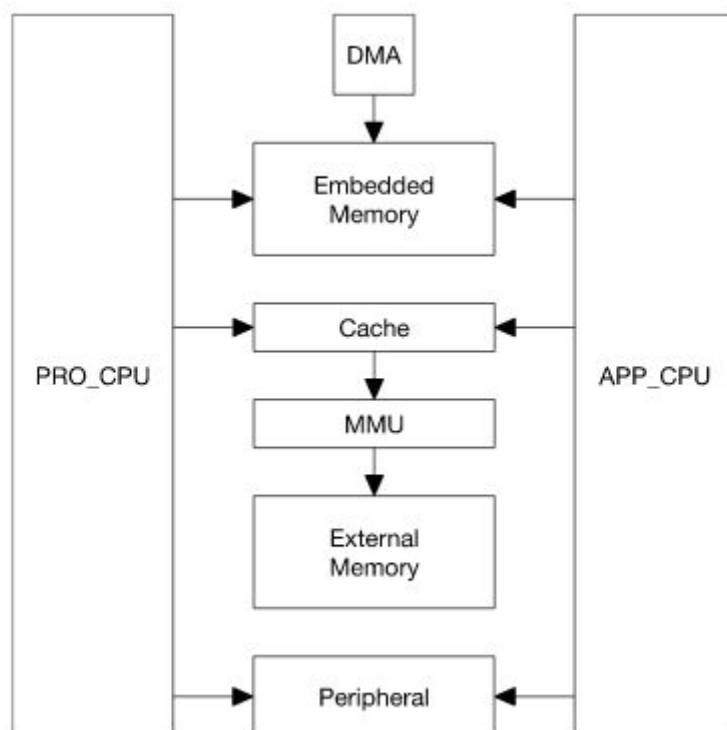


Rys. 1 Schemat wejść-wyjść ogólnego przeznaczenia płytki ESP32 DEVKIT

Z niektórymi pinami są powiązane pewne zależności, na przykład, iż znajdują się one w stanie wysokim podczas procesu bootowania. Dobrze jest to opisane w artykule [4], do którego link podajemy w bibliografii.

Sam mikrokontroler ESP32 to dwurdzeniowy układ oparty na architekturze harwardzkiej. Rdzenie procesora oznaczone są jako “PRO\_CPU” i “APP\_CPU” (“protocol”, “application”). Każdy z nich ma 4 GB (32-bit) przestrzeni adresowej. Przestrzenie adresowe są symetryczne między rdzeniami.

Magistrala danych i magistrala instrukcji są w notacji little-endian. Umożliwiają one bezpośredni dostęp do pamięci.



Rys. 2 Schemat blokowy przedstawiający strukturę systemu

Bus Type	Boundary Address		Size	Target
	Low Address	High Address		
	0x0000_0000	0x3F3F_FFFF		Reserved
Data	0x3F40_0000	0x3F7F_FFFF	4 MB	External Memory
Data	0x3F80_0000	0x3FBF_FFFF	4 MB	External Memory
	0x3FC0_0000	0x3FEF_FFFF	3 MB	Reserved
Data	0x3FF0_0000	0x3FF7_FFFF	512 KB	Peripheral
Data	0x3FF8_0000	0x3FFF_FFFF	512 KB	Embedded Memory
Instruction	0x4000_0000	0x400C_1FFF	776 KB	Embedded Memory
Instruction	0x400C_2000	0x40BF_FFFF	11512 KB	External Memory
	0x40C0_0000	0x4FFF_FFFF	244 MB	Reserved
Data / Instruction	0x5000_0000	0x5000_1FFF	8 KB	Embedded Memory
	0x5000_2000	0xFFFF_FFFF		Reserved

Rys. 3: Odwzorowanie adresów

Pamięć wbudowana składa się z 4 części:

- ROM (448 KB)
- SRAM (520 KB)
- RTC FAST (8 KB)
- RTC SLOW (8 KB)

ROM i SRAM są taktowane częstotliwością CPU\_CLK, co oznacza że procesor uzyskuje dostęp do nich dostęp w jednym cyklu. RTC FAST i RTC SLOW mają niższe częstotliwości, w przypadku FAST jest ona taka sama jak dla DMA (Direct Memory Access).

Domyślnie wbudowanym systemem operacyjnym w układzie ESP32 jest system FreeRTOS. W oficjalnej dokumentacji jest sekcja System API, w której możemy się doszukać wiele istotnych jak również ciekawych informacji.

## Dokumentacja poszczególnych części systemu

Ze względu na niewielkie rozmiary projektu poszczególne części systemu są ze sobą mocno powiązane, ale w tym rozdziale postaramy się przedstawić odrębnie wykorzystane możliwości sprzętowe, z których korzystaliśmy lub stworzone funkcjonalności. W celu zwiększenia czytelności niektóre sekcje przykładowego wykorzystanie w kodzie są ograniczone objętościowo i część fragmentów kodu jest zastąpiona słownym komentarzem.

### Wykorzystanie dwóch rdzeni

Tak jak wspomnieliśmy na wstępie wykorzystujemy środowisko programistyczne ArduinoIDE. W tym środowisku wymagana jest deklaracja funkcji setup() oraz loop(). Funkcja setup() to funkcja, która wywoływana jest automatycznie po włączeniu zasilania lub naciśnięciu przycisku RESET. Domyślnie podczas uruchamiania kodu w ArduinoIDE wykorzystuje się rdzeń numer 1 [5]. Taka sytuacja może doprowadzić do sytuacji, kiedy rdzeń 1 będzie przeciążony, a rdzeń numer 0 będzie miał dużą ilość wolnych zasobów. Obszarem, który dosyć szybko nas zainteresował była możliwość uruchamiania zadań systemu operacyjnego FreeRTOS [6] na wybranym rdzeniu. W celu strukturyzacji naszego programu i łatwego zarządzania zadaniami inicjalizujemy je właśnie we wspomnianej wcześniej funkcji setup().

Sygnatura funkcji xTaskCreatePinnedToCore umożliwiającą tworzenia zadań systemu FreeRTOS:

```
BaseType_t xTaskCreatePinnedToCore
(TaskFunction_t pvTaskCode,
const char *constpcName,
const uint32_t usStackDepth,
void *constpvParameters,
UBaseType_t uxPriority,
TaskHandle_t *constpvCreatedTask,
const BaseType_t xCoreID)
```

Przykład użycia powyższej funkcji w naszym programie:

```
xTaskCreatePinnedToCore (  
    TaskDHTCode,      /* Task function. */  
    "TaskDHT",        /* name of task. */  
    10000,            /* Stack size of task */  
    NULL,             /* parameter of the task */  
    1,                /* priority of the task */  
    &TaskDHTHandler,  /* Task handle to keep track of created task */  
    1);               /* pin task to core 0 */
```

Powyższej funkcji używaliśmy za każdym razem, aby utworzyć zadania systemu operacyjnego odpowiadające za odczyt danych z danego czujnika. Poprzez to, iż każdy czujnik miał odpowiadające sobie zadanie systemu operacyjnego można było zdefiniować osobne czasy opóźnień w pętlach, w których wykonywały się zadania.

Było to istotne z tego punktu widzenia, iż różne czujniki zbierały dane w różnych odstępach czasu. Ważnym faktem była też możliwość wyboru rdzenia, na którym uruchomimy dane zadanie. Poprzez wywołanie funkcji `xPortGetCoreID()` możemy sprawdzić numer rdzenia, na którym wywoływana jest zadanie systemu.

Przykład budowy funkcji uruchamianej jako zadanie systemu operacyjnego na przykładzie funkcji odpowiedzialnej za odczytywanie danych z czujnika temperatury:

```
void TaskDallasCode (void *pvParameters)  
{  
    sensors.begin ();  
    // w ogólności rozpoczęcie pracy czujników  
    for (;;) // wykonująca się w nieskończoność pętla  
    {  
        // odczyt danych z czujnika  
        sensors.requestTemperatures ();  
        // przetwarzanie danych  
        delay (1400); // uśpienie danego zadania  
    }  
}
```

## Semafor systemu FreeRTOS

Innym elementem zawierającym mechanizmy systemu operacyjnego było wykorzystanie semaforów. Aby skorzystać z API do semaforów należy dołączyć plik nagłówkowy "freertos/semphr.h".

Funkcje, które używaliśmy w naszym programie:

```
SemaphoreHandle_t xSemaphoreCreateBinary ()  
xSemaphoreTake (xSemaphore, xBlockTime)  
xSemaphoreGive (xSemaphore)
```

Drugim argumentem funkcji xSemaphoreTake jest czas oczekiwania na otrzymanie semafora. W przypadku zajęcia semafora funkcja zwraca mako pdTRUE, w przeciwnym wypadku pdFALSE. W naszym programie konieczność użycia semaforów wynikała z faktu, że w jednych zadaniach systemu operacyjnego zapisywaliśmy dane z czujników do zmiennych, a w innych zadaniach odczytywaliśmy te właśnie zmienne.

Przykład użycia semaforów w kodzie:

```
if (xSemaphoreTake (xBinarySemaphoreHumidity, ( TickType_t ) 1000 ) == pdTRUE)  
{  
    humidityGlobal = h;  
    xSemaphoreGive (xBinarySemaphoreHumidity);  
}
```

## Mechanizm przerwań

ESP32 oferuje do 32 obsług przerwań dla każdego rdzenia. Każde przerwania ma określony poziom oraz może być sklasyfikowane do dwóch grup - Hardware oraz Software [7].

Funkcja, za pomocą której dołączamy wystąpienie danej zmiany napięcia na jednym z pinów do funkcji obsługującej przerwanie:

```
attachInterrupt (GPIOpin, ISR, Mode);
```

GPIOpin - ustawia dany pin, jako 'interrupt pin' co oznacza, że ESP32 'monitoruje' dany pin pod kątem przerwań



ISR - funkcja wywoływana za każdym razem, gdy dochodzi do przerwania.

