POLITECHNIKA ŚLĄSKA

WYDZIAŁ AUTOMATYKI, ELEKTRONIKI I INFORMATYKI

KIERUNEK: TELEINFORMATYKA

Praca dyplomowa inżynierska

System inteligentnego sterowania ogrzewaniem domu

autor: Marcin Konopka

kierujący pracą: dr inż. Grzegorz Baron

konsultant: mgr inż. Jarosław Paduch

Gliwice, luty 2022

Spis treści

Streszczenie

Celem pracy jest stworzenie systemu sterującego ogrzewaniem domu poprzez zamykanie i otwieranie zaworów termostatycznych grzejników pokojowych oraz sterowanie pracą pieca (włącz/wyłącz).

W ramach pracy opracowano koncepcję systemu (wydzielono typy urządzeń, które tworzyć mają system, wybrano formę komunikacji między nimi czy w końcu wypracowano mechanizmy, według których działa system, oraz zasady, według których podejmowane są w nim decyzje), wybrano technologie (języki programowania, elementy elektroniczne, narzędzia programistyczne itd.) umożliwiające realizację systemu zgodnie z koncepcją, zaprojektowano płytkę jednego z urządzeń stanowiących system (urządzenia pokojowego) oraz zrealizowano oprogramowanie dla wszystkich typów urządzeń stanowiących ten system, jak również oprogramowanie umożliwiające zarządzanie systemem z poziomu graficznego interfejsu użytkownika.

W pracy opisano proces wypracowania koncepcji systemu, uzasadniono merytorycznie wybór konkretnych technologii, opisano proces projektowania płytki drukowanej urządzenia pokojowego, dokonano opisu oprogramowania poszczególnych urządzeń stanowiących system oraz oprogramowania interfejsu użytkownika, opisano proces uruchomienia tych urządzeń i/lub ich oprogramowania, napisano jak zarządzać systemem i skonfigurować urządzenia stanowiące system wraz z ich oprogramowaniem, dokonano weryfikacji i walidacji poprawności elementów systemu oraz podsumowano wyniki pracy i nakreślono kierunki dalszego rozwoju systemu.

Slowa kluczowe: sterowanie ogrzewaniem, automatyzacja

Rozdział 1

Wstęp

1.1 Wprowadzenie w problem

Wraz z rosnącymi kosztami energii oraz zwiększającym się w społeczeństwie przekonaniem o konieczności ochrony środowiska, powodowanym zaniepokojeniem o klimat oraz kwestie zdrowotne wynikłe z zanieczyszczeń powietrza, rozsądnym zdaje się skorzystanie z dobrodziejstw techniki w celu minimalizacji negatywnych skutków zjawisk związanych z tymi zagadnieniami.

1.2 Osadzenie problemu w dziedzinie

Związane z dziedziną Teleinformatyki rozwiązania techniczne tj. systemy mikroprocesorowe, metody i protokoły komunikacji czy sieci sensorowe umożliwiają automatyzację i inteligentne sterowanie ogrzewaniem domu w celu optymalizacji zużycia zasobów i minimalizacji negatywnych zjawisk wspomnianych w poprzednim akapicie.

1.3 Cel pracy

Celem pracy jest stworzenie systemu sterującego ogrzewaniem domu poprzez zamykanie i otwieranie zaworów termostatycznych grzejników pokojowych oraz sterowanie pracą pieca (włącz/wyłącz). Decyzja o zamykaniu czy otwieraniu za-

woru, ma być podejmowana uwzględniając nie tylko temperaturę w danym pokoju, ale również stan systemu.

1.4 Zakres pracy

W ramach projektu zostały zrealizowane:

- Koncepcja działania oraz architektury systemu
- Projekt urządzenia pokojowego (schemat ideowy, projekt płytki drukowanej)
- Oprogramowanie mikrokontrolera urządzenia pokojowego
- Oprogramowanie mikrokontrolera urządzenia sterującego pracą pieca (uruchamiane na urządzeniu zaprojektowanym i zbudowanym niezależnie od niniejszej pracy)
- Oprogramowanie urządzenia centralnego (oprogramowanie sterujące całością systemu uruchamiane na komputerze jednopłytkowym)
- Interfejs użytkownika umożliwiający zarządzanie systemem

1.5 Struktura pracy

We wstępie postarano się przybliżyć motywacje stojące za stworzeniem projektu, wprowadzić czytelnika w problemy poruszane w pracy, osadzono problem w dziedzinie oraz określono cel i zakres prac.

W rozdziale *Analiza tematu* sformułowano główny problem poruszany w pracy, osadzono go w kontekście aktualnego stanu wiedzy, dokonano wstępnej analizy zagadnień, których znajomość jest konieczna w realizacji projektu oraz dokonano przeglądu wybranych rozwiązań rynkowych związanych z tematyką sterowania zaworami grzejników pokojowych, oraz pracą pieca.

W kolejnym rozdziale (*Specyfikacja projektu*) wypracowano koncepcję systemu - wydzielono typy urządzeń stanowiących system, wybrano formę, w której się

one komunikują oraz sprecyzowano, jak będzie działał system. Dokonano również oceny realizowalności i kosztu wykonania projektu.

Rozdział *Koncepcja projektowa* opisuje proces wyboru elementów elektronicznych (urządzenie pokojowe), języków programowania, narzędzi użytych do realizacji projektu oraz protokołu wymiany informacji w systemie. Zawiera merytoryczne uzasadnienie konkretnych wyborów.

Kolejny rozdział, zatytułowany *Projekt urządzenia pokojowego*, zawiera opis projektu urządzenia pokojowego, przedstawia schemat ideowy tego urządzenia wraz z opisem konkretnych połączeń pomiędzy poszczególnymi układami, z których jest zbudowane. Przedstawiono w nim zaprojektowaną płytkę drukowaną urządzenia pokojowego oraz zbudowane prototypy.

W rozdziale *Oprogramowanie* opisano istotne zmienne, funkcje, procedury oraz metody oprogramowania stworzonego w ramach pracy. W rozdziale tym umieszczono również dokumentację interfejsu programistycznego (tzw. API) oprogramowania sterującego całością systemu.

Rozdziały *Uruchomienie* oraz *Instrukcja obsługi* zawierają instrukcje jak prawidłowo uruchomić system, jak przeprowadzić jego konfigurację (w tym uruchomienie i konfigurację poszczególnych urządzeń stanowiących system) oraz jak zarządzać systemem.

Rozdział *Weryfikacja i walidacja* opisuje metodykę testowania poprawności działania systemu i poszczególnych jego części oraz przedstawia wyniki tych testów.

Ostatni rozdział zawiera podsumowanie wyników pracy, wyciągnięte przez autora wnioski oraz nakreśla kierunek dalszych prac nad rozwojem systemu.

Rozdział 2

Analiza tematu

2.1 Sformułowanie problemu

Sterowanie ogrzewaniem domu jest szerokim zagadnieniem, niniejsza praca skupi się na problemie utrzymywania w pomieszczeniach (strefach) temperatury zgodnie z harmonogramem ustawionym przez użytkownika poprzez skoordynowane ze sobą zamykanie/otwieranie zaworów termostatycznych tradycyjnych grzejników pokojowych w tych pomieszczeniach oraz sterowanie pracą pieca.

2.2 Problem w kontekście aktualnego stanu wiedzy

System realizujący zadania związane z tym problemem można zrealizować na różne sposoby, czy to jako system rozproszony, gdzie realizacja tych zadań jest podzielona pomiędzy różne komunikujące się ze sobą urządzenia stanowiące system, lub też w formie, w której za całość zadań odpowiada de facto jedno urządzenie, wyposażone w wiele czujników temperatury - znajdujących się w każdym pomieszczeniu, przyłączonych do niego np. magistralą OneWire - oraz sterujące zaworem w każdym pomieszczeniu i pracą pieca. Tworzony w ramach tej pracy system będzie systemem rozproszonym.

