UNIWERSYTET W BIAŁYMSTOKU
INSTYTUT INFORMATYKI

Praca licencjacka

# Zdalne monitorowanie temperatury z wykorzystaniem komputera jednopłytkowego i czujnika DS18B20

Dawid Treszczotko
Numer albumu: 78323

Promotor:
dr hab. inż. Marek Parfieniuk

Białystok 2022

# Spis treści

# Wstęp

Niniejsza praca licencjacka dotyczy zaprojektowania aplikacji, która pozwala zdalnie monitorować temperaturę oraz sterować urządzeniem regulującym jej wartości. Aplikacja składa się z dwóch części. Jednej działającej na komputerze jednoukładowym, a drugiej na serwerze internetowym. Zadaniem pierwszej jest odczytywanie wartości z czujnika temperatury oraz nadzorowanie przekaźnika, który może dołączać lub odłączać napięcie zasilania do urządzenia regulacyjnego. Serwer zapewnia zdalny dostęp do komputera jednoukładowego oraz gromadzi dane o zmianach temperatury i stanu przełącznika. Pozwala obserwować zmiany temperatury i włączać lub wyłączać przekaźnik w razie potrzeby. W tym celu zapewnia graficzny interfejs użytkownika w formie strony WWW.

W pracy wykorzystano komputer jednoukładowy ESP32 WROOM, który jest zgodny ze standardem Arduino, ale posiada na wyposażeniu moduł do komunikacji bezprzewodowej WIFi. Do mierzenia temperatury posłużył zewnętrzny czujnik DS18S20. Zadaniem autora było opracowanie nie tylko programów, ale też połączeń urządzeń elektronicznych.

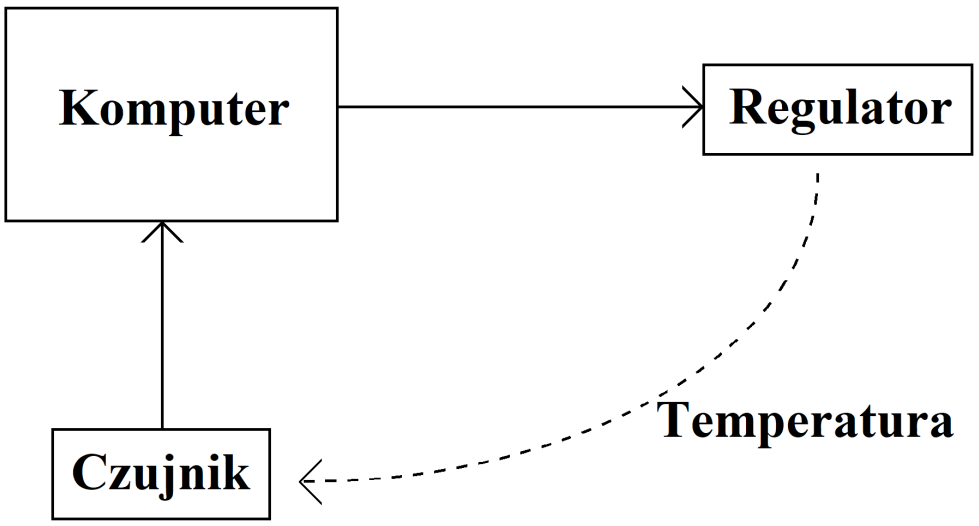
Praca wpisuje się w aktualne trendy technologiczne. Komputery jednoukładowe są obecnie powszechnie wykorzystywane do opracowywania aplikacji, w których oprogramowanie analizuje i wpływa na otoczenie systemu informatycznego. Opracowany system można też zaliczyć do "Internetu rzeczy"(ang. Internet of things), który zakłada powszechne łączenie urządzeń w celu koordynacji działania i gromadzenia kompleksowych danych.

# Rozdział 1

# Pomiar temperatury z użyciem komputera jednopłytkowego

## 1.1 Podstawowy schemat sytemu regulacji temperatury

Jednym z zastosowań komputerów jest sterowanie urządzeniami. Między innymi komputery umożliwiają regulowanie temperatury pomieszczeń oraz nadzorowanie nagrzewania i chłodzenia w procesach produkcji przemysłowej. Użycie komputera pozwala automatyzować regulowanie temperatury, zwiększać dokładność pomiaru i szybkość reakcji, oraz nadzorować sprawnie, zdalnie dużą liczbę układów regulacji [1].




Rysunek 1.1: Ogólny schemat systemu regulującego temperaturę

Podstawowy schemat systemu regulującego temperaturę został pokazany na rysunku 1.1. Wykorzystuje on komputer, czujnik temperatury oraz regulator. Czujnik jest urządzeniem, przekształcającym wartość fizyczną temperatury w sygnał, który można wprowadzić do komputera. Regulator jest układem oddziałującym na temperaturę obiektu lub miejsca, które należy ogrzewać lub chłodzić. Komputer natomiast na podstawie informacji z czujnika podejmuje decyzję co zrobić z urządzeniem regulującym i generuje dla niego odpowiedni sygnał sterujący.

## 1.2 Komputer jednopłytkowy

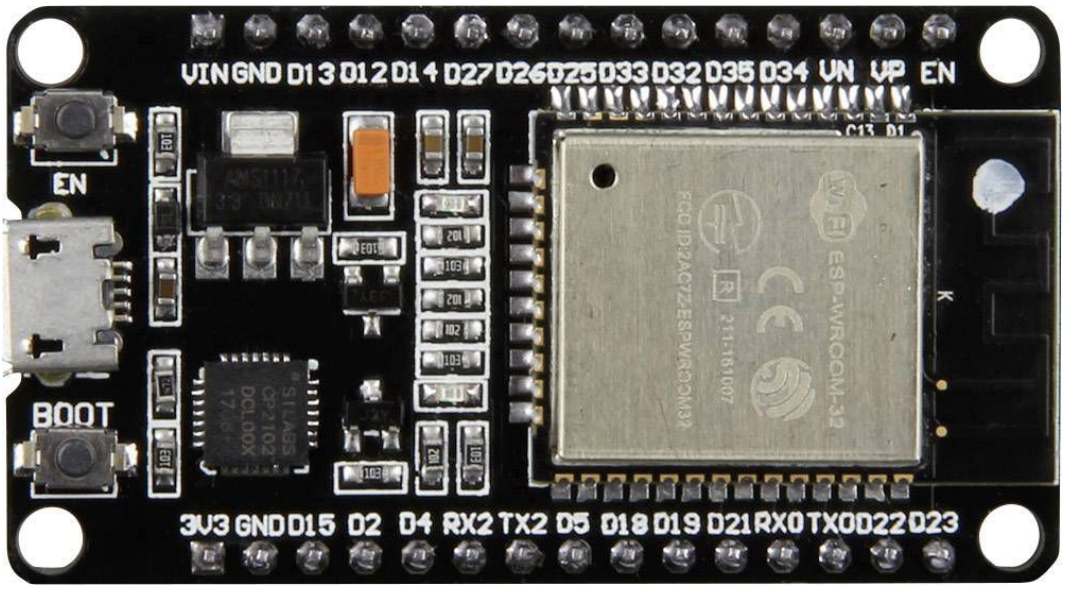
W przypadku regulacji temperatury oraz innych zastosowań związanych ze sterowaniem urządzeniami zwykle nie stosuje się normalnych komputerów osobistych czy serwerów. Ich miejsce zastępują komputery jednopłytkowe (ang. single-board computer) [2]. Urządzenia te cechują się ogólnie:

- niską ceną
- brakiem lub ograniczeniami interfejsu graficznego
- niewielkim rozmiarem
- niezawodnością
- trwałością

Na rynku istnieje wiele konstrukcji zbudowanych na jednej płytce drukowanej, które różnią się ceną, rozmiarami, wyposażeniem i innymi cechami użytkowymi [3]. Możemy znaleźć urządzenia o wyjątkowo małych rozmiarach, takie jak np. Arduino Nano. Istnieją też komputery stworzone z myślą o pracy takie jak ConnectCore 6, które nadają się do pracy w skrajnych temperaturach. Niektóre charakteryzują się słabą mocą obliczeniową, nie wykorzystują systemu operacyjnego, ale długo pracują na baterii. Inne natomiast mają możliwości komputera osobistego, obsługują system Linux, przy rozmiarze podobnym do karty kredytowej, np. Raspberry Pi.