Mode - definiuje typ kiedy przerwania powinno być wyzwolone. Danych jest pięć makr odpowiadających odpowiednim zmianom i/lub stanom napięcia na pinie: LOW, HIGH, CHANGE, FALLING, RISING.

Przykład z wywołania funkcji attachInterrupt oraz wywoływanej funkcji w razie wystąpienia warunków do przerwania (obsługa działania pilota):

```
attachInterrupt (digitalPinToInterrupt (RECV_PIN), detectsMovement, RISING);

void IRAM_ATTR detectsMovement ()
{
  if (irrecv.decode (&results))
  {
    Serial.println (results.value);
    irrecv.resume (); // Receive the next value
    remoteControlTab [remoteControlTabIndex] = results.value;
    remoteControlTabIndex++;
    if(results.value == CHANEL_PLUS) remoteControlParser ();
  }
}
```

Ponadto zdecydowaliśmy się wykorzystać fakt, iż niektóre piny są oznaczone jako TOUCHPIN, a więc reagują na ludzki dotyk. Istnieje też gotowa funkcja, która wiąże dotyk jednego z pinów do funkcji obsługującej przerwanie.

```
touchAttachInterrupt (PinNumber, callback, Threshold);
```

callback - funkcja wywoływana w przypadku wystąpienia przerwania

threshold - wartość, która musi być przekroczona, aby nastąpiło przerwanie, innymi słowy zapobiega, temu żeby lekkie muśnięcie wywołało przerwanie, zamiast intencyjnego dotyku.

Przykład z kodu (obsługa resetu dopuszczalnych wartości mierzonych parametrów):

```
touchAttachInterrupt (T4, gotTouch, 20);

void gotTouch ()
{
  Serial.println ("Touched\n");
  createCheck = 1;
  // zmienna powodująca wyłączenie diody/buzzera
  // w zadaniu odpowiedzialnym za monitorowanie
  // czy są przekroczone limity
}
```

## Pomiar temperatury:

Wykorzystujemy czujnik Dallas DS18S20. Jest on często wykorzystywany w podobnym projektach [8] Zawiera on trzy wyprowadzenia: napięcia zasilania, sygnał cyfrowy oraz masa układu. Z punktu widzenia projektu istotne elementy specyfikacji to zakres pomiarowy wynoszący od -55 °C do 125 °C oraz dokładność +/- 0,5 °C w zakresie -10 °C do 85 °C.



Wykorzystanie w kodzie projektu:

```
#include <DallasTemperature.h>.
const int oneWireBus = 5;

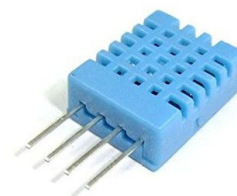
/* zadanie systemu FreeRTOS */

void TaskDallasCode (void *pvParameters)
{
    OneWire oneWire (oneWireBus);
    DallasTemperature sensors (&oneWire);
    sensors.begin ();

    for (;;)
    {
        sensors.requestTemperatures ();
        float temperatureC = sensors.getTempCByIndex (0);
        float temperatureF = sensors.getTempFByIndex (0);
        // przetwarzanie danych
        delay (1400);
    }
}
```

## Pomiar wilgotności powietrza:

Wykorzystujemy czujnik DHT11. Zawiera on cztery wyprowadzenia: napięcia zasilania, sygnał cyfrowy, niewykorzystywany pin oraz masę układu. Z punktu widzenia projektu istotne elementy specyfikacji to zakres pomiarowy wilgotności z dokładnością 5%. Ponadto jest możliwy jest pomiar temperatury powietrza z dokładnością 2°C (niewykorzystane przez nas w projekcie).



Wykorzystanie w kodzie projektu:

```
#include <DHT.h> // biblioteka od AdaFruit

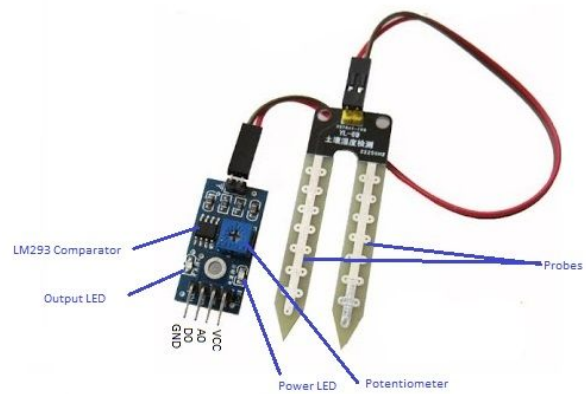
const int DHTPIN = 4;
const int DHTTYPE = DHT11;
void TaskDHTCode (void *pvParameters)
{
    DHT dht (DHTPIN, DHTTYPE);
    dht.begin ();
    for (;;)
    {
        float h = dht.readHumidity ();
        float t = dht.readTemperature ();
        float f = dht.readTemperature (true);
        /* przetwarzanie informacji */

        delay (2000);
    }
}
```

## Pomiar wilgotności gleby:

Czujnik wilgotności zakupiliśmy w sklepie internetowym [10]. Składa się on z sondy pomiarowej oraz modułu głównego. Ciężko określić jego producenta oraz znaleźć oficjalną dokumentację. Moduł główny zawiera 4 wyprowadzania: źródło napięcia, wyjście analogowe, wyjście cyfrowe oraz masa układu. Domyślnie wyjście cyfrowe jest w stanie wysokim, natomiast po wykryciu wilgotności przechodzi w stan niski. Czulość możemy regulować za pomocą wbudowanego potencjometru.

Wyjście A0 działa analogowo i mniejsza wartość napięcia wraz ze wzrostem wilgotności (jest do niego odwrotnie proporcjonalne). W naszym projekcie korzystamy tylko z danych z wyjścia analogowego, ponieważ wykorzystując dane z wyjście cyfrowego uzyskujemy niewielką ilość informacji. Przykład korzystanie w kodzie jest podobny do wcześniej przedstawionych czujników.



## System sterowania pilotem oraz ostrzegania

Dysponowaliśmy czujnikiem podczerwieni IR receiver CHQ1838 oraz pilotem IR NEC 38kHz. Częstotliwość pracy odbiornika i pilota jest taka sama i wynosi 38 kHz. Czujnik zawiera 3 piny: wyjście, napięcia zasilania oraz masa układu. Na stronach oferujących czujnik można przeczytać, iż jest on wytrzymały na zakłócenia pole elektrycznego, jednak nie współpracuje on bezbłędnie z posiadanym przez nas pilotem. Pilot wykorzystuje protokół NEC. W naszym programie po odbiorze przez odbiornik sygnału z pilota następuje przerwanie programowe i wywoływana jest funkcja analizująca otrzymany sygnał. Wartości otrzymane na skutek naciśnięcia poszczególnych klawiszy wyznaczyliśmy doświadczalnie.



Wykorzystanie w kodzie:

```
const int CHANEL_MINUS = 16753245;
/* kolejne wartości */
const int IR_FAILED = 4294967295;

int RECV_PIN = 19;
IRrecv irrecv (RECV_PIN);
decode_results results;

long decodeSingleNumber (long value)
/* funkcja zwracająca liczbę od 1-9 na podstawie wartości sygnału */

void remoteControlParser ()
/* analizowanie wprowadzonej przez użytkownika sekwencji i w przypadku
poprawnej sekwencji ustawienie nowych limitów wartości*/
void IRAM_ATTR detectsMovement ()
{
  if (irrecv.decode (&results))
  {
    // przetwarzanie wyniku
  }
}
```

```

}

// w funkcji setup
irrecv.enableIRIn ();
attachInterrupt (digitalPinToInterrupt (RECV_PIN), detectsMovement, RISING);

```

System ostrzegania działa w ten sposób, iż jedno zadanie systemu operacyjnego monitoruje aktualne wartości z obowiązującymi limitami. Przekroczenie tych wartości uruchamia buzzer lub włącza diodę. Buzzer wydaje on cichy, ale denerwujący dźwięk, co powinno skłonić użytkownika do zainteresowania problemem.

```

void TaskBuzzerCode(void * pvParameters)
{
    digitalWrite(ledPin, LOW);
    for (;;)
    {
        while (true)
        {
            if (resetLimits == 1)
                // zmienna możliwa do zmiany poprzez przerwanie dotykowe
            {
                break;
            }
            // analiza czy wartości są przekroczone i ewentualne włączenie buzzera
            delay (500);
        }
        digitalWrite (ledPin, LOW);
        resetLimits = 0;
        // powrót limitów do stanu początkowego
        Serial.println ("Limits were restarted");
    }
}

```

## Udostępnianiu wyników pomiarów przez sieć

Udostępnianie danych jest oparte na modelu klient-serwer (wymaga dołączenia plików nagłówkowych `WebServer.h` i `WiFi.h`). Nasze urządzenie łączy się z uprzednio skonfigurowaną siecią za pomocą jej nazwy i hasła.

```

const char * ssid = "NETWORK";
const char * password = "12345678";

WiFi.begin (ssid, password);
while (WiFi.waitForConnectResult () != WL_CONNECTED);

```

Następnie jest tworzony serwer przyjmujący zapytania na porcie 80 (który jest przypisany do obsługi HTTP).

```
WebServer server (80);  
server.handleClient ();
```

Zależnie od treści zapytania serwer wykonuje określoną funkcję.

```
server.on ("/", handleRoot);  
server.on ("/TEMPread", handleTEMP);  
server.on ("/HUMIread", handleHUMI);
```

```
void handleRoot ()  
{  
    server.send (200, "text/html", content);  
}  
void handleTEMP ()  
{  
    server.send (200, "text/plain", String (temperatureGlobal));  
}  
  
void handleHUMI ()  
{  
    server.send (200, "text/plain", String (humidityGlobal));  
}
```