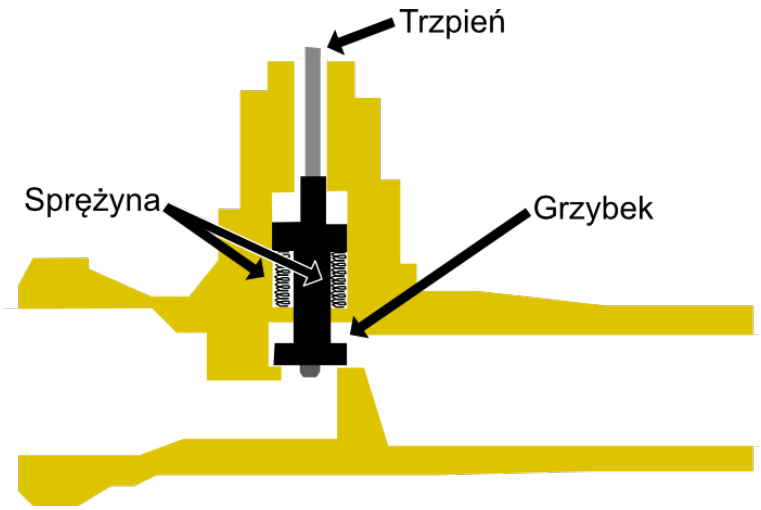
Aktualny stan techniki pozwala na realizację takiego systemu na wiele sposobów, chociażby z uwagi na istnienie wielu metod komunikacji (zarówno prze-

wodowej - np. OneWire, Ethernet - jak i bezprzewodowej - np. WiFi, ZigBee, Bluetooth - występujących często w różnych wersjach, topologiach itd.), licznych sposobów (protokołów) wymiany informacji (np. MQTT, HTTP, WebSocket), łączących obydwie ze wspomnianych wyżej, całych systemów tworzonych z myślą o automatyzacji domu (np. KNX) czy w końcu różnorodność i szeroką dostępność układów cyfrowych tj. mikrokontrolery, czujniki temperatury itd..

Języki programowania wysokiego poziomu (np. C++, Python, JavaScript) oraz dostępne narzędzia programistyczne, a w szczególności duża baza bibliotek dla tych języków umożliwiają szybką implementację algorytmów i procedur, dzięki którym możliwa jest realizacja tych zadań.

2.3 Zamykanie i otwieranie zaworu termostatycznego

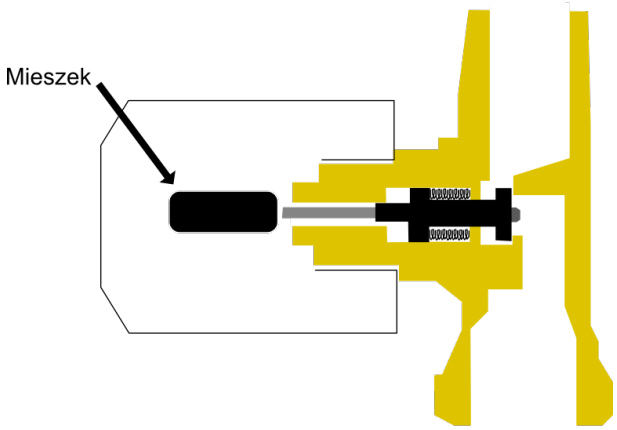
Głównymi elementami zaworu grzejnika termostatycznego są korpus, trzpień, grzybek i sprężyna [22] (patrz rys. 2.1). W celu zamknięcia zaworu konieczne jest wepchnięcie trzpienia, przezwyciężając siłę sprężyny. W zależności od tego, jak duża siła jest przyłożona do trzpienia, zawór jest otwarty mniej (większa siła) lub bardziej (mniejsza siła).



Rysunek 2.1: Poglądowy przekrój poprzeczny zaworu termostatycznego

Głowica termostatyczna

Głównym elementem głowicy termostatycznej jest sprężysty mieszek wypełniony cieczą o dużej rozszerzalności cieplnej (patrz rys. 2.2). Wzrost temperatury powoduje rozszerzanie się mieszka, co z kolei skutkuje wpychaniem trzpienia zaworu termostatycznego (tym samym przymykaniem lub zamykaniem zaworu). Gdy temperatura spada, mieszek kurczy się, a trzpień jest wypychany przez sprężynę znajdującą się w zaworze [19]. Zastosowanie głowicy termostatycznej nie umożliwia elektronicznego sterowania zaworem.

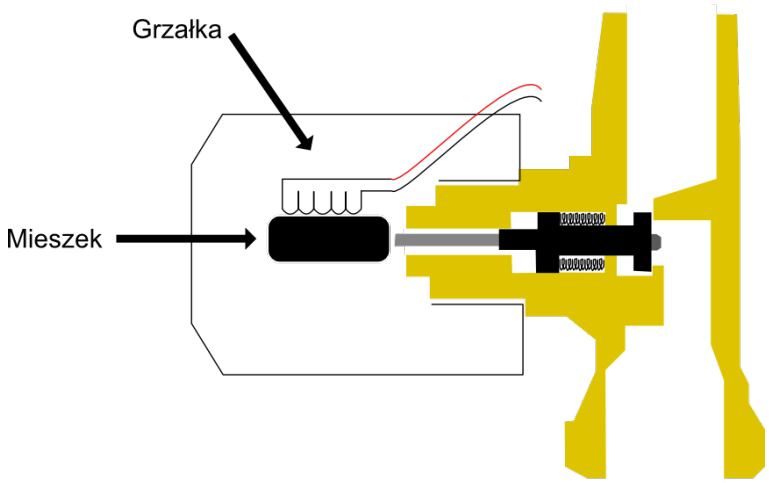


Rysunek 2.2: Głowica termostatyczna - rysunek poglądowy

Siłownik termoelektryczny

Zasada działania siłownika termoelektrycznego jest bardzo podobna jak w przypadku głowicy termostatycznej. Różnica polega na tym, że poza mieszkiem wewnątrz urządzenia znajduje się grzałka, która nagrzewa lub nie mieszek. W przypadku siłownika normalnie otwartego, gdy przez grzałkę zaczyna płynąć prąd i wydziela ona ciepło, zawór jest zamykany (rys. 2.3).

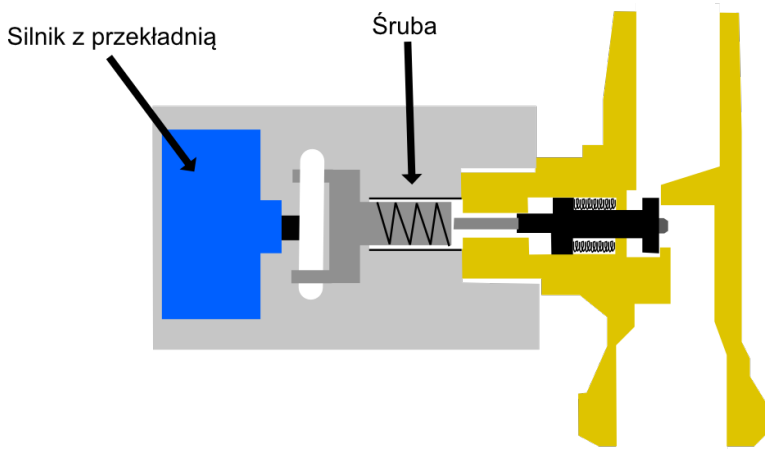
Siłownik termoeletryczny umożliwia elektroniczne sterowanie zaworem, np. poprzez sterowanie przekaźnikiem.