## 1.3 Komputer ESP32-WROOM

Przykładem ciekawego komputera jedno-układowego jest ESP-WROOM-32 [4] (`http://esp32.net/`) pokazany na rysunku 1.2. Posiada on zalety w postaci wbudowanego modułu Wi-Fi i czujników, przestępnej ceny oraz popularności.




Rysunek 1.2: Komputer jednopłytkowy ESP32 w formie ESP-WROOM-32 (Źródło: `https://asset.conrad.com/media10/isa/160267/c1/-/pl/1656367_BB_00_FB/image.jpg`)

Wbudowane standardy połączeń ułatwiają tworzenie systemów klasy IoT [5]. Komputer posiadający zaimplementowany interfejs Wi-Fi nie wymaga dołączania zewnętrznych urządzeń komunikacyjnych. Dzięki takiej konfiguracji unika się dodatkowej pracy i nakładów potrzebnych do połączenia komputera z siecią. ESP32-WROOM posiada wbudowane Wi-Fi wykorzystujące standardy 802.11 b/g/n. Połączenie z siecią odbywa się poprzez wpisanie danych routera do pamięci komputera.

Zaletą wybranego zestawu rozwojowego ESP32, czyli „ESP32 Starter Kit - zestaw startowy z modułem WiFi" (https://botland.com.pl/moduly-wifi-i-bt-esp32/19291-esp32-starter-kit-zestaw-startowy-z-modulem-wifi-esp32-5903351242950.html) jest dołączenie praktycznych modułów. Zawiera on czujnik temperatury, przełącznik, płytkę stykową oraz potrzebne połączenia. Pozwala to uniknąć problemów z pracami lutowniczymi, które mogłyby wykraczać poza kompetencje autora.

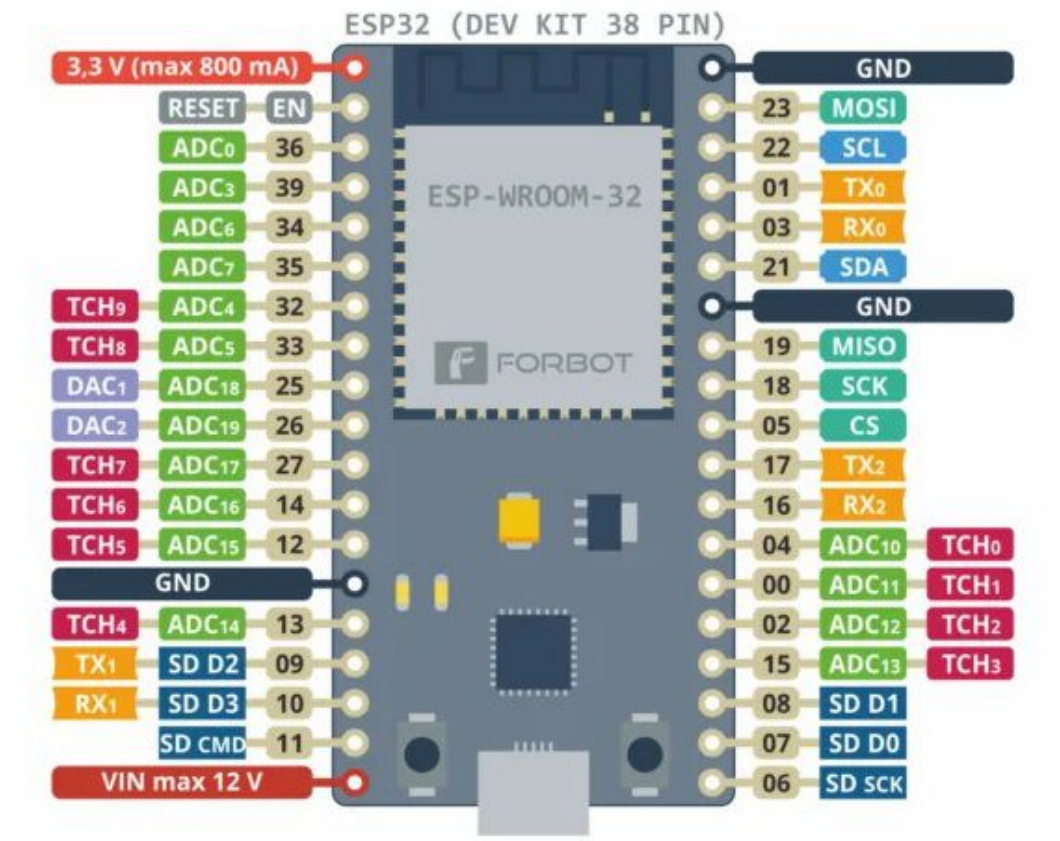
Cena komputera nie jest wygórowana, wynosi około 150 złotych. Dlatego urządzenie jest bardzo popularne. W porównaniu do innych komputerów jednoukładowych ESP32 jest urządzeniem ekonomicznym, daje duże możliwości za relatywnie niskie pieniądze.

Płytka ESP32 jest mała, o wymiarach 50 x 26 x 8 milimetrów. Możliwe jest zatem konstruowanie urządzeń zminiaturyzowanych.

Urządzenie jet wyposażone w port micro USB, którym jest zasilane. Średnio potrzebuje prądu o natężeniu 80 mA. Jednoczenie wymaga napięcia 5 V.

U dołu ESP32 znajduje się 30 portów ogólnego przeznaczenia, tak zwanych

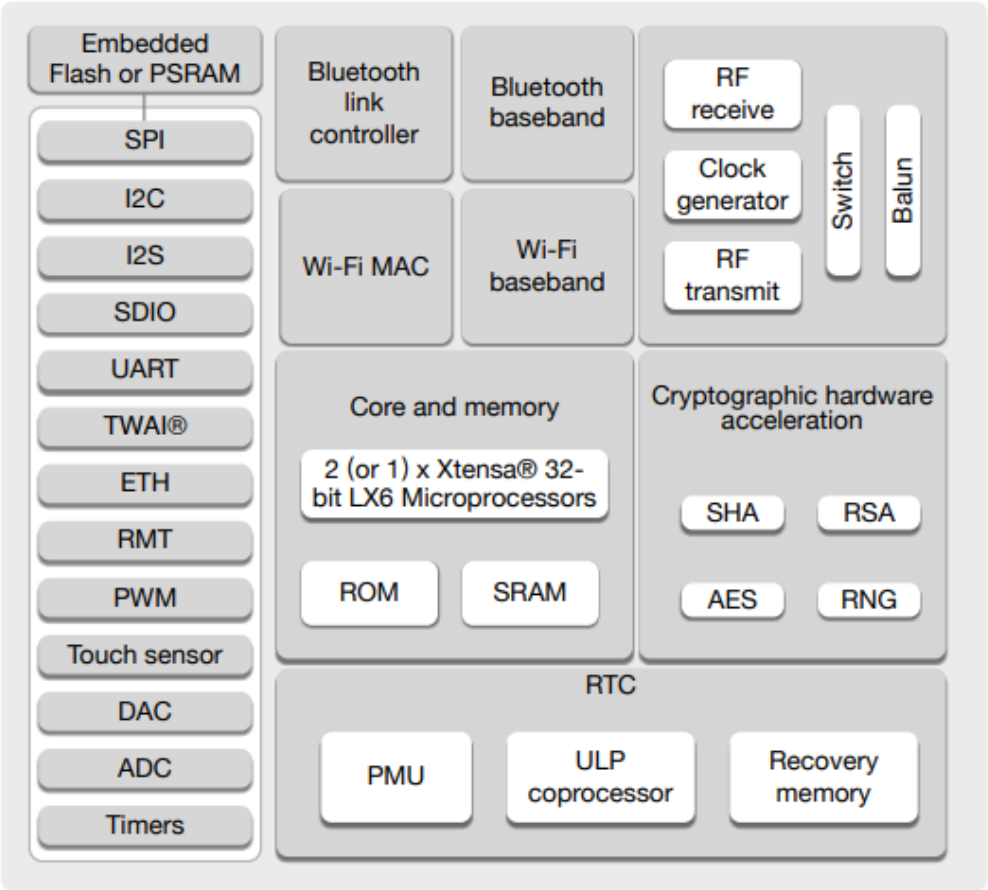
GPIO. Pozwalają one na wymianę informacji z innymi modułami lub systemami. Standardowe przeznaczenie poszczególnych pinów objaśnia rysunku 1.3.