Zmienna `content` to kod strony internetowej, którą otrzyma host po połączeniu z serwerem. Struktura jest opisana w HTML, styl w CSS, a dynamiczna zawartość w JavaScript.

```
// const char content []  
  
<!DOCTYPE html>  
<html>  
<head>  
    <meta charset = "utf-8" name="viewport"  
        content="width=device-width, initial-scale=1.0, user-scalable=no">  
    <title>ESP32 Weather Station</title>  
    <style>  
        <!-- Styl -->  
    </style>  
    <script type="application/javascript">  
        var temperatures = [0, 0, 0, 0, 0, 0];  
        setInterval (() => getSensorData (), 1000);  
        function getSensorData ()  
        {
```

```

var xhttp = new XMLHttpRequest ();
xhttp.onreadystatechange = function ()
{
    if (this.readyState == 4 && this.status == 200)
    {
        for (i = 5; i > 0; i--) temperatures [i] = temperatures [i - 1];
        temperatures [0] = parseInt (this.responseText);
        document.getElementById ("TEMPvalue").innerHTML = this.responseText;
        clear ();
        drawAxis ();
        draw ();
    }
}
xhttp.open ("GET", "TEMPread", true);
xhttp.send ();
//
var xhttp = new XMLHttpRequest ();
xhttp.onreadystatechange = function ()
{
    if (this.readyState == 4 && this.status == 200)
        document.getElementById ("HUMIvalue").innerHTML = this.responseText;
}
xhttp.open ("GET", "HUMIread", true);
xhttp.send ();
}
function clear ()
{
    // Wyczyść canvas
}
function drawAxis ()
{
    // Narysuj osie układu współrzędnych
}
function draw ()
{
    var canvas = document.getElementById ("plot");
    if (canvas.getContext)
    {
        var ctx = canvas.getContext ("2d");
        ctx.fillStyle = "#000000";
        for (j = 0; j < 6; j++) if (temperatures [j] !== 0)
        {
            ctx.beginPath ();
            ctx.arc (80 * j + (j === 0 ? 5 : (j === 5 ? -5 : 0)),
                250 - 100 * (temperatures [j] - 20) + (j === 0 ? 5 : 0),
                5, 0, Math.PI * 2, true);
            ctx.fill ();
        }
    }
}
}

```

```

</script>
</head>
<body>
  <h1>ESP32 Weather Station</h1>
  <p class = "text">Temperature: <span id = "TEMPvalue">0</span>&degC</p>
  <p class = "text">Humidity: <span id = "HUMIvalue">0</span>%</p>
  <br>
  <canvas id="plot" width="400" height="500"></canvas>
</body>
</html>

```

Potwierdzeniem poprawności połączenia jest wygenerowanie i wyświetlenie adresu IPv4. Można go wykorzystać, aby na bieżąco monitorować temperaturę oraz wilgotność.

```

Serial.print ("\nWiFi connected! Got IP: ");
Serial.println (WiFi.localIP ());

```

Wyniki są udostępniane za pomocą AJAX (Asynchronous JavaScript + XML). Na zapytanie typu GET opakowane w obiekt `XMLHttpRequest` serwer odsyła napis, który po konwersji na liczbę, reprezentuje jeden z obserwowanych parametrów pogodowych.

```

var xhttp = new XMLHttpRequest ();
xhttp.onreadystatechange = function ()
{
  if (this.readyState == 4 && this.status == 200)
  {
    for (i = 5; i > 0; i--) temperatures [i] = temperatures [i - 1];
    temperatures [0] = parseInt (this.responseText);
    document.getElementById ("TEMPvalue").innerHTML = this.responseText;
  }
}
xhttp.open ("GET", "TEMPread", true);
xhttp.send ();

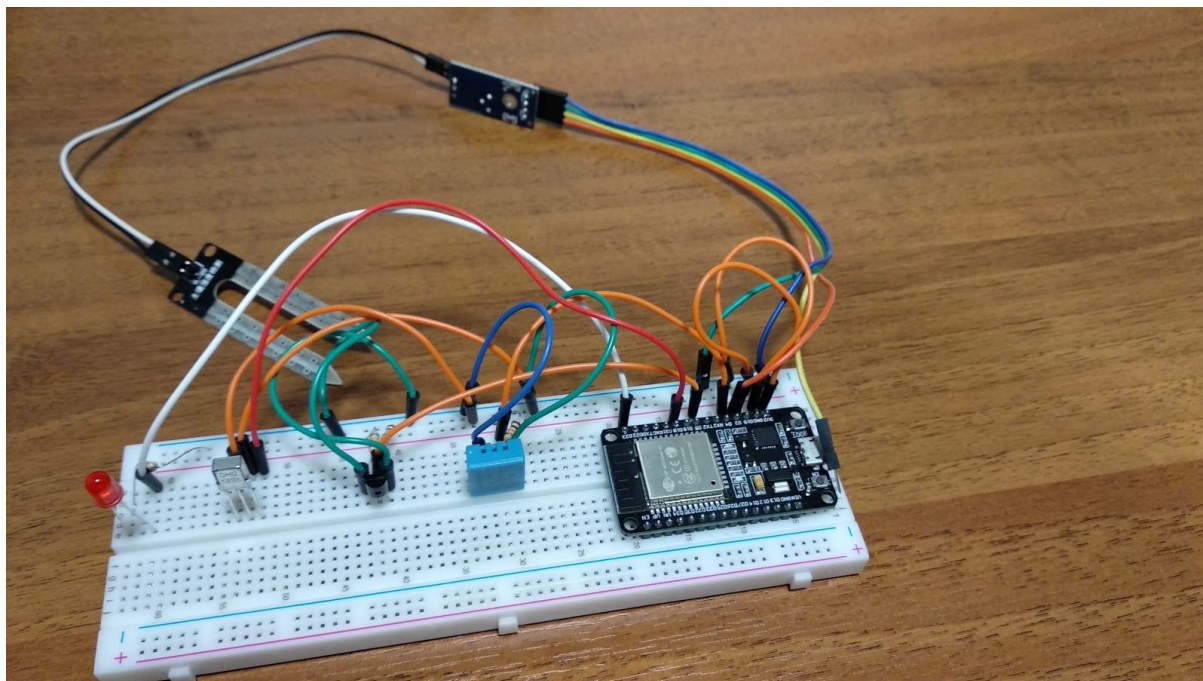
```

Ustawianie nowych wartości odbywa się w następującej kolejności:

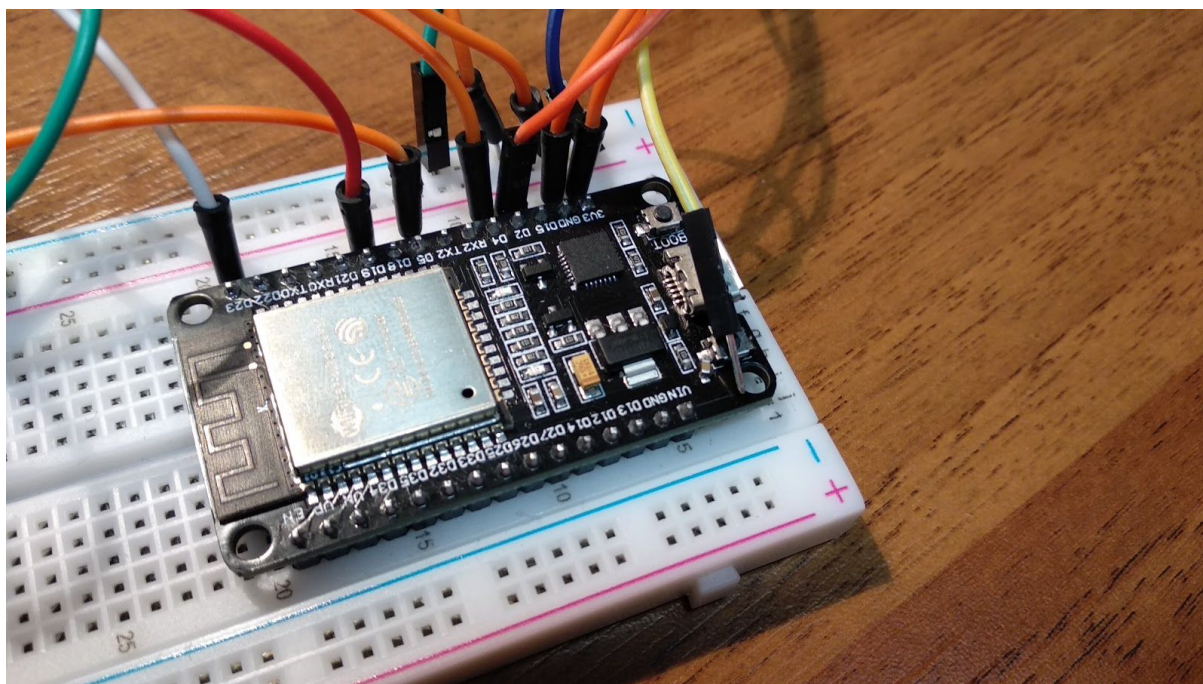
- utworzenie zapytania
- dodanie callbacka do obiektu (co się stanie po otrzymaniu odpowiedzi z serwera)
  - upewnienie się czy potencjalna odpowiedź jest poprawna
  - aktualizacja tablicy ostatnich wyników
  - ustawienie aktualnej wartości za pomocą DOM (Document Object Model)
- ustawienie pobierania odpowiednich danych
- wysłanie zapytania