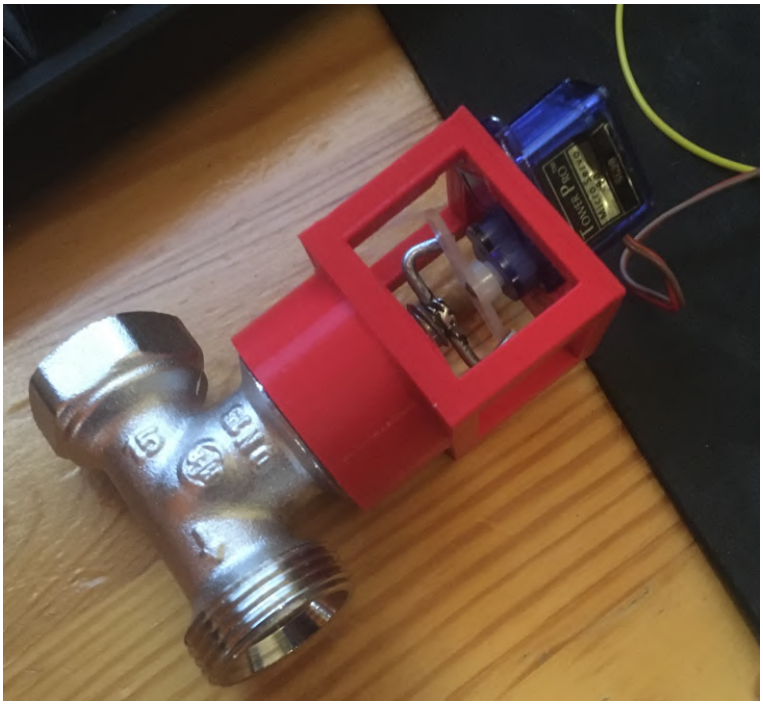
Rysunek 2.3: Siłownik termoelektryczny - rysunek poglądowy

Mechanizm z silnikiem z przekładnią

Innym rozwiązaniem jest umieszczenie na zaworze termostatycznym mechanizmu, którego głównymi elementami są śruba i silnik z przekładnią (rys. 2.4, rys. 2.5). Silnik dokręca śrubę, która z kolei wpycha trzpień, powodując zamknięcie zaworu. Gdy śruba jest wykręcana, znajdująca się w zaworze sprężyna wypycha trzpień, powodując tym samym otwarcie zaworu. Takim mechanizmem można sterować elektronicznie, np. sterując mostkiem H. Tworzony w ramach tej pracy system, będzie sterował zaworami właśnie w oparciu o to rozwiązanie.



Rysunek 2.4: Mechanizm z silnikiem z przekładnią - rysunek poglądowy



Rysunek 2.5: Mechanizm z silnikiem z przekładnią.

2.4 Rozwiązania rynkowe

Na rynku istnieje wiele rozwiązań, o różnym poziomie zaawansowanie, które umożliwiają zarówno sterowanie pracą pieca, jak i tych instalowanych na grzejnikach pokojowych w celu utrzymania w pomieszczeniach parametrów zadanych przez użytkownika. Poniżej dokonano przeglądu wybranych z nich.

Rozwiązanie marki Auraton opiera się na połączonych ze sobą drogą bezprzewodową (866MHz) głowicy instalowanej na grzejniku pokojowym oraz regulatorze, na którym użytkownik ustawia pożądane parametry. System uczy się, jak szybko zmienia się temperatura w pokoju, na podstawie czego przewiduje, jak mocno powinien być otwarty zawór grzejnika, aby utrzymać w pomieszczeniu odpowiednią temperaturę. System reguluje temperaturę lokalnie, w jednym pomieszczeniu, nie uwzględnia stanu ani nie steruje piecem [16, 2].

Głowice Fritz od AVM umożliwiają utrzymywanie w pomieszczeniu temperatury zgodnie z parametrami ustawionymi przez użytkownika (harmonogram lub szablony automatyzacji związane z innymi urządzeniami marki Fritz) poprzez regulację otwarcia zaworu grzejnika pokojowego. Komunikacja bezprzewodowa odbywa się przy pomocy standardu DECT [18].

Nest 3 jest inteligentnym termostatem sterującym ogrzewaniem domu. Nest posiada bardzo zaawansowane funkcje takie jak uczenie się cyklu życia użytkownika i odpowiednie dostosowanie temperatury do jego preferencji oraz szerokie możliwości automatyzacji w połączeniu z innymi inteligentnymi urządzeniami w domu. Możliwe jest zarówno sterowanie jedną strefą grzewczą, jak i wieloma (poprzez rozdzielnię ogrzewania) [20].

Istniejące na rynku rozwiązania oferują zazwyczaj sterowanie albo piecem, albo zaworami grzejników pokojowych. Skoordynowane sterowanie obydwoma wymaga zatem zazwyczaj dodatkowych systemów automatyzacji domu tj. np. Node-RED czy Home Assistant (w celu połączenia funkcjonalności jednych z drugimi). Tworzony w ramach tej pracy system ma natomiast realizować obydwie te funkcje, a skoordynowane sterowanie obydwoma jest główną ideą, która będzie determinować proces tworzenia i projektowania oprogramowania oraz urządzeń stanowiących system.

Rozdział 3

Specyfikacja projektu

3.1 Architektura

Pierwszym problemem z którym należy się zmierzyć, jest opracowanie architektury systemu, czyli ustalenie jakie urządzenia będą konieczne do jego działania oraz jakie funkcje będą one pełnić.

Przetwarzanie danych w sposób rozproszony jest problematyczne, a zatem preferowaną formą jest, aby system podejmował decyzje centralnie, w punkcie, który ma wiedzę o stanie wszystkich urządzeń tworzących system. Oczywistym jest, że w każdym pomieszczeniu musi znajdować się również urządzenie (lub urządzenia) odpowiedzialne za pomiar temperatury i zamykanie oraz otwieranie zaworów grzejników pokojowych. Musi istnieć również urządzenie, które steruje piecem.

Ostatecznie najrozsądniejszym zdaje się zatem wydzielenie trzech typów urządzeń, które będą tworzyły całość systemu:

- Urządzenie centralne - hub zbierający dane pomiarowe ze wszystkich pomieszczeń, podejmujący decyzję o włączeniu lub wyłączeniu pieca, otwarciu lub zamknięciu zaworu grzejnika w danym pokoju oraz stanowiący interfejs pomiędzy systemem a użytkownikiem.

- Urządzenie pokojowe - urządzenie znajdujące się w każdym pomieszczeniu, dokonujące pomiarów temperatury oraz zamykające i otwierające zawór grzejnika pokojowego.

- Urządzenie sterujące piecem - urządzenie włączające i wyłączające piec.

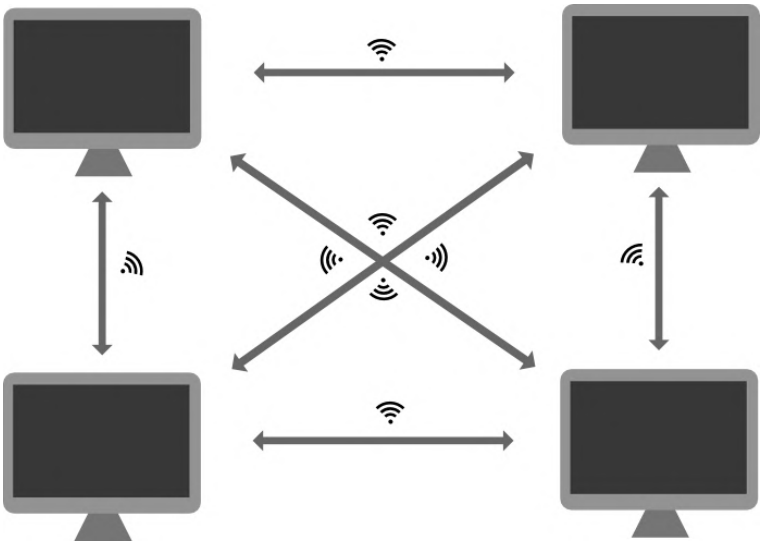
3.2 Komunikacja

Kolejnym istotnym zagadnieniem z punktu widzenia koncepcji systemu jest komunikacja pomiędzy urządzaniami. Z uwagi na to, że tworzenie infrastruktury sieciowej w przypadku metod komunikacji przewodowej jest trudniejsze i wymaga znacznie większych nakładów pracy niż infrastruktura sieci bezprzewodowej, preferowaną formą komunikacji będzie właśnie ta bezprzewodowa. W analizie wzięto pod uwagę niezwykle popularne w innych rozwiązaniach IoT (internetu rzeczy) standardy WiFi, ZigBee i Bluetooth.