Rysunek 1.3: Opis wyprowadzeń modułu ESP32 DevKit (Źródło: `https://forbot.pl/blog/leksykon/esp32`)

Sercem ESP32 jest mikrokontroler w formie układu scalonego SoC (ang. System-On-Chip), który mieści jednostki obliczeniowe, pamięć i różne urządzenia peryferyjne, takie jak układy odmierzania czasu, kontrolery interfejsów, układ zarządzania zasilaniem, koprocesory komunikacyjne i kryptograficzne, czujnik Halla czy czujnik dotykowy. Schemat blokowy struktury SoCa pokazano na rysunku 1.4 (Źródło: `https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf`).

Mikrokontroler ESP32 wykorzystuje procesor dwurdzeniowy Dual Core Tensilica LX6 o taktowaniu 240 MHz. Komputer wyposażono w 520 KB pamięci SRAM, 448 KB pamięci ROM oraz 4 MB pamięci Fash.




Rysunek 1.4: Opis blokowy wnętrza procesora (Źródło: `https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf`)

## 1.4 Oprogramowanie ESP32-WROOM

Komputer ESP32 należy do rodziny Arduino i dzięki temu można korzystać z szerokiej gamy bibliotek [6] [7] [8]. Są one przydatne w programowaniu, gdyż w znaczący sposób ułatwiają implementowanie różnych rozwiązań.

Między innymi, aby skonfigurować moduł Wi-Fi można wykorzystać bibliotekę Arduino „Wifi.h". W tym celu należy do przedrostka „WiFi" dodać „.begin" oraz w nawiasach wpisać ssid, oraz hasło routera, z którym podejmowana jest próba połączenia. W przypadku zarejestrowania komputera w sieci istnieje możliwość pobrania jego lokalnego IP. W tym celu stosuje się metodę „localIP" z przedrostkiem „WiFi".

Urządzenie jedynie połączone z routerem nie daje zbyt wielkich możliwości. W celu komunikacji klient-serwer stosuje się bibliotekę „ESPAsyncWebServer.h". Z jej wykorzystaniem płytka ma możliwość odbierania zgłoszeń od klientów i odpowiadania na nie. Biblioteka wykorzystuje asynchroniczność, dzięki czemu istnieje moż-

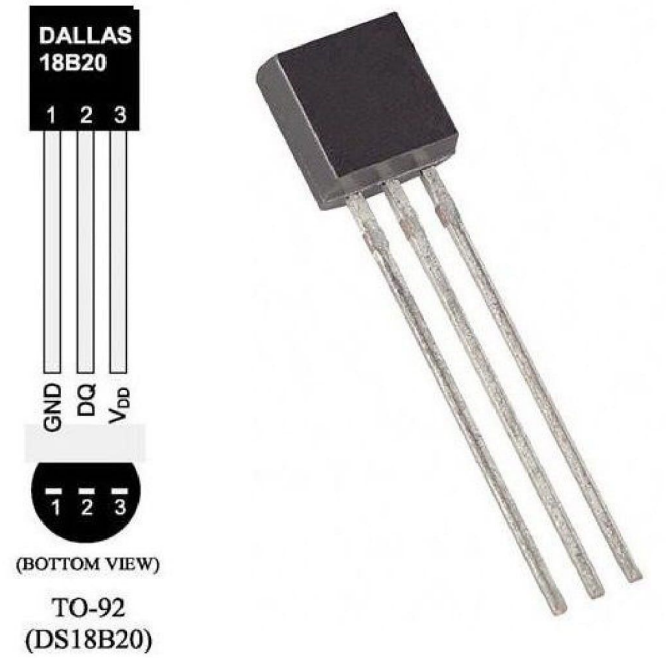
liwość obsługi wielu zapytań jednocześnie.

Warto wspomnieć też o dwóch bibliotekach ułatwiających pracę z czujnikiem temperatury. Mowa o „OnewWire.h” (`https://www.pjrc.com/teensy/td_libs_OneWire.html`) oraz „DallasTemperature.h” (`https://github.com/milesburton/Arduino-Temperature-Control-Library`). Za ich pośrednictwem w łatwy sposób dokonuje się odczytu. Odbywa się to poprzez wywołanie metody „sensors.requestTemperatures()”, a następnie „sensors.getTempCByIndex(0)”. Wartością w nawiasie określa się index kolejnego czujnika, z którego czytamy temperaturę.

## 1.5 Czujnik temperatury DS18B20

Komputer ESP32 pomimo wielu zalet nie posiada czujnika temperatury otoczenia. Aby stworzyć system regulujący temperaturę opisany w rozdziale 1.1, należało dobrać i dołączyć urządzenie zewnętrzne. Tak jak w przypadku komputerów, na rynku znajdziemy wiele konstrukcji [1].

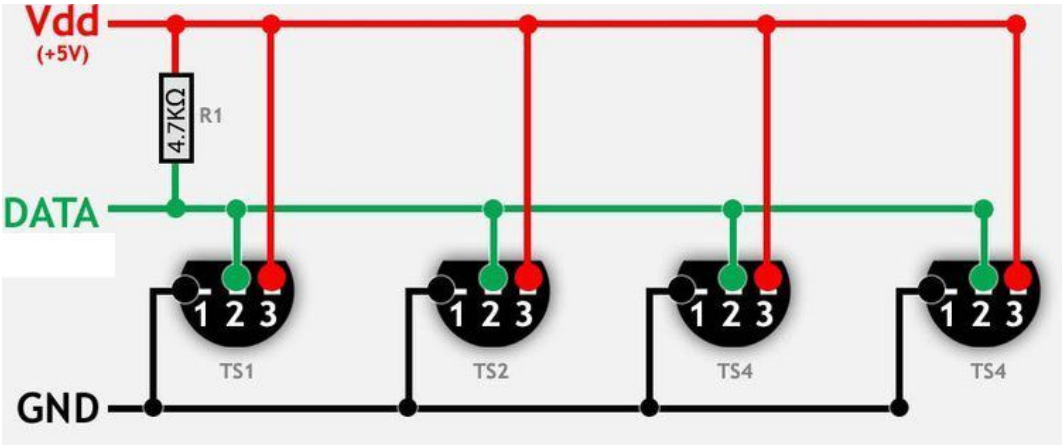
Jednym z bardziej popularnych czujników jest DS18B20 [9]. Czujnik ten jest bardzo rozpowszechniony, ponieważ ma dobre parametry odczytu wartości, prosty interfejs komunikacji oraz relatywnie niską cenę, około 20 złotych.




Rysunek 1.5: Czujnik temperatury DS18B20 w obudowie TO92 (Źródło: `https://www.images.izi.ua/122630019` )

Urządzenie charakteryzuje się bardzo dużym zakresem odczytu. Jest w stanie mierzyć temperatury od -50 °C do 125 °C. Jego dokładność szacuje się na poziomie 0,5 °C dla zakresu od -10 °C do 85 °C. W szerszym zakresie od -30 °C do 100 °C dokładność jest to 1 °C. Pomiar pozostałych temperatur może być obarczony przekłamaniami na poziomie 2 °C.

DS18B20 możemy spotkać w obudowie TO92, którą pokazano na rysunku 1.5. Z czujnika wyprowadzone są trzy nóżki. Skrajne odpowiadają za zasilenie napięciem oraz uziemienie. Środkowa odpowiada za wymianę danych i to właśnie nią wysyłane są informacje o odczycie.




Rysunek 1.6: Przyłączenie wielu czujników do jednej linii danych w technologii „1-Wire” ( Link: `www.abcrc.yourtechnicaldomain.com/data/gfx/pictures/large/9/7/6179_9.jpg?28621` )

Komunikacja z czujnikiem odbywa się poprzez interfejs „1-Wire”[10]. Jego zaimplementowanie pozwala na wykorzystanie jednej linii danych dla wielu czujników. Zostało to pokazane na rysunku 1.6. Poszczególne czujniki identyfikuje się poprzez unikalny 64-bitowy kod szeregowy. Technologia „1-Wire” pozwala również na zastosowanie tak zwanego „zasilania pasożytniczego”. Polega ono na braku połączenia z napięciem poprzez jedną ze skrajnych nóżek. Prąd potrzebny do wykonania pomiaru oraz odpowiedzi jest magazynowany w urządzeniu, gdy przychodzi do niego prośba o odczyt temperatury. Dzieje się tak dzięki wewnętrznemu kondensatorowi. Takie rozwiązanie jest bardzo korzystne, gdy istotne jest zmniejszenie ilości połączeń dochodzących do czujnika.