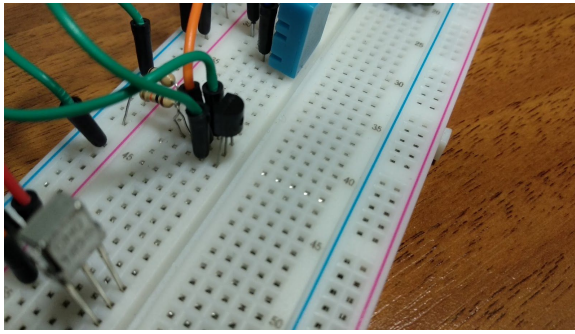
## Opis funkcjonalności dla użytkownika końcowego



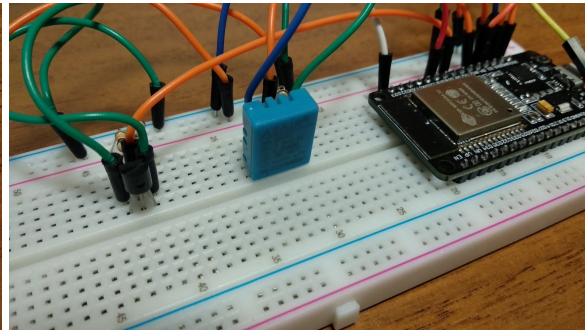
*Zdjęcie 1: Moduł monitorujący stan powietrza i gleby*



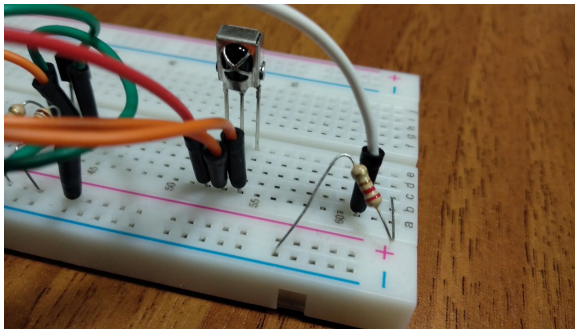
*Zdjęcie 2: Najważniejszy element modułu - ESP32-DevKit*



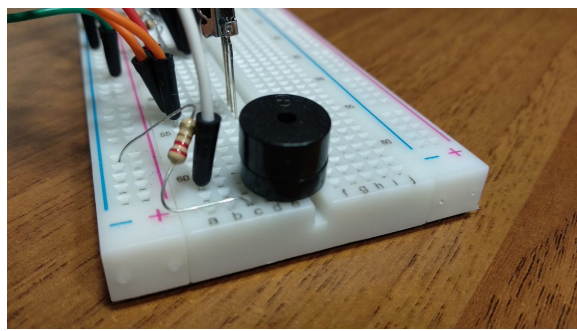
*Zdjęcie 3: Czujnik Dallas*



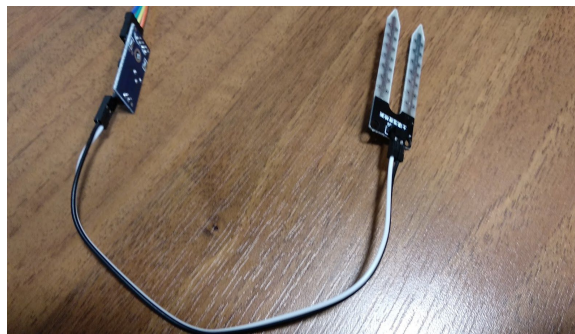
*Zdjęcie 4: Czujnik DHT11*



*Zdjęcie 5: Czujnik podczerwieni*



*Zdjęcie 6: Buzzer*



*Zdjęcie 7: Czujnik wilgotności gleby*

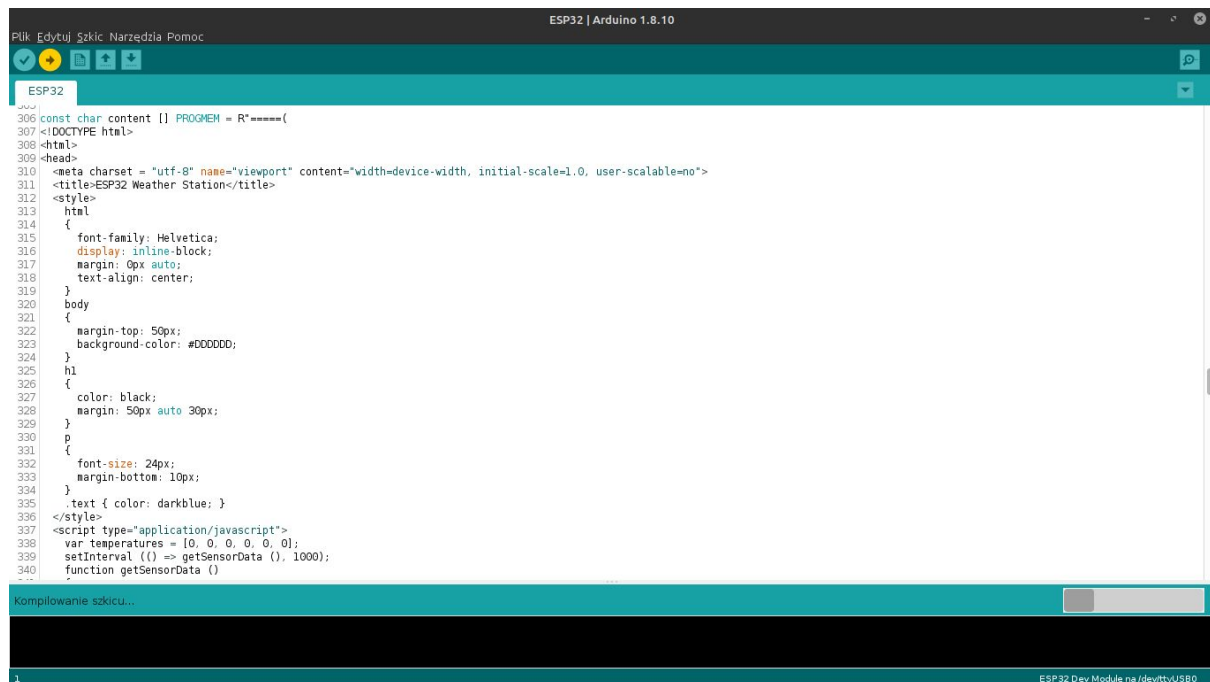
Pierwszym krokiem, który posiadacz urządzenia musi wykonać w celu wykonania pomiaru temperatury jest pobranie lub sklonowanie projektu z repozytorium w serwisie <https://github.com/kbieniasz/WheatherStation>. Zalecamy stosowanie edytora Arduino IDE. Biblioteki można zainstalować ręcznie lub skopiować biblioteki z folderu libraries z pobranego projektu do folderu libraries w folderze Arduino na swoim komputerze.



Po podłączeniu sprzętu z użyciem kabla z przejściówką microUSB, można wgrać projekt pamiętając, by po pojawieniu się Komunikatu 1 nacisnąć przycisk BOOT znajdujący się obok wejścia microUSB.

Komunikat 2 świadczy o poprawności procesów kompilacji i wgrywania.

ESP32 wysyła już wiadomości, które są do odczytania przez monitor portu szeregowego (Screenshot 2). Do poprawnego działania niezbędne jest naciśnięcie drugiego z przycisków, EN (skrót od enable). Jeśli istnieje sieć o nazwie i hasle podanych w kodzie programu, zostanie zwrócony adres IP.