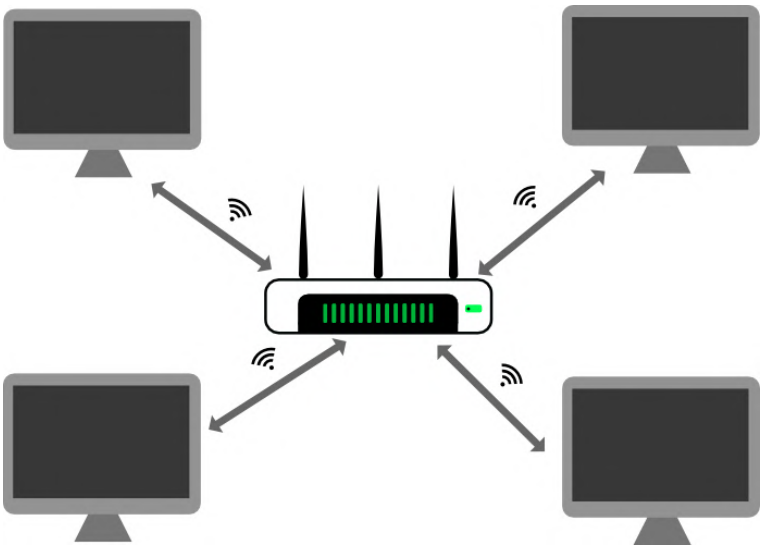
WiFi(IEEE 802.11) to zestaw standardów transmisji bezprzewodowej w pasmach 2.4GHz oraz 5GHz opracowany w celu tworzenia bezprzewodowych sieci lokalnych. Standardy w nim zawarte opisują niższe warstwy modelu sieciowego ISO OSI RM (model odniesienia łączenia systemów otwartych) - warstwę fizyczną i podwarstwę MAC (sterowania dostępem do medium transmisyjnego) [15, 25]. Ponieważ IEEE 802.11 może dzielić z IEEE 802.3 (Ethernet) protokoły i standardy wyższych warstw tego modelu, komunikacja pomiędzy urządzeniami przyłączonymi do sieci poprzez WiFi i Ethernet nie stanowi żadnego problemu, gdyż mogą one stanowić jednolitą sieć lokalną.

Podstawowe tryby pracy sieci WiFi (topologia sieci) [23]:

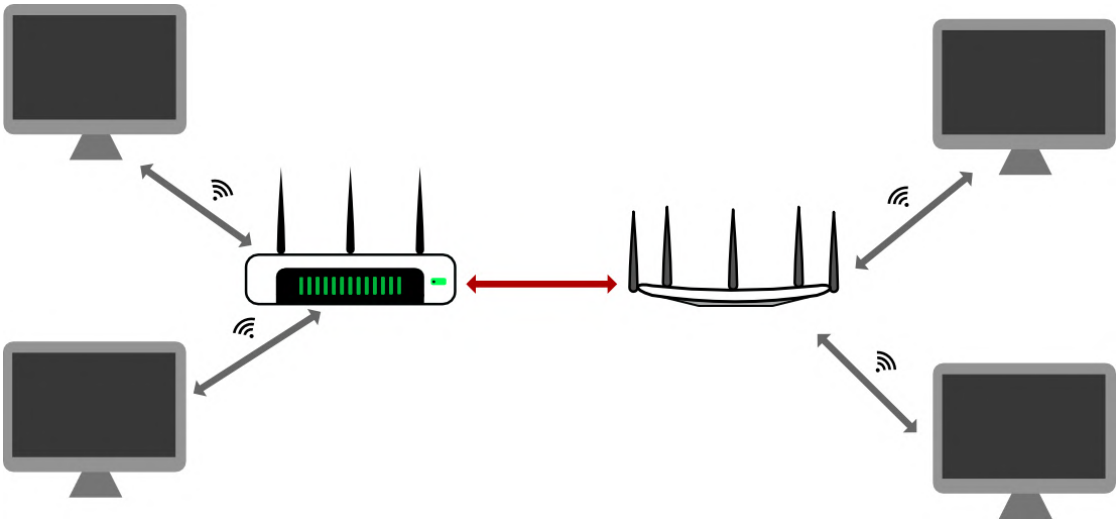
- Tryb ad-hoc (bez infrastruktury, IBSS), w którym komunikacja pomiędzy urządzeniami nie wymaga pośrednictwa jakiejkolwiek infrastruktury. Urządzenia w sieci komunikują się bezpośrednio między sobą (rys. 3.1).
- Tryb z infrastrukturą (BSS), w którym wymagany jest przynajmniej jeden punkt dostępowy (AP). Wszystkie ramki danych w sieci przekazywane są za pośrednictwem punktu dostępowego (rys. 3.2).
- Tryb rozszerzony (ESS), w którym kilka punktów dostępowych jest ze sobą połączonych drogą przewodową lub bezprzewodową (rys. 3.3).



Rysunek 3.1: Topologia sieci WiFi w trybie ad-hoc.



Rysunek 3.2: Topologia sieci WiFi w trybie z infrastrukturą.



Rysunek 3.3: Topologia sieci WiFi w trybie rozszerzonym.

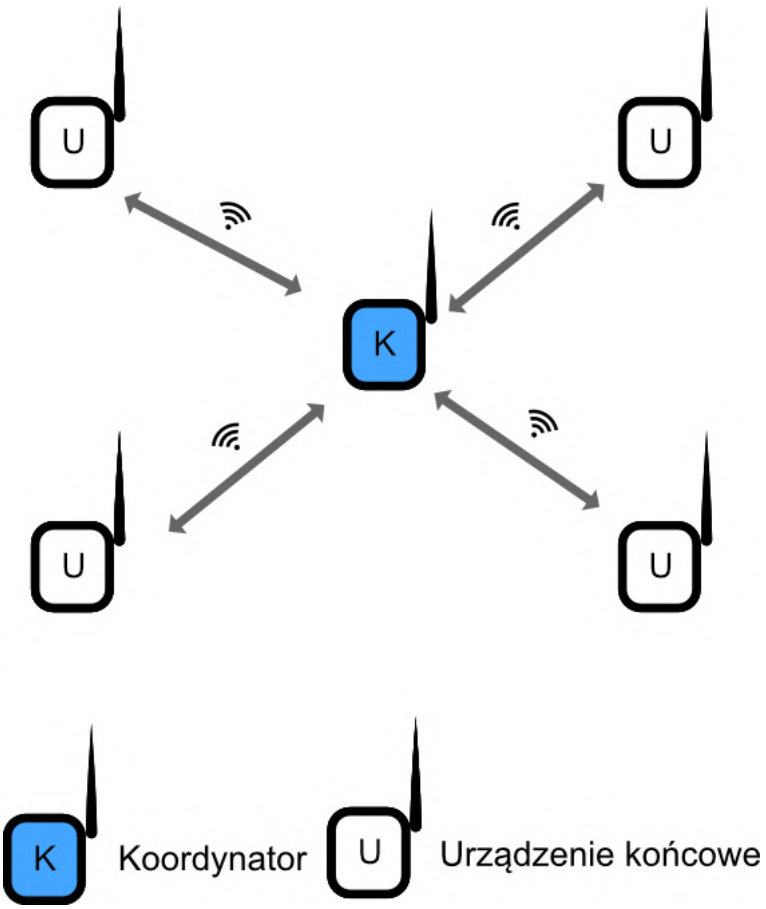
WiFi cechuje się dużą przepustowością, zasięgiem oraz poborem mocy (zarówno w trybie uśpienia i pracy) [14].

ESP-WIFI-MESH jest protokołem sieciowym nabudowanym na protokole WiFi, umożliwia on tworzenie bezprzewodowych sieci lokalnych z urządzeń rozsianych po dużym obszarze. Sieć jest zorganizowana w topologii siatki (kraty) - wszystkie urządzenia mogą komunikować się bezpośrednio między sobą jak i przekazywać między sobą ramki danych w ceu dostarczenia ich do adresata. Możliwa jest również organizacja sieci w topologii drzewa [8]. ESP-WIFI-MESH jest protokołem nieustandaryzowanym.

ZigBee to specyfikacja protokołów umożliwiających transmisję danych w sieciach bezprzewodowych niskiej przepływności (do 250kbps), działających w topologiach gwiazdy, drzewa lub siatki, na częstotliwościach 2.4GHz (globalnie) oraz w zależności od regionu świata 868 lub 915 MHz. Niższe warstwy specyfikacji (warstwa fizyczna i MAC) oparte są na standardzie IEEE 802.5.14 (Personalne sieci bezprzewodowe o niskiej przepływności) [26].