## 1.6 Cel pracy

Celem niniejszej pracy jest utworzenie systemu pozwalającego na zdalne monitorowanie oraz regulowanie temperatury. Ze względu na zalety urządzenia wymienione w punkcie 1.2 wybrano do pracy komputer ESP32. Ze względu na cechy i funkcjonalności opisane w punkcie 1.5 wykorzystano czujnik temperatury D18B20. Przyjęto założenie, że do obsługi systemu zostanie użyty samodzielnie opracowany serwer pośredniczący. Obsługę systemu ma zapewnić graficzny interfejs użytkownika w formie strony WWW.

Przyjęto, że komputer jednoukładowy zostanie bezpośrednio połączony z czujnikiem oraz przekaźnikiem. Będzie on pełnił dwie funkcje. Jedna z nich to komunikacja z czujnikiem oraz odczyt temperatury. Druga zaś to sterowanie przełącznikiem.

W projekcie zdecydowano się na diody LED do symulowania urządzenia grzejnego, które ma być sterowane przekaźnikiem. Ma to na celu zapewnienie bezpieczeństwa w czasie rozwijania systemu, np. zminimalizowania pożaru, porażenia i oparzenia. Prawdziwa grzałka działałaby pod niebezpiecznym napięciem 230 wolt, marnowałaby duży prąd, a jej włączenie i wyłączenie byłoby trudne do zauważenia.

Przyjęto, że pomiędzy użytkownikiem a komputerem będzie znajdował się serwer pośredni. Jego zadaniem będzie ciągła kontrola połączenia komputera z siecią oraz zapis aktualnej temperatury i stanu przełącznika. Serwer umożliwi też obsługiwanie systemu bez potrzeby instalowania żadnego oprogramowania po stronie użytkownika.

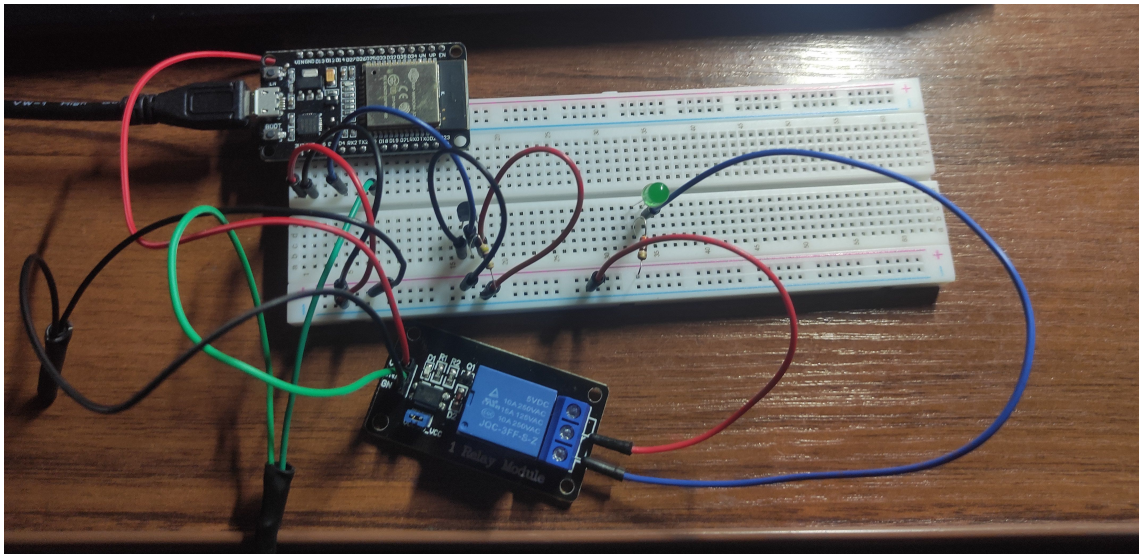
Założono, że interfejsem użytkownika będzie strona WWW. Umożliwi ona konfigurowanie serwera, zdalne sterowanie przekaźnikiem oraz graficzne przedstawienie gromadzonych danych. Użytkownik będzie mógł monitorować dane na bieżąco oraz analizować względem poprzednio zarejestrowanych.

# Rozdział 2

# Architektura systemu

## 2.1 Połączenie komputera ESP32 z czujnikiem i przekaźnikiem

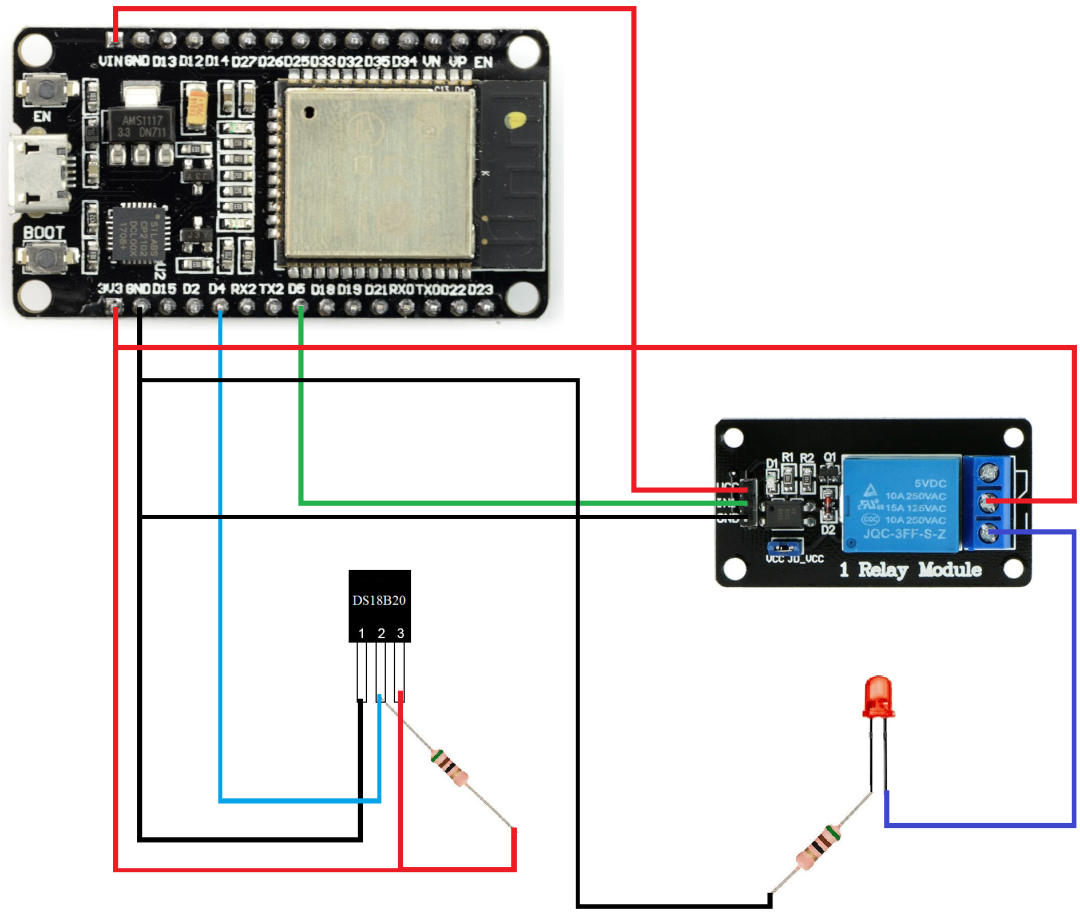
Abstrakcyjny schemat systemu regulacji temperatury przedstawiony na rysunku 1.1 w niniejszej pracy przyjął fizyczną formę przedstawioną na fotografii 2.1. Szczegółowy schemat połączeń przedstawiono na rysunku 2.2.



Rysunek 2.1: Zdjęcie połączenia komputera z czujnikiem i przełącznikiem

Do wykonania połączeń wykorzystano płytkę stykową oraz przewody dołączone do zestawu [11]. Dzięki temu elementy można w przyszłości rozłączyć i wykorzystać w innych projektach.

Czujnik temperatury wykorzystuje trzy połączenia. Skrajne jego nóżki odpowiadają za zasilanie. Lewa została połączona z uziemieniem, to znaczy z pinem „GND”




Rysunek 2.2: Schemat połączenia komputera z czujnikiem i przełącznikiem

na płytce ESP32. Prawą dostarczono zasilanie 3 wolt z pinu „3V3". Środkowa nóżka odpowiada za komunikację i została połączona z pinem „D4". W celu prawidłowego funkcjonowania czujnika należało dostarczyć na to wejście urządzenia napięcie 3 wolt. Jednak, aby natężenie płynącego prądu nie uszkodziło czujnika ani mikrokontrolera, należało zastosować rezystor. Jego opór to 4,7 kΩ.