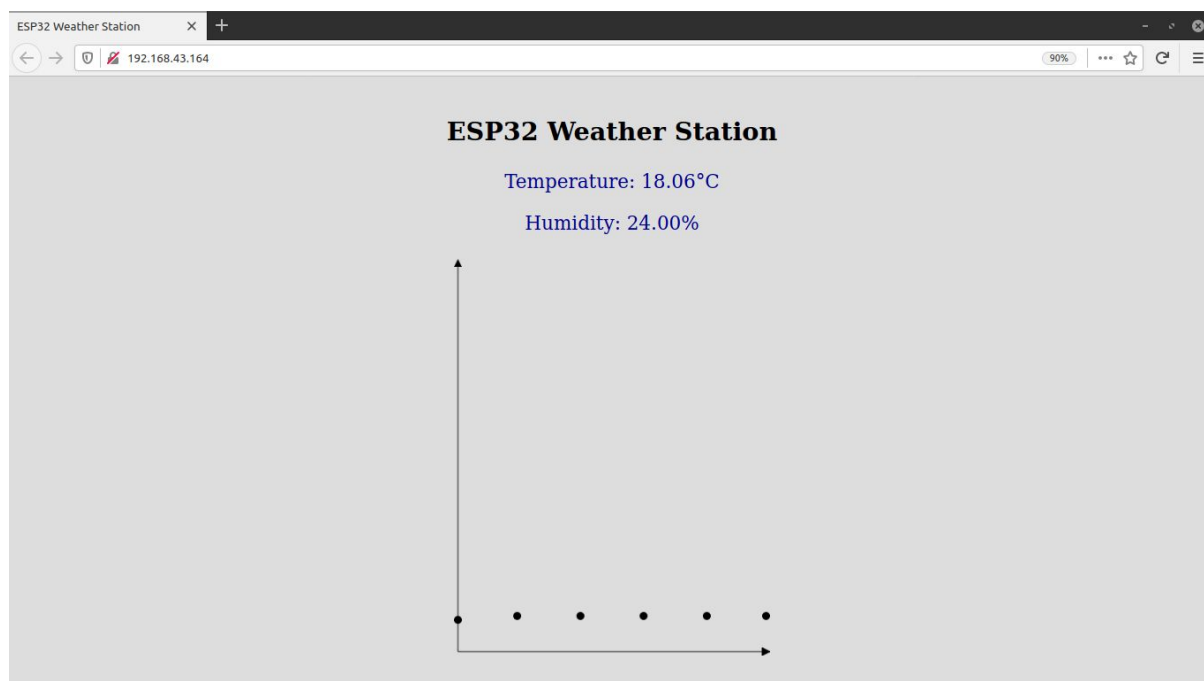
```
ESP32
306 const char content [] PROGMEM = R"====(
307 <!DOCTYPE html>
308 <html>
309 <head>
310 <meta charset = "utf-8" name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no">
311 <title>ESP32 Weather Station</title>
312 <style>
313   html
314   {
315     font-family: Helvetica;
316     display: inline-block;
317     margin: 0px auto;
318     text-align: center;
319   }
320   body
321   {
322     margin-top: 50px;
323     background-color: #000000;
324   }
325   h1
326   {
327     color: black;
328     margin: 50px auto 10px;
329   }
330   p
331   {
332     font-size: 24px;
333     margin-bottom: 10px;
334   }
335   .text { color: darkblue; }
336 </style>
337 <script type="application/javascript">
338   var temperatures = [0, 0, 0, 0, 0, 0];
339   setInterval (() => getSensorData (), 1000);
340   function getSensorData ()
341   {
342     // ...
343   }
344 </script>
345 </html>
346 )";
347
348 // ...
349
350 // ...
351
352 // ...
353
354 // ...
355
356 // ...
357
358 // ...
359
360 // ...
361
362 // ...
363
364 // ...
365
366 // ...
367
368 // ...
369
370 // ...
371
372 // ...
373
374 // ...
375
376 // ...
377
378 // ...
379
380 // ...
381
382 // ...
383
384 // ...
385
386 // ...
387
388 // ...
389
390 // ...
391
392 // ...
393
394 // ...
395
396 // ...
397
398 // ...
399
400 // ...
401
402 // ...
403
404 // ...
405
406 // ...
407
408 // ...
409
410 // ...
411
412 // ...
413
414 // ...
415
416 // ...
417
418 // ...
419
420 // ...
421
422 // ...
423
424 // ...
425
426 // ...
427
428 // ...
429
430 // ...
431
432 // ...
433
434 // ...
435
436 // ...
437
438 // ...
439
440 // ...
441
442 // ...
443
444 // ...
445
446 // ...
447
448 // ...
449
450 // ...
451
452 // ...
453
454 // ...
455
456 // ...
457
458 // ...
459
460 // ...
461
462 // ...
463
464 // ...
465
466 // ...
467
468 // ...
469
470 // ...
471
472 // ...
473
474 // ...
475
476 // ...
477
478 // ...
479
480 // ...
481
482 // ...
483
484 // ...
485
486 // ...
487
488 // ...
489
490 // ...
491
492 // ...
493
494 // ...
495
496 // ...
497
498 // ...
499
500 // ...
501
502 // ...
503
504 // ...
505
506 // ...
507
508 // ...
509
510 // ...
511
512 // ...
513
514 // ...
515
516 // ...
517
518 // ...
519
520 // ...
521
522 // ...
523
524 // ...
525
526 // ...
527
528 // ...
529
530 // ...
531
532 // ...
533
534 // ...
535
536 // ...
537
538 // ...
539
540 // ...
541
542 // ...
543
544 // ...
545
546 // ...
547
548 // ...
549
550 // ...
551
552 // ...
553
554 // ...
555
556 // ...
557
558 // ...
559
560 // ...
561
562 // ...
563
564 // ...
565
566 // ...
567
568 // ...
569
570 // ...
571
572 // ...
573
574 // ...
575
576 // ...
577
578 // ...
579
580 // ...
581
582 // ...
583
584 // ...
585
586 // ...
587
588 // ...
589
590 // ...
591
592 // ...
593
594 // ...
595
596 // ...
597
598 // ...
599
600 // ...
601
602 // ...
603
604 // ...
605
606 // ...
607
608 // ...
609
610 // ...
611
612 // ...
613
614 // ...
615
616 // ...
617
618 // ...
619
620 // ...
621
622 // ...
623
624 // ...
625
626 // ...
627
628 // ...
629
630 // ...
631
632 // ...
633
634 // ...
635
636 // ...
637
638 // ...
639
640 // ...
641
642 // ...
643
644 // ...
645
646 // ...
647
648 // ...
649
650 // ...
651
652 // ...
653
654 // ...
655
656 // ...
657
658 // ...
659
660 // ...
661
662 // ...
663
664 // ...
665
666 // ...
667
668 // ...
669
670 // ...
671
672 // ...
673
674 // ...
675
676 // ...
677
678 // ...
679
680 // ...
681
682 // ...
683
684 // ...
685
686 // ...
687
688 // ...
689
690 // ...
691
692 // ...
693
694 // ...
695
696 // ...
697
698 // ...
699
700 // ...
701
702 // ...
703
704 // ...
705
706 // ...
707
708 // ...
709
710 // ...
711
712 // ...
713
714 // ...
715
716 // ...
717
718 // ...
719
720 // ...
721
722 // ...
723
724 // ...
725
726 // ...
727
728 // ...
729
730 // ...
731
732 // ...
733
734 // ...
735
736 // ...
737
738 // ...
739
740 // ...
741
742 // ...
743
744 // ...
745
746 // ...
747
748 // ...
749
750 // ...
751
752 // ...
753
754 // ...
755
756 // ...
757
758 // ...
759
760 // ...
761
762 // ...
763
764 // ...
765
766 // ...
767
768 // ...
769
770 // ...
771
772 // ...
773
774 // ...
775
776 // ...
777
778 // ...
779
780 // ...
781
782 // ...
783
784 // ...
785
786 // ...
787
788 // ...
789
790 // ...
791
792 // ...
793
794 // ...
795
796 // ...
797
798 // ...
799
800 // ...
801
802 // ...
803
804 // ...
805
806 // ...
807
808 // ...
809
810 // ...
811
812 // ...
813
814 // ...
815
816 // ...
817
818 // ...
819
820 // ...
821
822 // ...
823
824 // ...
825
826 // ...
827
828 // ...
829
830 // ...
831
832 // ...
833
834 // ...
835
836 // ...
837
838 // ...
839
840 // ...
841
842 // ...
843
844 // ...
845
846 // ...
847
848 // ...
849
850 // ...
851
852 // ...
853
854 // ...
855
856 // ...
857
858 // ...
859
860 // ...
861
862 // ...
863
864 // ...
865
866 // ...
867
868 // ...
869
870 // ...
871
872 // ...
873
874 // ...
875
876 // ...
877
878 // ...
879
880 // ...
881
882 // ...
883
884 // ...
885
886 // ...
887
888 // ...
889
890 // ...
891
892 // ...
893
894 // ...
895
896 // ...
897
898 // ...
899
900 // ...
901
902 // ...
903
904 // ...
905
906 // ...
907
908 // ...
909
910 // ...
911
912 // ...
913
914 // ...
915
916 // ...
917
918 // ...
919
920 // ...
921
922 // ...
923
924 // ...
925
926 // ...
927
928 // ...
929
930 // ...
931
932 // ...
933
934 // ...
935
936 // ...
937
938 // ...
939
940 // ...
941
942 // ...
943
944 // ...
945
946 // ...
947
948 // ...
949
950 // ...
951
952 // ...
953
954 // ...
955
956 // ...
957
958 // ...
959
960 // ...
961
962 // ...
963
964 // ...
965
966 // ...
967
968 // ...
969
970 // ...
971
972 // ...
973
974 // ...
975
976 // ...
977
978 // ...
979
980 // ...
981
982 // ...
983
984 // ...
985
986 // ...
987
988 // ...
989
990 // ...
991
992 // ...
993
994 // ...
995
996 // ...
997
998 // ...
999
1000 // ...
1001
1002 // ...
1003
1004 // ...
1005
1006 // ...
1007
1008 // ...
1009
1010 // ...
1011
1012 // ...
1013
1014 // ...
1015
1016 // ...
1017
1018 // ...
1019
1020 // ...
1021
1022 // ...
1023
1024 // ...
1025
1026 // ...
1027
1028 // ...
1029
1030 // ...
1031
1032 // ...
1033
1034 // ...
1035
1036 // ...
1037
1038 // ...
1039
1040 // ...
1041
1042 // ...
1043
1044 // ...
1045
1046 // ...
1047
1048 // ...
1049
1050 // ...
1051
1052 // ...
1053
1054 // ...
1055
1056 // ...
1057
1058 // ...
1059
1060 // ...
1061
1062 // ...
1063
1064 // ...
1065
1066 // ...
1067
1068 // ...
1069
1070 // ...
1071
1072 // ...
1073
1074 // ...
1075
1076 // ...
1077
1078 // ...
1079
1080 // ...
1081
1082 // ...
1083
1084 // ...
1085
1086 // ...
1087
1088 // ...
1089
1090 // ...
1091
1092 // ...
1093
1094 // ...
1095
1096 // ...
1097
1098 // ...
1099
1100 // ...
1101
1102 // ...
1103
1104 // ...
1105
1106 // ...
1107
1108 // ...
1109
1110 // ...
1111
1112 // ...
1113
1114 // ...
1115
1116 // ...
1117
1118 // ...
1119
1120 // ...
1121
1122 // ...
1123
1124 // ...
1125
1126 // ...
1127
1128 // ...
1129
1130 // ...
1131
1132 // ...
1133
1134 // ...
1135
1136 // ...
1137
1138 // ...
1139
1140 // ...
1141
1142 // ...
1143
1144 // ...
1145
1146 // ...
1147
1148 // ...
1149
1150 // ...
1151
1152 // ...
1153
1154 // ...
1155
1156 // ...
1157
1158 // ...
1159
1160 // ...
1161
1162 // ...
1163
1164 // ...
1165
1166 // ...
1167
1168 // ...
1169
1170 // ...
1171
1172 // ...
1173
1174 // ...
1175
1176 // ...
1177
1178 // ...
1179
1180 // ...
1181
1182 // ...
1183
1184 // ...
1185
1186 // ...
1187
1188 // ...
1189
1190 // ...
1191
1192 // ...
1193
1194 // ...
1195
1196 // ...
1197
1198 // ...
1199
1200 // ...
1201
1202 // ...
1203
1204 // ...
1205
1206 // ...
1207
1208 // ...
1209
1210 // ...
1211
1212 // ...
1213
1214 // ...
1215
1216 // ...
1217
1218 // ...
1219
1220 // ...
1221
1222 // ...
1223
1224 // ...
1225
1226 // ...
1227
1228 // ...
1229
1230 // ...
1231
1232 // ...
1233
1234 // ...
1235
1236 // ...
1237
1238 // ...
1239
1240 // ...
1241
1242 // ...
1243
1244 // ...
1245
1246 // ...
1247
1248 // ...
1249
1250 // ...
1251
1252 // ...
1253
1254 // ...
1255
1256 // ...
1257
1258 // ...
1259
1260 // ...
1261
1262 // ...
1263
1264 // ...
1265
1266 // ...
1267
1268 // ...
1269
1270 // ...
1271
1272 // ...
1273
1274 // ...
1275
1276 // ...
1277
1278 // ...
1279
1280 // ...
1281
1282 // ...
1283
1284 // ...
1285
1286 // ...
1287
1288 // ...
1289
1290 // ...
1291
1292 // ...
1293
1294 // ...
1295
1296 // ...
1297
1298 // ...
1299
1300 // ...
1301
1302 // ...
1303
1304 // ...
1305
1306 // ...
1307
1308 // ...
1309
1310 // ...
1311
1312 // ...
1313
1314 // ...
1315
1316 // ...
1317
1318 // ...
1319
1320 // ...
1321
1322 // ...
1323
1324 // ...
1325
1326 // ...
1327
1328 // ...
1329
1330 // ...
1331
1332 // ...
1333
1334 // ...
1335
1336 // ...
1337
1338 // ...
1339
1340 // ...
1341
1342 // ...
1343
1344 // ...
1345
1346 // ...
1347
1348 // ...