W sieciach ZigBee występują trzy typy urządzeń [26]:

- koordynator - w każdej sieci znajduje się tylko jeden koordynator. Jest węzłem początkowym do którego przyłączają się inne urządzenia w sieci.
- router - przekazuje (trasuje) pakiety pomiędzy urządzeniam
- urządzenie końcowe

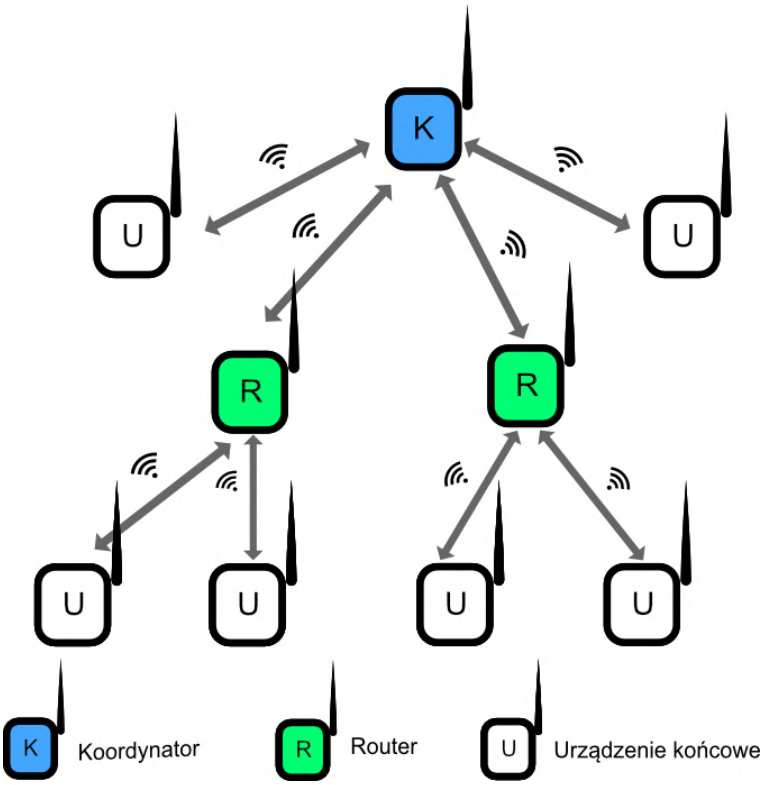


Rysunek 3.4: Topologia gwiazdy w sieci ZigBee.

Topologie sieci ZigBee [7]:

- Topologia gwiazdy - każde urządzenie końcowe komunikuje się bezpośrednio z koordynatorem. Cała komunikacja w sieci odbywa się za pośrednictwem koordynatora (rys. 3.4).

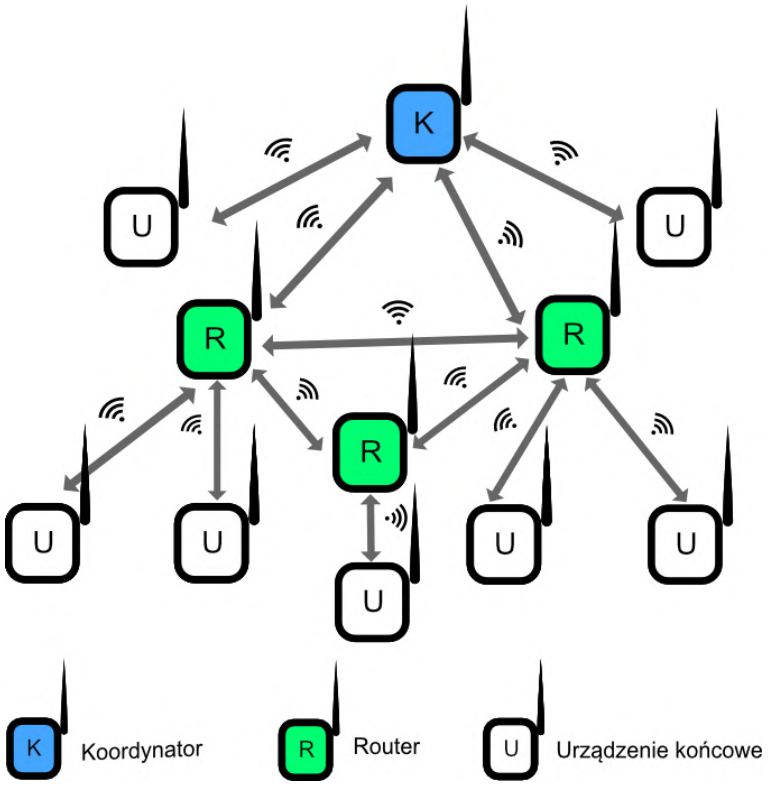
- Topologia drzewa - w tej topologii urządzenia końcowe (zwane dziećmi) mogą komunikować się tylko ze swoim rodzicem (router lub koordynator), co w przypadku problemów rodzica powoduje, że tracą one dostęp do sieci (rys. 3.5).



Rysunek 3.5: Topologia drzewa w sieci ZigBee.

- Topologia siatki - wszystkie urządzenia w sieci mogą komunikować się bezpośrednio między sobą lub poprzez pośredników. W przypadku gdy transmisja jedną z dróg transmisyjnych zawiedzie, węzeł jest zdolny do wykonania jej inną trasą. To najbardziej elastyczna topologią sieci ZigBee. Umożliwia bezproblemowe dodawanie i usuwanie urządzeń z sieci (rys. 3.6).

ZigBee charakteryzuje się znacznie niższymi od WiFi poborem energii oraz przepustowością jak również słabszym zasięgiem [14, 26], należy mieć jednak na uwadze, że możliwość przekazywania (trasowania) pakietów znacząco powiększa potencjalny zasięg sieci.



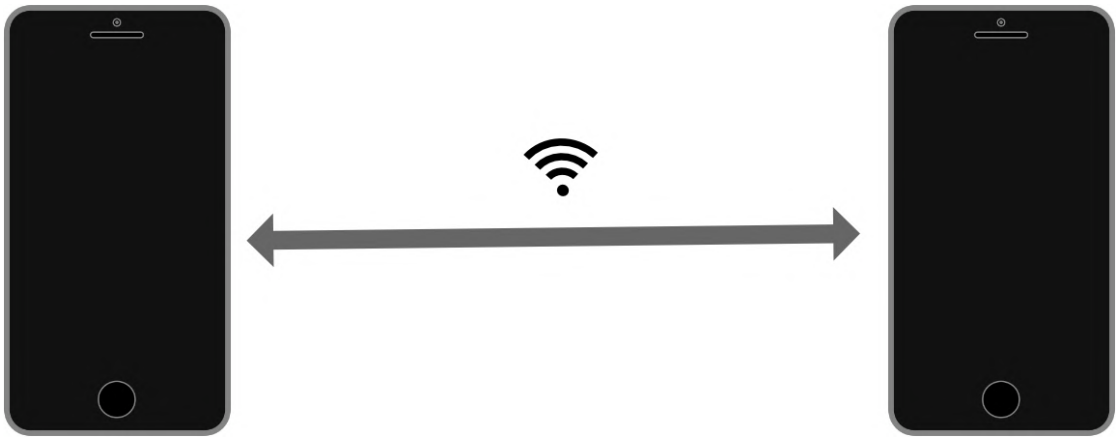
Rysunek 3.6: Topologia siatki w sieci ZigBee.

Bluetooth to standard bezprzewodowej transmisji krótkiego zasięgu stworzony głównie w celu transmisji pomiędzy urządzeniami mobilnymi oraz tworzenia sieci bezprzewodowych o zasięgu osobistym. Bluetooth operuje w zakresie częstotliwości od 2.402GHz do 2.48 GHz [24]. Najistotniejszym z punktu widzenia zastosowań IoT jest aktualnie wariant Bluetooth Low Energy (BLE).