Tak jak czujnik, do funkcjonowania przełącznik potrzebuje trzech połączeń. Skrajne odpowiadają za napięcie oraz uziemienie. Ujemny biegun zasilania, tak jak przy czujniku, jest połączony z pinem „GND". W przypadku bieguna dodatniego, z uwagi na potrzebę zasilania napięciem 5 wolt, zastosowano pin „VIN". Komunikacja z ESP32 odbywa się poprzez środkowy pin przełącznika. Został on połączony z pinem „D5".

Urządzenie włączane oraz wyłączane przez przełącznik może być zasilane prądem stałym lub przemiennym. W drugim przypadku przy napięciu do 250 wolt może być zasilane natężeniem maksymalnie 10 amper. Gdy woltów jest dwukrotnie mniej, największa wartość amperażu to 15 jednostek. W przypadku prądu stałego znaczenie

ma tylko napięcie, które może wynosić maksymalnie 5 wolt.

Wykorzystanym urządzeniem do włączania bądź wyłączania jest dioda LED. Jest to element elektroluminescencyjny, który świeci pod wpływem doprowadzonego napięcia. Do pracy potrzebuje ona napięcia 3 wolt. Z tego względu została połączona z pinem „3V3". Jednak nie jest to połączenie bezpośrednie, gdyż kabel na swojej drodze przechodzi przez przełącznik. Druga nóżka diody połączona została z uziemieniem, to jest z pinem „GND". Przewód nie łączy się bezpośrednio, gdyż po drodze znajduje się rezystor o oporze 330Ω, który ma ograniczać prąd płynący przez diodę, tak aby nadmierna moc jej nie uszkodziła.

## 2.2 Serwer nadzorujący

Do wymiany danych pomiędzy stroną WWW a komputerem ESP32 zastosowano serwer pośredni. Dzięki temu nie ma potrzeby instalowania żadnego oprogramowania po stronie użytkownika.

Serwer został wykonany z wykorzystaniem języka Java w technologii Spring boot [12]. Jest to obecnie bardzo popularna technologia tworzenia aplikacji systemów internetowych.

Wraz ze startem serwera tworzone są wstępne obiekty, na których trzymane są wartości czujnika oraz stan przełącznika i połączenia. Pokazuje to kod na listingu 2.1.

```
Mikrokontroler mikrokontroler = new Mikrokontroler( "192.168.0.18",
    false);
mikrokontrolerRepository.save(mikrokontroler);
Czujnik czujnik = new Czujnik("192.168.0.18", "0");
czujnikRepository.save(czujnik);
Przelacznik przelacznik = new Przelacznik("192.168.0.18", false);
przelacznikRepository.save(przelacznik);
```

Listing 2.1: Kod odpwiedzialny za utworzenie obiektów startowych

Po stworzeniu obiektów serwer sprawdza wartość temperatury. W przypadku prawidłowego odczytu ustala połączenie z ESP32 jako stabilne oraz zapisuje odczyt. Kolejny krok to odczytanie wartości przełącznika. Po tych pojedynczych wstępnych czynnościach następuje ciągłe zbieranie informacji o aktualnym stanie połączenia oraz wartości temperatury. Pokazano to na listingu 2.2.

```
while (true) {
try {
        HttpClient client = HttpClient.newHttpClient();
        HttpRequest request = HttpRequest.newBuilder()
        .uri(URI.create("http://192.168.0.18/temperaturec"))
        .timeout(Duration.ofMillis(2000))
        .build();

        HttpResponse response = client.send(request,
        HttpResponse.BodyHandlers.ofString());

        String zmienna1=(String) response.body();
        double zmienna2= Double.parseDouble(zmienna1);
        if (Math.abs(zmienna2-Float.parseFloat(czujnik
        .getWartosc()))<20) {
                czujnik.setWartosc((String) response.body());
                czujnikRepository.save(czujnik);
        }

        mikrokontroler.setPolaczony(true);
        mikrokontrolerRepository.save(mikrokontroler);

        Thread.sleep(1000);
} catch (Exception e) {
        mikrokontroler.setPolaczony(false);
        mikrokontrolerRepository.save(mikrokontroler);
        Thread.sleep(1000);
}
}
```

Listing 2.2: Kod odpwiedzialny za ciągłe monitorowanie temperatury oraz stan połączenia

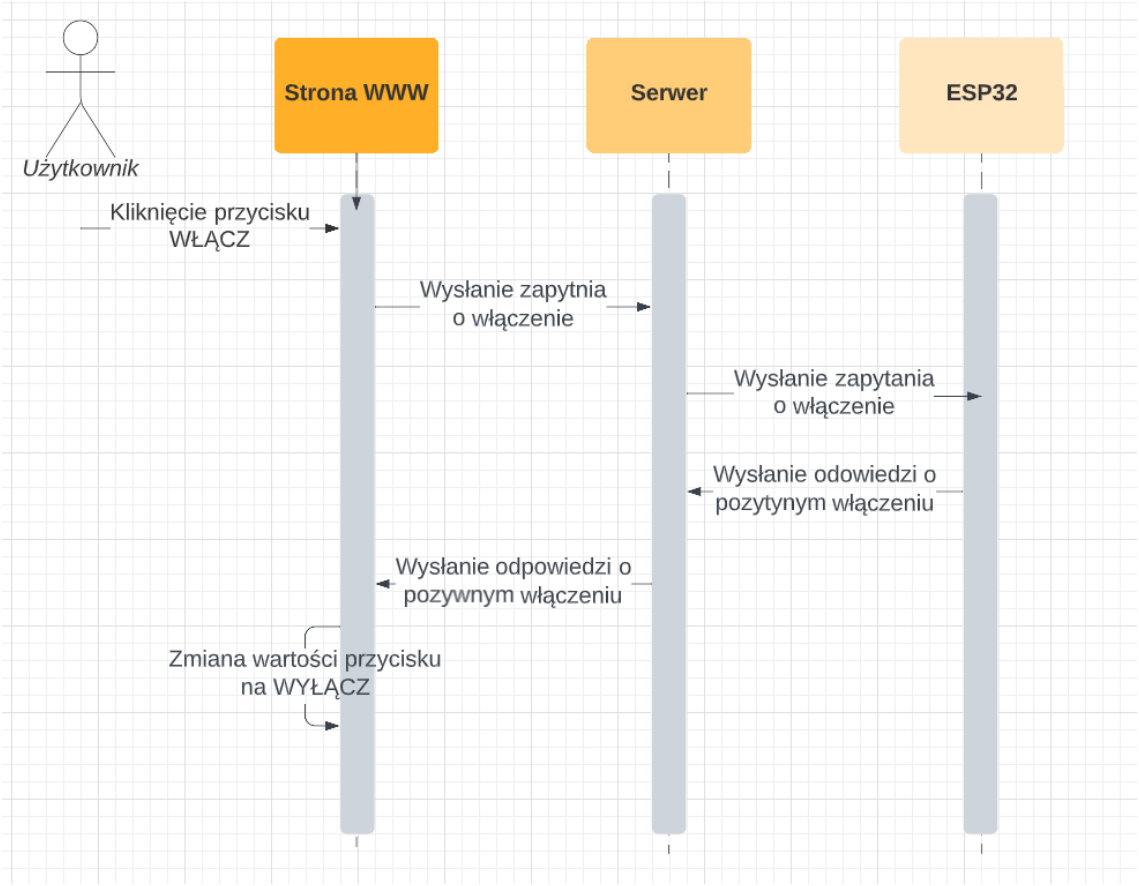
## 2.3 Schemat komunikacji

Komunikacja między stroną WWW, serwerem oraz ESP32 odbywa się poprzez protokół HTTP [13].

W celu pobrania informacji o temperaturze na interfejs użytkownika wysyłane jest zapytanie na serwer. Wartość odczytu jest pobierana z wcześniej zapisanych zasobów i wysyłana do strony WWW. Dzieje się tak, gdyż serwer po uruchomieniu sam co sekundę odpytuje ESP32 o wskazania czujnika temperatury. Pokazano to na

listingu 2.2.