1349
1350 // ...
1351
1352 // ...
1353
1354 // ...
1355
1356 // ...
1357
1358 // ...
1359
1360 // ...
1361
1362 // ...
1363
1364 // ...
1365
1366 // ...
1367
1368 // ...
1369
1370 // ...
1371
1372 // ...
1373
1374 // ...
1375
1376 // ...
1377
1378 // ...
1379
1380 // ...
1381
1382 // ...
1383
1384 // ...
1385
1386 // ...
1387
1388 // ...
1389
1390 // ...
1391
1392 // ...
1393
1394 // ...
1395
1396 // ...
1397
1398 // ...
1399
1400 // ...
1401
1402 // ...
1403
1404 // ...
1405
1406 // ...
1407
1408 // ...
1409
1410 // ...
1411
1412 // ...
1413
1414 // ...
1415
1416 // ...
1417
1418 // ...
1419
1420 // ...
1421
1422 // ...
1423
1424 // ...
1425
1426 // ...
1427
1428 // ...
1429
1430 // ...
1431
1432 // ...
1433
1434 // ...
1435
1436 // ...
1437
1438 // ...
1439
1440 // ...
1441
1442 // ...
1443
1444 // ...
1445
1446 // ...
1447
1448 // ...
1449
1450 // ...
1451
1452 // ...
1453
1454 // ...
1455
1456 // ...
1457
1458 // ...
1459
1460 // ...
1461
1462 // ...
1463
1464 // ...
1465
1466 // ...
1467
1468 // ...
1469
1470 // ...
1471
1472 // ...
1473
1474 // ...
1475
1476 // ...
1477
1478 // ...
1479
1480 // ...
1481
1482 // ...
1483
1484 // ...
1485
1486 // ...
1487
1488 // ...
1489
1490 // ...
1491
1492 // ...
1493
1494 // ...
1495
1496 // ...
1497
1498 // ...
1499
1500 // ...
1501
1502 // ...
1503
1504 // ...
1505
1506 // ...
1507
1508 // ...
1509
1510 // ...
1511
1512 // ...
1513
1514 // ...
1515
1516 // ...
1517
1518 // ...
1519
1520 // ...
1521
1522 // ...
1523
1524 // ...
1525
1526 // ...
1527
1528 // ...
1529
1530 // ...
1531
1532 // ...
1533
1534 // ...
1535
1536 // ...
1537
1538 // ...
1539
1540 // ...
1541
1542 // ...
1543
1544 // ...
1545
1546 // ...
1547
1548 // ...
1549
1550 // ...
1551
1552 // ...
1553
1554 // ...
1555
1556 // ...
1557
1558 // ...
1559
1560 // ...
1561
1562 // ...
1563
1564 // ...
1565
1566 // ...
1567
1568 // ...
1569
1570 // ...
1571
1572 // ...
1573
1574 // ...
1575
1576 // ...
1577
1578 // ...
1579
1580 // ...
1581
1582 // ...
1583
1584 // ...
1585
1586 // ...
1587
1588 // ...
1589
1590 // ...
1591
1592 // ...
1593
1594 // ...
1595
1596 // ...
1597
1598 // ...
1599
1600 // ...
1601
1602 // ...
1603
1604 // ...
1605
1606 // ...
1607
1608 // ...
1609
1610 // ...
1611
1612 // ...
1613
1614 // ...
1615
1616 // ...
1617
1618 // ...
1619
1620 // ...
1621
1622 // ...
1623
1624 // ...
1625
1626 // ...
1627
1628 // ...
1629
1630 // ...
1631
1632 // ...
1633
1634 // ...
1635
1636 // ...
1637
1638 // ...
1639
1640 // ...
1641
1642 // ...
1643
1644 // ...
1645
1646 // ...
1647
1648 // ...
1649
1650 // ...
1651
1652 // ...
1653
1654 // ...
1655
1656 // ...
1657
1658 // ...
1659
1660 // ...
1661
1662 // ...
1663
1664 // ...
1665
1666 // ...
1667
1668 // ...
1669
1670 // ...
1671
1672 // ...
1673
1674 // ...
1675
1676 // ...
1677
1678 // ...
1679
1680 // ...
1681
1682 // ...
1683
1684 // ...
1685
1686 // ...
1687
1688 // ...
1689
1690 // ...
1691
1692 // ...
1693
1694 // ...
1695
1696 // ...
1697
1698 // ...
1699
1700 // ...
1701
1702 // ...
1703
1704 // ...
1705
1706 // ...
1707
1708 // ...
1709
1710 // ...
1711
1712 // ...
1713
1714 // ...
1715
1716 // ...
1717
1718 // ...
1719
1720 // ...
1721
1722 // ...
1723
1724 // ...
1725
1726 // ...
1727
1728 // ...
1729
1730 // ...
1731
1732 // ...
1733
1734 // ...
1735
1736 // ...
1737
1738 // ...
1739
1740 // ...
1741
1742 // ...
1743
1744 // ...
1745
1746 // ...
1747
1748 // ...
1749
1750 // ...
1751
1752 // ...
1753
1754 // ...
1755
1756 // ...
1757
1758 // ...
1759
1760 // ...
1761
1762 // ...
1763
1764 // ...
1765
1766 // ...
1767
1768 // ...
1769
1770 // ...
1771
1772 // ...
1773
1774 // ...
1775
1776 // ...
1777
1778 // ...
1779
1780 // ...
1781
1782 // ...
1783
1784 // ...
1785
1786 // ...
1787
1788 // ...
1789
1790 // ...
1791
1792 // ...
1793
1794 // ...
1795
1796 // ...
1797
1798 // ...
1799
1800 // ...
1801
1802 // ...
1803
1804 // ...
1805
1806 // ...
1807
1808 // ...
1809
1810 // ...
1811
1812 // ...
1813
1814 // ...
1815
1816 // ...
1817
1818 // ...
1819
1820 // ...
1821
1822 // ...
1823
1824 // ...
1825
1826 // ...
1827
1828 // ...
1829
1830 // ...
1831
1832 // ...
1833
1834 // ...
1835
1836 // ...
1837
1838 // ...
1839
1840 // ...
1841
1842 // ...
1843
1844 // ...
1845
1846 // ...
1847
1848 // ...
1849
1850 // ...
1851
1852 // ...
1853
1854 // ...
1855
1856 // ...
1857
1858 // ...
1859
1860 // ...
1861
1862 // ...
1863
1864 // ...
1865
1866 // ...
1867
1868 // ...
1869
1870 // ...
1871
1872 // ...
1873
1874 // ...
1875
1876 // ...
1877
1878 // ...
1879
1880 // ...
1881
1882 // ...
1883
1884 // ...
1885
1886 // ...
1887
1888 // ...
1889
1890 // ...
1891
1892 // ...
1893
1894 // ...
1895
1896 // ...
1897
1898 // ...
1899
1900 // ...
1901
1902 // ...
1903
1904 // ...
1905
1906 // ...
1907
1908 // ...
1909
1910 // ...
1911
1912 // ...
1913
1914 // ...
1915
1916 // ...
1917
1918 // ...
1919
1920 // ...
1921
1922 // ...
1923
1924 // ...
1925
1926 // ...
1927
1928 // ...
1929
1930 // ...
1931
1932 // ...
1933
1934 // ...
1935
1936 // ...
1937
1938 // ...
1939
1940 // ...
1941
1942 // ...
1943
1944 // ...
1945
1946 // ...
1947
1948 // ...
1949
1950 // ...
1951
1952 // ...
1953
1954 // ...
1955
1956 // ...
1957
1958 // ...
1959
1960 // ...
1961
1962 // ...
1963
1964 // ...
1965
1966 // ...
1967
1968 // ...
1969
1970 // ...
1971
1972 // ...
1973
1974 // ...
1975
1976 // ...
1977
1978 // ...
1979
1980 // ...
1981
1982 // ...
1983
1984 // ...
1985
1986 // ...
1987
1988 // ...
1989
1990 // ...
1991
1992 // ...
1993
1994 // ...
1995
1996 // ...
1997
1998 // ...
1999
2000 // ...
2001
2002 // ...
2003
2004 // ...
2005
2006 // ...
2007
2008 // ...
2009
2010 // ...
2011
2012 // ...
2013
2014 // ...
2015
2016 // ...
2017
2018 // ...
2019
2020 // ...
2021
2022 // ...
2023
2024 // ...
2025
2026 // ...
2027
2028 // ...
2029
2030 // ...
2031
2032 // ...
2033
2034 // ...
2035
2036 // ...
2037
2038 // ...
2039
2040 // ...
2041
2042 // ...
2043
2044 // ...
2045
2046 // ...
2047
2048 // ...
2049
2050 // ...
2051
2052 // ...
2053
2054 // ...
2055
2056 // ...
2057
2058 // ...
2059
2060 // ...
2061
2062 // ...
2063
2064 // ...
2065
2066 // ...
2067
2068 // ...
2069
2070 // ...
2071
2072 // ...
2073
2074 // ...
2075
2076 // ...
2077
2078 // ...
2079
2080 // ...
2081
2082 // ...
2083
2084 // ...
2085
2086 // ...
2087
2088 // ...
2089
2090 // ...
2091
2092 // ...
2093
2094 // ...
2095
2096 // ...
2097
2098 // ...
2099
2100 // ...
2101
2102 // ...
2103
2104 // ...
2105
2106 // ...
2107
2108 // ...
2109
2110 // ...
2111
2112 // ...
2113
2114 // ...
2115
2116 // ...
2117
2118 // ...
2119
2120 // ...
2121
2122 // ...
2123
2124 // ...
2125
2126 // ...
2127
2128 // ...
2129
2130 // ...
2131
2132 // ...
2133
2134 // ...
2135
2136 // ...
2137
2138 // ...
2139
2140 // ...
2141
2142 // ...
2143
2144 // ...
2145
2146 // ...
2147
2148 // ...
2149
2150 // ...
2151
2152 // ...
2153
2154 // ...
2155
2156 // ...
2157
2158 // ...
2159
2160 // ...
2161
2162 // ...
2163
2164 // ...
2165
2166 // ...
2167
2168 // ...
2169
2170 // ...
2171
2172 // ...
2173
2174 // ...
2175
2176 // ...
2177
2178 // ...
2179
2180 // ...
2181
2182 // ...
2183
2184 // ...
2185
2186 // ...
2187
2188 // ...
2189
2190 // ...
2191
2192 // ...
2193
2194 // ...
2195
2196 // ...
2197
2198 // ...
2199
2200 // ...
2201
2202 // ...
2203
2204 // ...
2205
2206 // ...
2207
2208 // ...
2209
2210 // ...
2211
2212 // ...
2213
2214 // ...
2215
2216 // ...
2217
2218 // ...
2219
2220 // ...
2221
2222 // ...
2223
2224 // ...
2225
2226 // ...
2227
2228 // ...
2229
2230 // ...
2231
2232 // ...
2233
2234 // ...
2235
2236 // ...
2237
2238 // ...
2239
2240 // ...
2241
2242 // ...
2243
2244 // ...
2245
2246 // ...
2247
2248 // ...
2249
2250 // ...
2251
2252 // ...
2253
2254 // ...
2255
2256 // ...
2257
2258 // ...
2259
2260 // ...
2261
2262 // ...
2263
2264 // ...
2265
2266 // ...
2267
2268 // ...
2269
2270 // ...
2271
2272 // ...
2273
2274 // ...
2275
2276 // ...
2277
2278 // ...
2279
2280 // ...
2281
2282 // ...
2283
2284 // ...
2285
2286 // ...
2287
2288 // ...
2289
2290 // ...
2291
2292 // ...
2293
2294 // ...
2295
2296 // ...
2297
2298 // ...
2299
2300 // ...
2301
2302 // ...
2303
2304 // ...
2305
2306 // ...
2307
2308 // ...
2309
2310 // ...
2311
2312 // ...
2313
2314 // ...
2315
2316 // ...
2317
2318 // ...
2319
2320 // ...
2321
2322 // ...
2323
2324 // ...
2325
2326 // ...
2327
2328 // ...
2329
2330 // ...
2331
2332 // ...
2333
2334 // ...
2335
2336 // ...
2337
2338 // ...
2339
2340 // ...
2341
2342 // ...
2343
2344 // ...
2345
2346 // ...
2347
2348 // ...
2349
2350 // ...
2351
2352 // ...
2353
2354 // ...
2355
2356 // ...
2357
2358 // ...
2359
2360 // ...
2361
2362 // ...
2363
2364 // ...
2365
2366 // ...
2367
2368 // ...
2369
2370 // ...
2371
2372 // ...
2373
2374 // ...
2375
2376 // ...
2377
2378 // ...
2379
2380 // ...
2381
2382 // ...
2383
2384 // ...
```