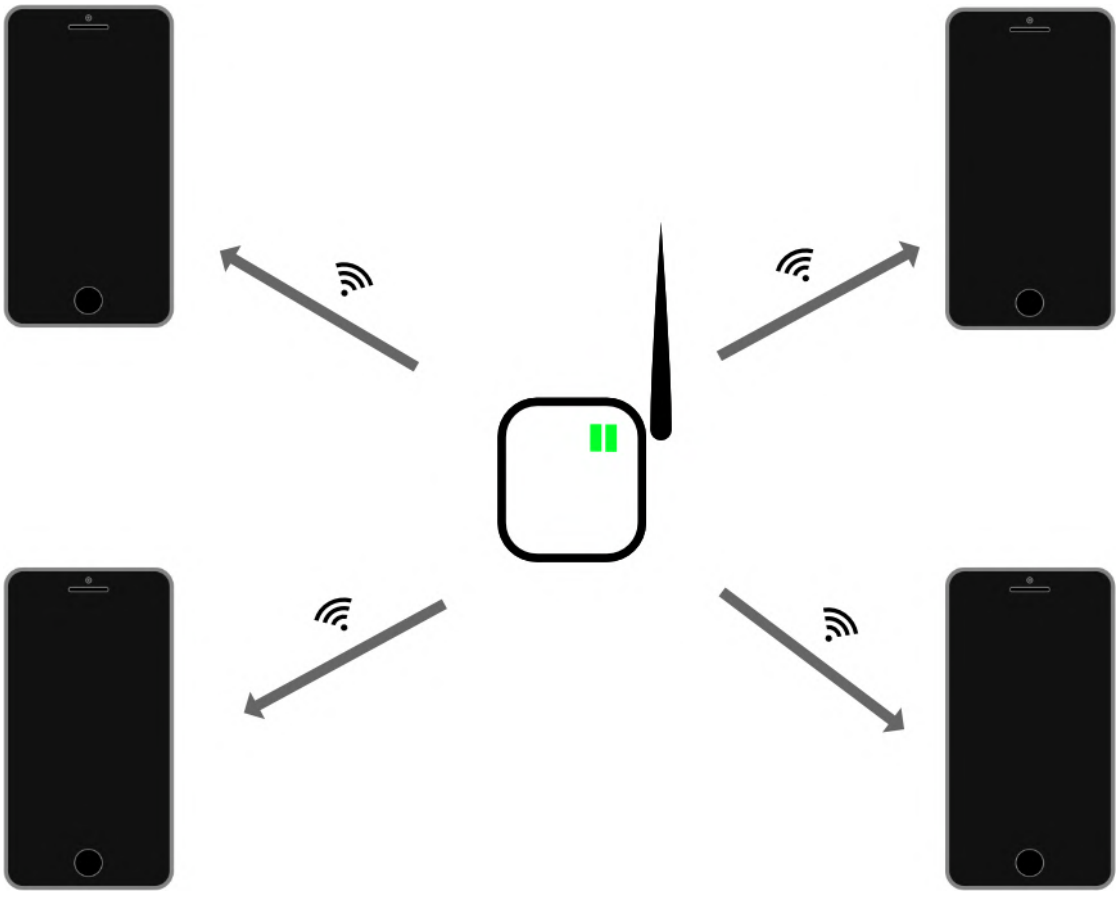
Topologie Bluetooth [3]:

- Punkt-punkt - dwa urządzenia łączą się bezpośrednio między sobą (rys. 3.7).

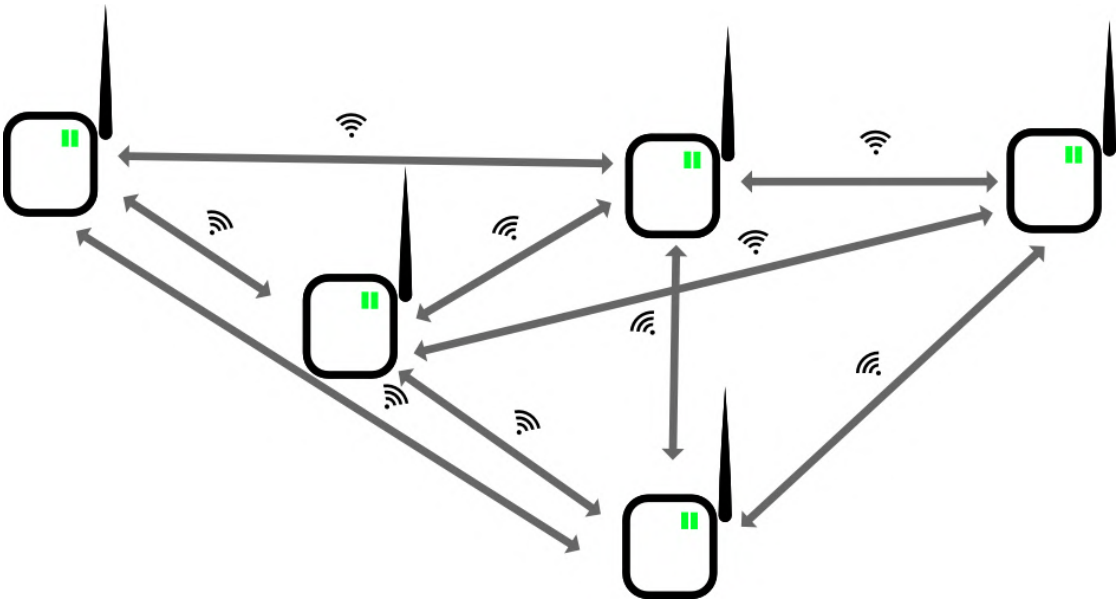
- Broadcast - topologia ta umożliwia bezpołączeniową transmisję jeden do wielu. Jest wykorzystywana np. do lokalnego dzielenia informacji czy nawigacji wewnątrz budynku (rys. 3.8).



Rysunek 3.7: Połączenie Bluetooth typu punkt-punkt.



Rysunek 3.8: Bluetooth broadcast.



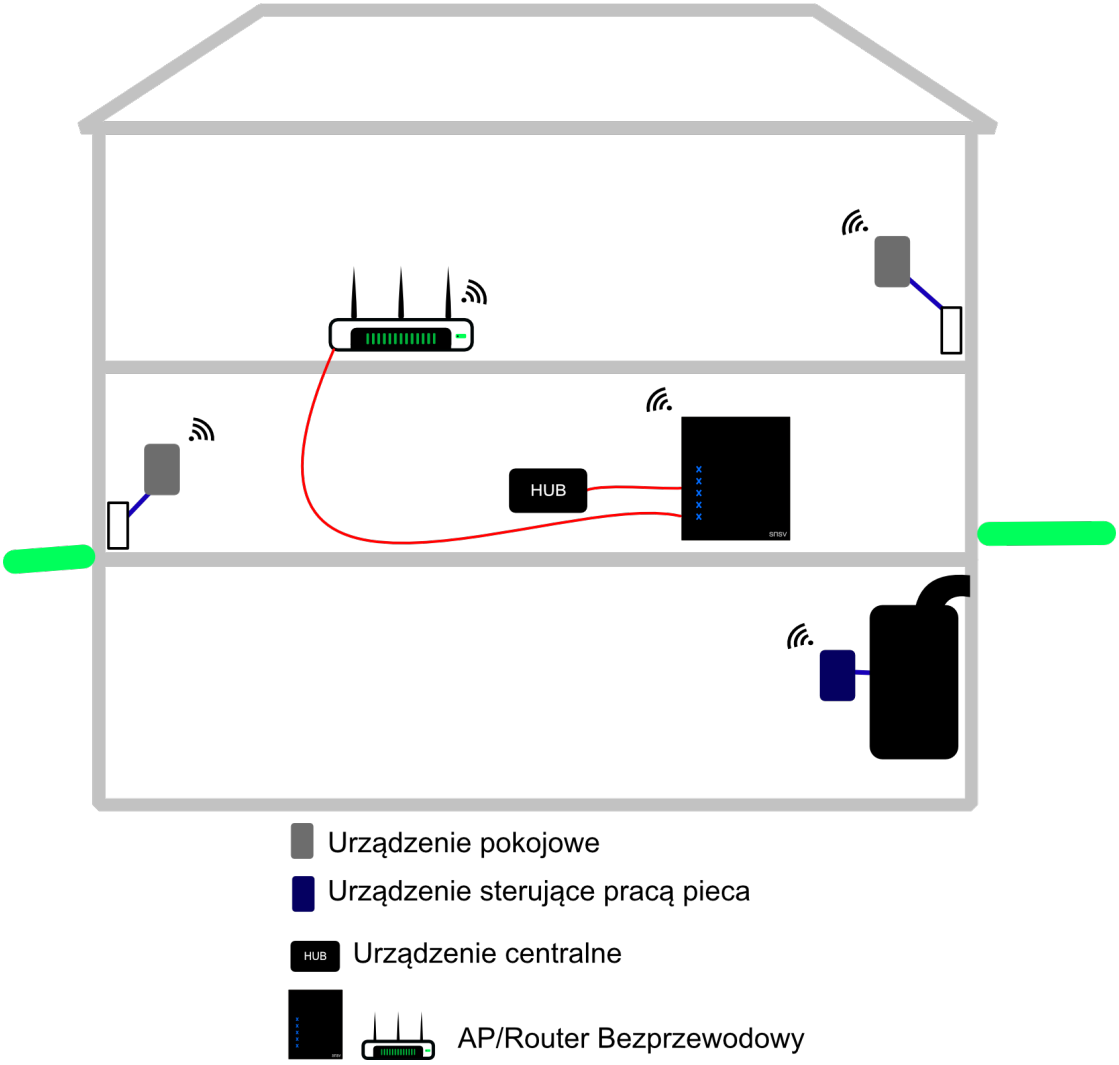
Rysunek 3.9: Topologia siatki sieci Bluetooth.