Schemat komunikacji dla włączenia przełącznika jest bardziej złożony, dlatego pokazano go na rysunku 2.3.




Rysunek 2.3: Sposób wymiany danych między stroną WWW, serwerem oraz ESP32

## 2.4 Oprogramowanie ESP32

Współpraca ESP32 z serwerem opiera się na trzech funkcjach. Służą one do odczytu temperatury oraz odczytywania i modyfikowania stanu przełącznika [14].

Na potrzeby pobrania temperatury użyto kody pokazane na listingach 2.3 oraz 2.4.

```
server.on("/temperaturec", HTTP_GET, [](AsyncWebServerRequest
*request){
        request->send_P(200, "text/plain",
        readDSTemperatureC().c_str());
});
```

Listing 2.3: Nadrzędny kod odczytywania temperatury

```
String readDSTemperatureC() {
        sensors.requestTemperatures();
        float tempC = sensors.getTempCByIndex(0);
        return String(tempC);
}
```

Listing 2.4: Zasadnicze instrukcje odczytujące temperaturę

Interesujące jest to, że do komunikacji pomiędzy ESP32 a przełącznikiem nie potrzeba żadnej funkcji. Wystarczy tylko sprawdzenie, czy na połączeniu jest stan niski czy wysoki. Pokazuje kod na listingu 2.5.

```
server.on("/relayStatus", HTTP_GET, [] (AsyncWebServerRequest
*request) {
String inputMessage;
String inputParam;
int Status=1;
if (request->hasParam(PARAM_INPUT_1)) {
        inputMessage = request->getParam(PARAM_INPUT_1)->value();
        inputParam = PARAM_INPUT_1;
        if(RELAY_NO){
                Status=digitalRead(relayGPIOs[inputMessage.toInt()
                -1]);
        }
        else{
                Status=digitalRead(relayGPIOs[inputMessage.toInt()
                -1]);
        }
}
else {
        inputMessage = "No␣message␣sent";
        inputParam = "none";
}
if(Status==1)
        Status=0;
else
        Status=1;
request->send(200, "text/plain", String(Status));
});
```

Listing 2.5: Kod odczytujący stan przełącznika

Ostatnią funkcją, którą możemy wywołać na ESP32, jest ta odpowiedzialna za uruchomienie bądź wyłączenie przekaźnika. Obrazuje to listing 2.6.

```
server.on("/update", HTTP_GET, [] (AsyncWebServerRequest
*request) {
String inputMessage;
String inputParam;
String inputMessage2;
String inputParam2;
if (request->hasParam(PARAM_INPUT_1) & request->hasParam
(PARAM_INPUT_2)) {
        inputMessage = request->getParam(PARAM_INPUT_1)->value();
        inputParam = PARAM_INPUT_1;
        inputMessage2 = request->getParam(PARAM_INPUT_2)->value();
        inputParam2 = PARAM_INPUT_2;
        if(RELAY_NO){
                digitalWrite(relayGPIOs[inputMessage.toInt()-1],
                !inputMessage2.toInt());
        }
        else{
                digitalWrite(relayGPIOs[inputMessage.toInt()-1],
                inputMessage2.toInt());
        }
}
else {
        inputMessage = "No␣message␣sent";
        inputParam = "none";
}
request->send(200, "text/plain", "OK");
});
```

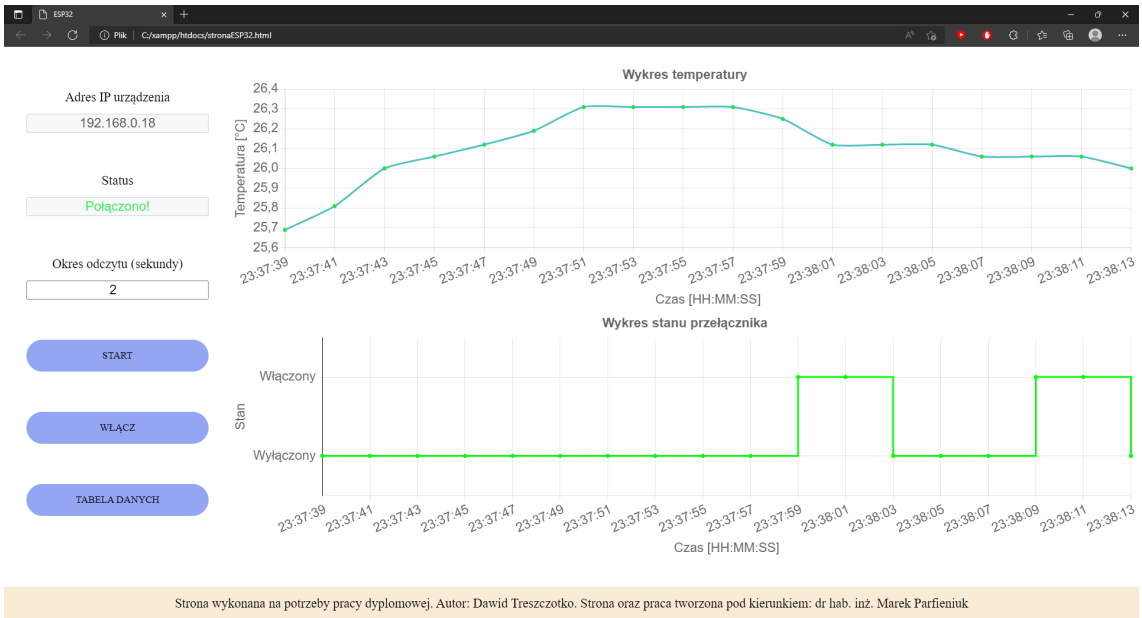
Listing 2.6: Fragment kodu zmieniającego stan przełącznika

# Rozdział 3

# Interfejs użytkownika i sposoby użycia

## 3.1 Ogólny wygląd głównego okna

Do obsługi systemu przez użytkownika został stworzony interfejs graficzny w formie strony WWW opierającej się na technologiach HTML5, JS i CSS [15]. Ogólny wygląd strony pokazano na rysunku 3.1. Informacje zgrupowane w dwie sekcje. Pierwsza udostępnia dane konfiguracyjne. Druga sekcja odpowiada za wyświetlanie wykresów obrazujących zebrane dane o temperaturze i stanie czujnika.




Rysunek 3.1: Aplikacja internetowa do obsługi systemu

## 3.2 Konfiguracja systemu


Rysunek 3.2: Panel konfiguracyjny interfejsu graficznego

Panel konfiguracyjny obejmuje pola pokazane na rysunku 3.2.

W polu „Adres IP urządzenia” znajduje się informacja o adresie IP, który został przypisany do komputera jednopłytkowego. Jest to jego adres w sieci.

Pole „Status” pokazuje aktualny stan połączenia ESP32 z routerem. Po uruchomieniu strony w przeglądarce najpierw wykonywane jest zapytanie do serwera o obecny status. Symbolizuje to napis „Trwa łączenie...”. Następnie zależnie od sytuacji pojawia się informacja o poprawnym połączeniu lub jego braku. Symbolizują to odpowiednio zielony napis "Połączono!"albo czerwony "Rozłączono!".

Pole „Okres odczytu (sekundy)” określa, z jaką częstotliwością strona ma zbierać dane. Określa się ją w sekundach. Domyślnie w okno wpisana jest wartość 2, jednak można ją zmienić na inną.

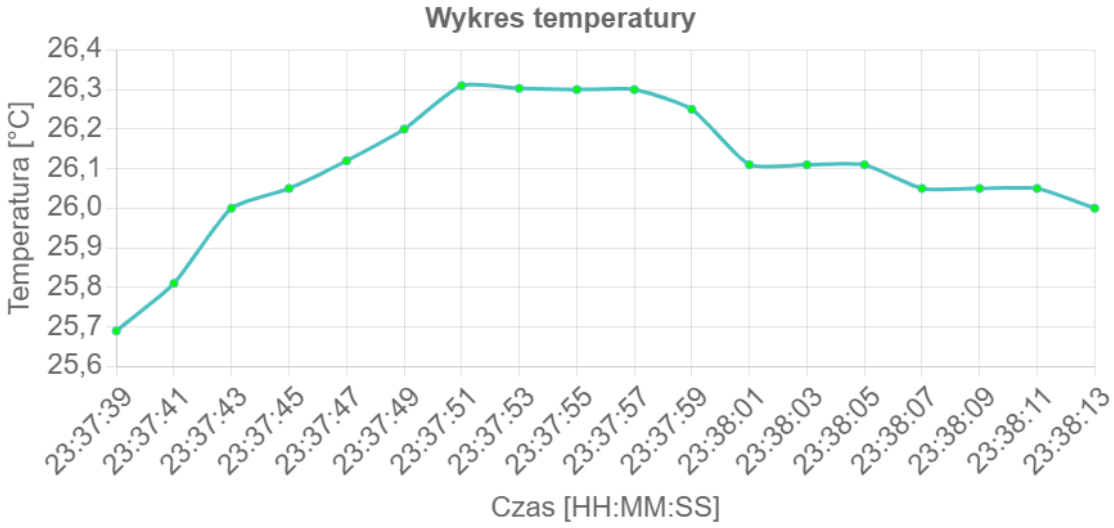
Przycisk „START” odpowiada za rozpoczęcie rejestrowania danych. Po jego kliknięciu zmieni się napis ze Start na Stop, a strona zgodnie ze wpisanym interwałem zbiera informacje z serwera.