*Screenshot 2: Monitor portu szeregowego*



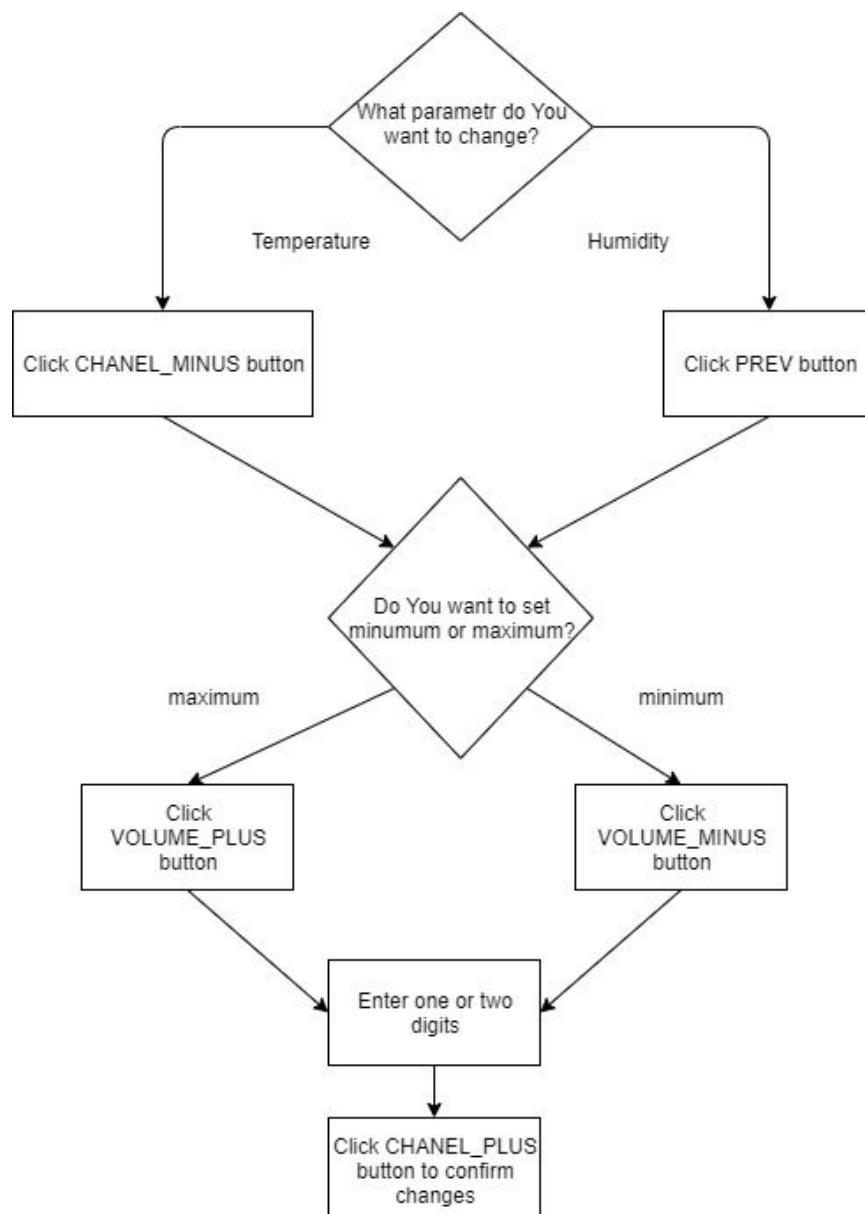
*Screenshot 3: Okno z działającym projektem*



*Screenshot 4: Strona generowana przez urządzenie*

Wygenerowany adres IP należy otworzyć w przeglądarce internetowej. Dane o temperaturze i wilgotności są uaktualniane co sekundę. Dodatkowo rysowany jest wykres z wynikami 6 ostatnich pomiarów temperatury. Użytkownik może teraz przystąpić do konfiguracji powiadomień o niekorzystnych warunkach pogodowych.

Urządzenie umożliwia ustawienie dopuszczalnych wartości temperatury i wilgotności. W przypadku przekroczenia tych wartości w zależności od konfiguracji końcowej zaświeci się dioda lub zacznie brzęczeć buzzer. Wyłączenie powiadomiania i powrót do początkowych wartości limitów następuje poprzez naciśnięcie pinu dotykowego numer 4, podpisanego na płytce jako D13. Poniżej znajduje się rysunkowa instrukcja w jaki sposób możemy ustawić limit temperatury. Należy wspomnieć, iż często z powodu zakłóceń nie udaje się poprawnie odczytywać sygnałów z pilota. W obecnej wersji programu użytkownik może wspierać się informacjami na przesyłanymi na przez łącze Serial.