- Topologia siatki - umożliwia tworzenie rozleglejszych sieci poprzez bezpośrednią komunikację urządzeń oraz przekazywanie danych między nimi (rys. 3.9).

Bluetooth charakteryzuje się bardzo niskim poborem prądu oraz niską, choć w zupełności wystarczającą z punktu widzenia niniejszego projektu przepustowością.

Wszystkie analizowane formy komunikacji bezprzewodowej spełniają kryteria i umożliwiają realizację założeń systemu, ostatecznie jednak zdecydowano się na wybór komunikacji WiFi. Powodem takiego wyboru są przede wszystkim duża dostępność układów umożliwiających komunikację poprzez WiFi na rynku, ich niska cena (co jest niezwykle istotne w sytuacji, gdy system tworzy wiele urządzeń) oraz popularność w zastosowaniach IoT. Inną zaletą jest możliwość bezpośredniego dostępu urządzeń WiFi do sieci lokalnej (oraz internetu) bez konieczności stosowania urządzeń pośrednich (bramki itd.) (rys. 3.10). Dostępne dla konsumenta na rynku układy umożliwiające komunikację ZigBee (tj. np. XBee) są wyraźnie droższe niż układy oparte o wybraną technologię WiFi [5, 4] i jest to główny powód, dla którego nie zdecydowano się na wybór tej formy komunikacji. ESP-WIFI-MESH nie jest protokołem ustandaryzowanym, dlatego też nie zdecydowano się na wybór tej

technologii w celu uniknięcia ewentualnych problemów z implementacją komunikacji w systemie (najpopularniejsze biblioteki jak *painlessMesh* nie umożliwiają np. komunikacji protokołem IP [11], która choć nie jest konieczna, jak wspomniano wcześniej, jest zaletą).



Rysunek 3.10: Koncepcja systemu w istniejącej sieci lokalnej

3.3 Działanie systemu

Poruszanym w tym podrozdziale zagadnieniem, będzie wypracowanie zasad, według których koordynowane jest otwieranie oraz zamykanie zaworów grzejników pokojowych z włączaniem oraz wyłączaniem pieca. Celem niniejszego projektu jest realizacja tych zadań w sposób efektywny, a zatem wypracowane mechanizmy muszą prowadzić do zmniejszenia zużycia energii przy równoczesnym utrzymaniu w pomieszczeniach parametrów porządnych przez użytkownika (ustawiona temperatura oraz histereza). Należy również wziąć pod uwagę i dążyć do minimalizacji ilości włączeń i wyłączeń pieca oraz otwarć i zamknięć zaworów grzejników pokojowych.

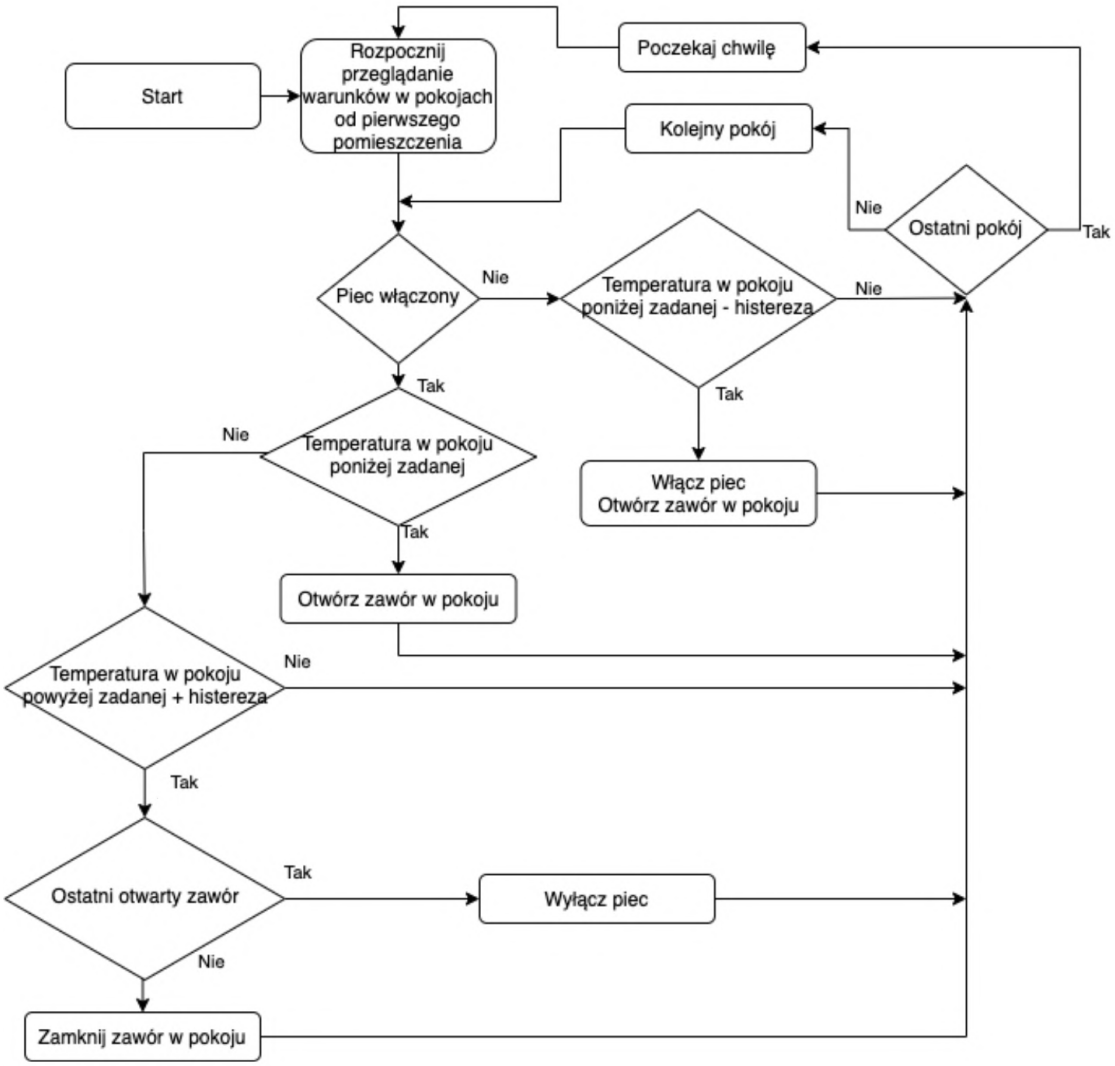
Pierwszym podejściem, jest po prostu otwieranie zaworu grzejnika gdy temperatura w pomieszczeniu spadnie poniżej zadanej temperatury pomniejszonej o histerezę, przy równoczesnym włączaniu w tej sytuacji pieca (jeśli jeszcze nie pracuje), oraz zamykaniu go gdy temperatura w pomieszczeniu uzyska wartość większą niż zadana temperatura powiększona o histerezę połączonym z wyłączaniem pieca (w sytuacji gdy było to ostatnie dogrzewane pomieszczenie). Takie podejście może jednak prowadzić do nadmiernej ilości włączeń i wyłączeń pieca, może bowiem dochodzić do sytuacji, że chwilę po zakończeniu dogrzewania jednego pomieszczenia i po wyłączeniu pieca, okaże się, że dogrzewania wymaga inne pomieszczenie.