Przycisk „WŁĄCZ” służy do włączenia przekaźnika. Po włączeniu etykiety, ta zmienia się na „WYŁĄCZ” oraz następuje odpowiednie przełączenie jego funkcjonalności.

Po kliknięciu przycisku „TABELA DANYCH” wyświetlana jest tabela pokazująca wartość temperatury i stan przełącznika wraz z czasem odczytu tych wartości.

## 3.3 Obrazowanie temperatury

Do obrazowania temperatury użyto wykresu opatrzonego tytułem „Wykres temperatury”. Wykres, wypełniony przykładowymi danymi, został przedstawiony na rysunku 3.3.




Rysunek 3.3: Wykres temperatury z opracowanej aplikacji

Aby rozpocząć generowanie wykresu należy najpierw uruchomić zbieranie danych, klikając przycisk „START”. Następnie zgodnie z wpisanym interwałem wykonuje się wielokrotnie skrypt Javascript [16] pokazany na listingu 3.1.

```
czas=new Date();
poziomaOs.push(czas.toLocaleTimeString());

const Http2 = new XMLHttpRequest();
const url2='http://localhost:8080/czujnik/192.168.0.18';
```

```
Http2.open("GET", url2);
Http2.send();

Http2.onloadend = (e) => {
        if(Http2.status==200)
        {
                stanTemp.push(Http2.responseText)
                wykresTemperatura.destroy();
                rysujTemperatura(stanTemp,poziomaOs);
        }
}
```

Listing 3.1: Kod cyklicznego pobierania temperatury

Do wygenerowania wykresu wykorzystano bibliotekę „chart.js” (Link: `https://www.chartjs.org/`). Fragment tworzący wykres pokazano na listingu 3.2.

```
wykresTemperatura = new Chart(ctx, {
        type: 'line',
        data: {
                labels:  pozioma,
                datasets: [{
                        label: 'temperatura',
                        data: lista,
                        borderColor: '#00FF00',
                        backgroundColor: '#00FF00',
                        fill: false,
                        borderColor: 'rgb(75, 192, 192)',
                        tension: 0.2
                }]
        },
        options: {
                plugins:{
                         tooltip: {
                                callbacks: {
                                        label: (item) =>
                                        `${item.dataset.label}:
                                        ${item.formattedValue} C`,
                                },
                        },
                        legend:{display: false},
                        title:{
                                display:true,
                                font: {size: 22,},
                                text: "Wykres temperatury"
```

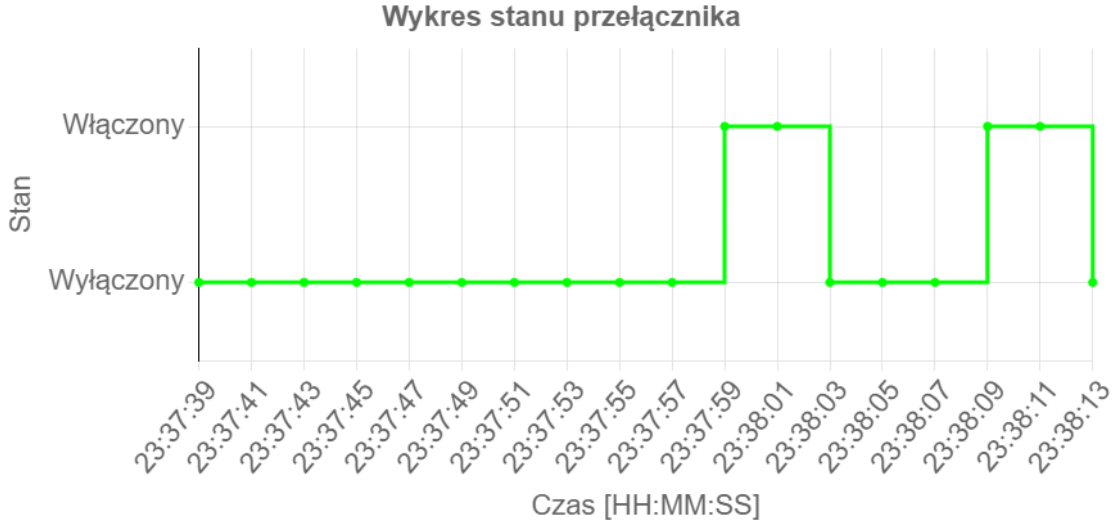
```
                        }
                },
        animation: {
                y: {
                        easing: "easeInOutCircs",
                        duration: document.getElementById("okres")
                        .value*100
                }
        },
        }
});
```

Listing 3.2: Kod tworzenia wykresu temperatury

Biblioteka umożliwia mnóstwo ustawień, którymi można ustawić najdrobniejsze szczegóły formatowania tekstu.

## 3.4 Obrazowanie stanu urządzenia

Wykres „Wykres stanu przełącznika" pokazuje, czy przełącznik jest aktualnie włączony czy wyłączony. Przykładowy przebieg został przedstawiony na rysunku 3.4.




Rysunek 3.4: Wykres temperatury z prawej sekcji serwisu internetowego

Zbieranie danych odbywa się za pomocą kodu pokazanego na listingu 3.3, który jest wykonywany cyklicznie, zgodnie z wartością w polu „okres odczytu (sekundy)".

Dane tak jak w przypadku temperatury są prezentowane poprzez bibliotekę „char.js", jednak tutaj typ wykresu jest określony jako „line".

```
czas=new Date();
poziomaOs.push(czas.toLocaleTimeString());

const Http = new XMLHttpRequest();
const url='http://localhost:8080/przelacznik/192.168.0.18';
Http.open("GET", url);
Http.send();

Http.onloadend = (e) => {
        if(Http.status==200)
        {
                if(Http.responseText=='true')
                {
                        stan='łąWczony';
                        stanOn_Off.push('łąWczony');
                }
                else
                {
                        stanOn_Off.push('łąWyczony');
                        stan='łąWyczony';
                }

                wykresPrzelacznik.destroy();
                rysujOn_Off(stanOn_Off, poziomaOs);
        }
```

Listing 3.3: Kod tworzenia wykresu stanu przełącznika

## 3.5 Okno eksportu danych

Przycisk „Tabela danych" powoduje wyświetlenie okna z tabelą, którą pokazano na rysunku 3.5. W kolumnach jest zapisana godzina pomiaru, wartość temperatury oraz stan przekaźnika. Ta forma reprezentacji informacji pozwala dla użytkownika na skopiowanie wartości temperatury oraz ich dalszą obróbkę.

Aby wyjść z tego sposobu reprezentacji danych i powrócić do strony domyślnej należy kliknąć przycisk „Zamknij". Znajduje się on na samym szczycie tabeli.