# Przeprowadzone prace, napotkane problemy oraz dalsze możliwości rozwoju

## Czynności wstępne

Tak jak wspominaliśmy zdecydowaliśmy się używać środowiska Arduino IDE. Jest ono łatwe w obsłudze i umożliwia wygodne zarządzanie bibliotekami. W internecie można znaleźć wiele przydatnych poradników jak tworzyć programy na ESP32 w tym środowisku [dok]. W systemach Linux Mint i Ubuntu było konieczne wpisanie w terminalu komendy `sudo chmod 666 ttyUSB0`, by możliwy był zapis i odczyt do portu szeregowego za pomocą USB. Niestety, czynność należało powtarzać po każdym odłączeniu i podłączeniu płytki. Uniknięcie tej niewygodnej procedury jest możliwe przez odpowiednią konfigurację katalogu `/etc/udev/rules.d`. Częstym błędem był brak zainstalowanego modułu Pythona o nazwie `serial`, rozwiązany komendą `pip install pyserial`.

## System FreeRTOS

Ciekawym zjawiskiem był fakt, iż w pewnej fazie rozwoju programu podczas uruchamiania wszystkich zadań procesów na jednym rdzeniu - numer 0, wszystkie czujniki prawidłowo działały oprócz czujnika DHT, który pokazywał, iż jest błąd odczytu. Po uruchomieniu go na innym rdzeniu problem znikł. Niestety problem ponownie wystąpił wraz ze wzrostem rozmiaru programu, ale jest on znacznie rzadszy. Ten przykład, iż możliwości rdzeni są ograniczone i dobrze, że wykorzystujemy je oba. Należy pamiętać też, aby usypiać zadania, na przykład służące do odczytu co pewien czas wartości z czujników, aby inne zadania o niższym priorytecie nie zostały zgłodzone.

Wprowadzenie semaforów było spowodowane tym, że zdarzyła nam się sytuacja, kiedy dwa zadania wywoływały funkcje `Serial.println()`, że dało się zaobserwować przeplot informacji. Napis "Failed to read from DHT sensor!" pochodzi z zadania funkcji odpowiedzialnej za odczyt wilgotności, natomiast pozostałe wartości liczbowe z funkcji odpowiedzialnej za pomiar temperatury.

```
Failed to read from DHT sensor!  
Failed to read from DHT sensor!22.44°C  
Failed to read from DHT sensor!  
  
Failed to read from DHT sensor!72  
Failed to read from DHT sensor!.  
Failed to read from DHT sensor!3  
Failed to read from DHT sensor!9°F  
Failed to read from DHT sensor!  
x Failed to read from DHT sensor!  
x Failed to read from DHT sensor!
```

Możliwością rozwoju byłoby wykorzystanie możliwości takich jak oszczędzanie energii oraz przechodzenie całego urządzenia w tryb uśpienia [11].

## Mechanizm przerwań oraz obsługa pilota

Warto zdawać sobie sprawę, iż w przypadku zdarzenia (naciśnięcie guzika pilota), które prowadzi do przerywania w rzeczywistości wystąpi wiele przerwań i funkcja powiązana z danym przerywaniem nie zostanie wywołana tylko jeden raz. Trzeba mieć to na uwadze przy tworzeniu funkcji wywoływanej przy przerywaniu. Obszarem, który można poprawić to działanie pilota.

## Czujniki wartości

Należy dokładnie sprawdzić przed podpięciem czujnika DHT czy wszystkie kable są dobrze podpięte, ponieważ czujnik jest dość wrażliwy. Zdarzyło nam się źle podpiąć kable, co spowodowało stopienie się części plastiku i wyświetlanie nieprawidłowych wartości. Wszystkie wykorzystane przez mierzą wartości z pewną dokładnością pomiarową. Wyniki byłyby dokładniejsze, gdybyśmy brali średnią z ostatnich kilku wyników. Czujnik pomiaru wilgotności gleby ze względu na słabą jakość dołączonych do zestawu kabli męsko-żeńskich często nie jest dobrze podpięty. Należy zwrócić uwagę, aby na module głównym świeciło się obie diody.

## WiFi

Istnieje wiele bibliotek umożliwiających stworzenie serwera działającego na urządzeniach programowalnych w środowisku Arduino. Zdecydowaliśmy się na rozwiązanie umiarkowanie proste, jednak w pełni wystarczające do wysyłania zapytań podobnych w swojej formie do RESTa. Praca w jednej sieci zapewnia generowanie tego samego lokalnego adresu IP przy każdym połączeniu. Rozwój funkcjonalności sieciowych byłby zorientowany na ustawienie stałego adresowania między sieciami, a w dalszej kolejności na możliwość komunikacji z urządzeniem z dowolnego miejsca na Ziemi.

## Wersja zaawansowana urządzenia

Urządzenie mogłoby być wykorzystywane również bez ciągłego udziału komputera. Komputer byłby potrzebny tylko, aby wgrać program, natomiast urządzenie byłoby podpięte do koszyka na baterie, dzięki czemu byłoby zasilane. Dodatkowo mogłaby być dołączona karta pamięci, która by przechowywała zmierzone wartości zważywszy na rozmiar danych właściwie na bardzo długi okres.

## Repozytorium

W serwisie github stworzyliśmy zdalne repozytorium pod adresem: <https://github.com/kbieniasz/WeatherStation>

Kod stworzonego przez nas programu znajduje się w podfolderze WeatherStation. W folderze libraries znajdują się biblioteki, które wykorzystujemy w programie. Zdecydowaliśmy się je dołączyć do repozytorium, żeby ułatwić postronnym osobom wtórne wykorzystanie projektu. Docelowo w folderze będzie znajdować się powyższa dokumentacja oraz krótki filmik prezentujący działania urządzenia.



# Bibliografia

[1] ESP32 Hardware Reference

<https://docs.espressif.com/projects/esp-idf/en/latest/get-started/index.html>

[2] Strona sklepu z zakupionym przez nas urządzeniem

<https://botland.com.pl/pl/moduly-wifi/8893-esp32-wifi-bt-42-platforma-z-modulem-esp-wroom-32-zgodny-z-esp32-devkit.html>

[3] Schemat przedstawiający wejścia-wyjścia głównego przeznaczenia płytki ESP32-DevKit w wersji z 30 pinami

<https://pl.pinterest.com/pin/307370743313343637/>

[4] Poradnik objaśniający użycie pinów

<https://randomnerdtutorials.com/esp32-pinout-reference-gpios/>

[5] Sekcja oficjalnej dokumentacji traktująca o systemie operacyjnym FreeRTOS

<https://docs.espressif.com/projects/esp-idf/en/latest/api-reference/system/freertos.html>

[6] Poradnik o przypisywaniu zadań systemowi operacyjnemu do rdzenia

<https://randomnerdtutorials.com/esp32-dual-core-arduino-ide/>

[7] Artykuł pokazujący wykorzystanie mechanizmu przerwań

<https://randomnerdtutorials.com/esp32-pir-motion-sensor-interrupts-timers/>

[8] Artykuł pokazujący wykorzystanie czujnika Dallas

<https://randomnerdtutorials.com/esp32-ds18b20-temperature-arduino-ide/>

[9] Artykuł pokazujący wykorzystanie czujnika DHT

<https://randomnerdtutorials.com/esp32-dht11-dht22-temperature-humidity-sensor-arduino-ide/>

[10] Sklep z zakupionym przez nas czujnikiem wilgotności gleby

<https://botland.com.pl/pl/czujniki-wilgotnosci/1588-czujnik-wilgotnosci-gleby.html>

[11] Sekcja dokumentacji dotycząca trybów uśpienia

[https://docs.espressif.com/projects/esp-idf/en/latest/api-reference/system/sleep\\_modes.html](https://docs.espressif.com/projects/esp-idf/en/latest/api-reference/system/sleep_modes.html)

Inne przydatne źródła:

Dokumentacja czujnika Dallas:

<https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf>

Dokumentacja czujnika DHT:

<https://download.kamami.pl/p197416-dht11.pdf>

Dokumentacja czujnika podczerwieni

[https://botland.com.pl/index.php?controller=attachment&id\\_attachment=2454](https://botland.com.pl/index.php?controller=attachment&id_attachment=2454)

Oferta ze strony sklepu internetowego z pilotem jakim dysponujemy:

[https://botland.com.pl/pl/odbiorniki-podczerwieni/2169-pilot-ir-nec-38khz-odbiornik-podczerwieni-1838t-modul-i-przewody.html?gclid=CjwKCAiAu9vwBRAEEiwAzvjq-weacX3M2AzpRvX1s4P-aZyMZQywbGXB7nioUxsbA4aQbKQlOPJ9hoCvo4QAvD\\_BwE](https://botland.com.pl/pl/odbiorniki-podczerwieni/2169-pilot-ir-nec-38khz-odbiornik-podczerwieni-1838t-modul-i-przewody.html?gclid=CjwKCAiAu9vwBRAEEiwAzvjq-weacX3M2AzpRvX1s4P-aZyMZQywbGXB7nioUxsbA4aQbKQlOPJ9hoCvo4QAvD_BwE)

Przykład wykorzystania pilota i czujnika podczerwieni:

<http://www.esp32learning.com/code/esp32-and-infrared-receiver-example.php>

Zdjęcia, których nie jesteśmy autorami:

Dallas: <https://www.addicore.com/DS18B20-Digital-Temperature-Sensor-p/161.htm>

DHT 11: <https://kamami.pl/temperatury/197416-dht11.html>

Schemat systemu i odwzorowanie adresów: wycięte z dokumentacji ESP32-DevKitC