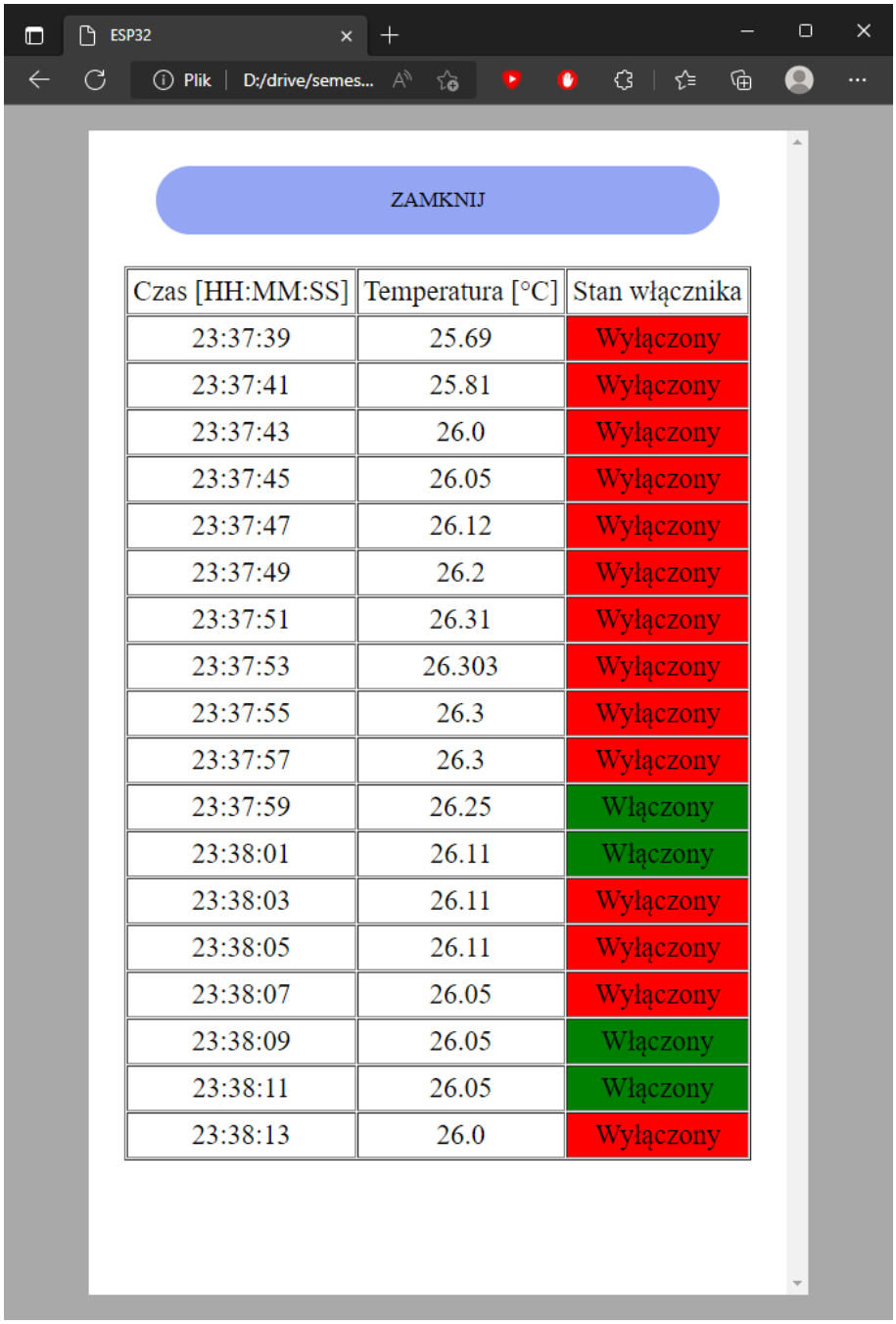
Kod odpowiedzialny za dodawanie kolejnych rekordów pokazano na listingu 3.4.

```
function dodajDaneDoTabeli(czas, temperatura, stan){
```

```
        var table = document.getElementById("tabela");
        var row = table.insertRow(i);
        var cell1 = row.insertCell(0);
        var cell2 = row.insertCell(1);
        var cell3 = row.insertCell(2);
        cell1.innerHTML = czas;
        cell2.innerHTML = temperatura;
        cell3.innerHTML = stan;
        if(stan=="Wylaczony")
        cell3.style.backgroundColor="red";
        if(stan=="Wlaczony")
        cell3.style.backgroundColor="green";

        i++;
}
```

Listing 3.4: Kod dodawania rekordów do tabeli




| Czas [HH:MM:SS] | Temperatura [°C] | Stan włącznika |
|---|---|---|
| 23:37:39 | 25.69 | Wyłączony |
| 23:37:41 | 25.81 | Wyłączony |
| 23:37:43 | 26.0 | Wyłączony |
| 23:37:45 | 26.05 | Wyłączony |
| 23:37:47 | 26.12 | Wyłączony |
| 23:37:49 | 26.2 | Wyłączony |
| 23:37:51 | 26.31 | Wyłączony |
| 23:37:53 | 26.303 | Wyłączony |
| 23:37:55 | 26.3 | Wyłączony |
| 23:37:57 | 26.3 | Wyłączony |
| 23:37:59 | 26.25 | Włączony |
| 23:38:01 | 26.11 | Włączony |
| 23:38:03 | 26.11 | Wyłączony |
| 23:38:05 | 26.11 | Wyłączony |
| 23:38:07 | 26.05 | Wyłączony |
| 23:38:09 | 26.05 | Włączony |
| 23:38:11 | 26.05 | Włączony |
| 23:38:13 | 26.0 | Wyłączony |

Rysunek 3.5: Tabela z danymi

# Podsumowanie

Zrealizowano wszystkie cele pracy. Praktycznym wynikiem jest aplikacja do sterowania systemem obejmującym komputer jednopłytkowy, czujnik i przekaźnik. Aby umożliwić użytkownikowi zdalny dostęp, opracowano serwis WWW z estetycznymi wykresami.

Praca wymagała poznania technologii sprzętowej i programowania komputerów jednoukładowych w technologi Arduino, której nie obejmował program studiów. Wysiłku wymagało opracowanie fizycznego układu oraz dobranie odpowiednich urządzeń. Pewnej pomysłowości wymagało też opracowanie interfejsu internetowego w oparciu o technologie spring boot.

Praca pozwoliła autorowi bliżej poznać zagadnienie "Internet rzeczy", które jest przyszłościową dziedziną informatyki.

# Bibliografia


[1] D. Ibrahim, *Microcontroller Based Temperature Monitoring and Control*. Newnes, 2002, ISBN: 978-0-7506-5556-9.

[2] T. K. Kimmo Karvinen, *Czujniki dla początkujących. Poznaj otaczający Cię świat za pomocą elektroniki, Arduino i Raspberry Pi*. Helion, 2015, ISBN: 978-83-283-0368-3.

[3] M. Long, „Best Single-Board Computers for Every Use and Budget in 2021," 2021.

[4] Espressif Systems, *ESP32WROOM32 Datasheet*, `https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf`, 2022.

[5] L. R. G. Rajesh Singh Anita Gehlot, *IoT based Projects: Realization with Raspberry Pi, NodeMCU and Arduino*. BPB Publications, 2020, ISBN: 9389328527.

[6] N. R. W. Michael Margolis Brian Jepson, *Arduino Cookbook: Recipes to Begin, Expand, and Enhance Your Projects, 3rd Edition*. Helion S.A., 2021, ISBN: 978-83-283-7161-3.

[7] Espressif, *Getting Started About Arduino ESP32*, `https://docs.espressif.com/projects/arduino-esp32/en/latest/getting_started.html`, 2022.

[8] Espressif, *Arduino core for the ESP32*, `https://github.com/espressif/arduino-esp32/`, 2022.

[9] Maxim Integrated Products, Inc, „DS18B20 - Programmable Resolution 1-Wire Digital Thermometer," 2019.

[10] M. Cina, *Das 1-Wire-Praxisbuch*. Elektor; Neuauflage, 2020, ISBN: 978-3895763502.

[11] W. Andrews, *Zrób to sam z Arduino Zaawansowane projekty dla doświadczonych twórców*. Wydawnictwo Naukowe PWN, 2017, ISBN: 9788301196998.

[12] C. Walls, *Spring Boot in Action*. Manning Publications, 2015, ISBN: 9781617292545.

[13] M. S. David Gourley Brian Totty, *HTTP: The Definitive Guide. The Definitive Guide*. O'Reilly Media, Inc., 2002, ISBN: 9781565925090.

[14] S. Spanulescu, *ESP32 Programming for the Internet of Things - Second Edition: HTML, JavaScript, MQTT and WebSockets Solutions*. Sever Spanulescu, 2020, ISBN: 9781005298302.

[15] J. Duckett, *HTML i CSS. Zaprojektuj i zbuduj witrynę WWW. Podręcznik Front-End Developera*. Helion, 2017, ISBN: 978-83-283-4481-5.

[16] S. Tomasz, *JavaScript. Tworzenie nowoczesnych aplikacji webowych*. Helion, 2020, ISBN: 978-83-283-5637-5.