Biorąc pod uwagę powyższy problem, jego rozwiązaniem byłoby wykorzystanie w procesie podjęcia decyzji o otwarciu zaworów grzejników stanu pieca. Drugim podejściem byłoby zatem otwarcie wraz z, spowodowanym spadkiem w którymś pomieszczeniu temperatury poniżej zadanej pomniejszonej o histerezę, włączeniem pieca zaworów we wszystkich pomieszczeniach i zamykanie ich kolejno gdy temperatura w danym pomieszczeniu uzyska wartość wyższą niż zadana powiększona o histerezę. Takie rozwiązanie ma jednak inne wady. Pierwszą z nich jest, że pomieszczenie z najniższą temperaturą (to z którego pomiar doprowadził do włączenia pieca) nagrzewałoby się stosunkowo wolno, ponieważ energia wytwarzana przez piec byłaby początkowo dzielona na wszystkie pomieszczenia. Drugą wadą jest nadmierna liczba otwarć i zamknięć zaworów w pomieszczeniach, które tak naprawdę nie wymagają wcale częstego dogrzewania grzejnikami pokojowymi, gdyż tempe-

ratura w nich utrzymuje się ponad nastawioną przez użytkownika bez takiego dogrzewania np. na skutek pracy w tych pomieszczeniach urządzeń elektrycznych emitujących ciepło tj. np. komputery. Prowadziłoby to również do niepotrzebnego zużywania energii na dogrzewanie tych pomieszczeń, pomimo iż temperatura w nich spełniałaby już wymogi użytkownika bez dogrzewania grzejnikami pokojowymi.

Aby rozwiązać problemy drugiego podejścia, należy w procesie podjęcia decyzji o zamykaniu i otwieraniu zaworów pokojowych wziąć pod uwagę zarówno stan pieca jak i temperaturę w danym pomieszczeniu. Trzecim i wybranym ostatecznie podejściem będzie zatem wprowadzenie dwóch progów temperaturowych otwarcia zaworu w zależności czy piec jest włączony, czy wyłączony. W rozwiązaniu tym przyjęto, że pierwszym progiem (dla pieca wyłączonego) będzie ustawiona temperatura pomniejszona o histerezę, a drugim (dla pieca włączonego) temperatura ustawiona. Kiedy w sytuacji gdy piec jest wyłączony, temperatura w pomieszczeniu spadnie poniżej pierwszego progu, zostanie w tym pomieszczeniu otwarty zawór grzejnika pokojowego oraz włączony zostanie piec. Gdy piec jest włączony, zawory będą otwierane w każdym pomieszczeniu, w którym temperatura jest lub spadnie poniżej drugiego progu temperaturowego. Zawory będą zamykane w pomieszczaniach, w których temperatura uzyska wartość ustawionej temperatury powiększonej o histerezę. Piec zostanie wyłączony wraz z uzyskaniem tej wartości w ostatnim pomieszczeniu, w którym otwarty jest zawór. W celu wykorzystania energii skumulowanej w piecu zawór w tym pomieszczeniu nie zostanie zamknięty.

Pomiar temperatury w pomieszczeniu oraz przekazanie średniego wyniku trzech ostatnich pomiarów do punktu centralnego będzie realizowany przez urządzenie pokojowe w odstępach jednominutowych. Podejmowanie decyzji o konieczności włączenia lub wyłączenia pieca oraz zamknięcia bądź otwarcia zaworu grzejnika pokojowego w którychś pomieszczeniach będzie wykonywane przez urządzenie centralne w odstępach minutowych. Działania te nie będą wykonywane przez urządzenia w sposób synchroniczny.



Rysunek 3.11: Schemat działania systemu.

3.4 Ocena realizowalności i kosztu wykonania projektu

Mając na uwadze szeroką dostępność układów elektronicznych, komputerów jednopłytkowych czy innych urządzeń mikroprocesorowych, ich relatywnie niską cenę, jak również nie nadto skomplikowane wymagania odnośnie do oprogramowania każdego z wyszczególnionych w tym rozdziale urządzeń stanowiących system, realizacja niniejszego projektu jest możliwa w rozsądnej liczbie roboczogodzin oraz rozsądnym kosztem (nakładem finansowym). Koszt realizacji jednego urządzenia pokojowego nie powinien przekroczyć kwoty 150zł - docelowo w ramach projektu powstaną 3 lub 4 prototypy tego urządzenia. Innymi kosztami będzie zakup i/lub montaż urządzeń, na które w ramach tej pracy realizowane będzie jedynie oprogramowanie (urządzenie sterujące pracą pieca, urządzenie centralne). Zgodnie ze wczesnymi szacunkami dokonanymi na podstawie znajomości specyfiki tych urządzeń (między innymi ich roli w systemie), koszt urządzenia centralnego i urządzenia sterującego pracą pieca nie powinien przekroczyć odpowiednio 300zł (komputer jednopłytkowy) i 100zł (układ oparty na module NodeMCU). Należy mieć na uwadze, że przyjęte tutaj koszty realizacji czy zakupu poszczególnych urządzeń są konserwatywne, i ostatecznie prawdopodobnie okażą się one niższe.

Rozdział 4

Koncepcja projektowa

4.1 Urządzenie centralne (hub)

Z uwagi na to, że bardzo często w domu znajduje się już jakieś stale włączone urządzenie mikroprocesorowe (np. serwer NAS, Raspberry Pi itd.) rozsądnym byłoby aby mogło ono pełnić również funkcję urządzenia centralnego projektowanego systemu w celu ograniczenia nadmiarowej ilości urządzeń w sieci.

Urządzenie centralne systemu należy zatem rozumieć jako dowolne urządzenie lub urządzenia świadczące w sieci pewną usługę lub zbiór usług. Ponieważ urządzenia mikroprocesorowe oparte są o różne architektury, w celu zapewnienia największej przenoszalności oprogramowania świadczącego te usługi pożądanym jest, aby było ono zrealizowane w językach wysokiego poziomu, których interpretacja jest możliwa na szerokiej gamie urządzeń, lub też które są kompilowalne na wiele platform docelowych.

Oprogramowanie urządzenia centralnego - oprogramowanie sterujące całością systemu - musi gromadzić ostatnie dane pomiarowe urządzeń pokojowych oraz dane o stanie pieca i podejmować na podstawie tych danych decyzję o otwarciu bądź zamknięciu w danym pomieszczeniu zaworu grzejnika pokojowego lub też o włączeniu lub wyłączeniu pieca. Oprogramowanie to musi również implementować interfejs komunikacji z innym oprogramowaniem (tzw. API), umożliwiający np. zmianę ustawień i parametrów systemu i konieczny w celu komunikacji z interfejsem użytkownika.

Z uwagi na dużą popularność, szeroką bazę bibliotek oraz na możliwość uruchomiania kodu na wielu platformach docelowych, zadecydowano, że oprogramowanie zarządzające całością systemu zostanie zrealizowane w języku JavaScript (NodeJS). Język ten umożliwia również łatwą implementację API (np. dzięki bibliotece Express). Interfejs ten zostanie zaimplementowany zgodnie z zasadami projektowania REST (zmiana stanu poprzez reprezentacje). Takie podejście umożliwi łatwą implementację interfejsu użytkownika oraz łatwą integrację systemu z innym oprogramowaniem zarządzania inteligentnym domem jak np. Homebridge czy Home Assistant.

Domyślnie rolę urządzenia centralnego systemu będzie pełnić Raspberry Pi3 pracujące pod kontrolą systemu operacyjnego Raspberry Pi OS (Raspbian).

4.2 Urządzenie pokojowe

Zgodnie z wypracowaną w poprzednim rozdziale koncepcją architektury systemu, urządzenie pokojowe musi realizować trzy funkcje - pomiar temperatury, zamykanie oraz otwieranie zaworu grzejnika pokojowego oraz komunikację z urządzeniem centralnym systemu. Realizacją tych zadań powinien sterować mikrokontroler.

4.2.1 Wybór elementów elektronicznych

Komunikacja i sterowanie realizacją zadań

Z uwagi na wygodę, możliwość większej integracji układu oraz koszty, najlepszym rozwiązaniem wydaje się zastosowanie mikrokontrolerów wyposażonych w moduł WiFi (niżeli osobno dwóch układów - mikrokontrolera i modułu WiFi).

Przed dokonaniem porównania dostępnych na rynku modułów, należy określić kryteria, według których będą one porównywane, a zatem wyznaczyć istotne z punktu widzenia projektowanego urządzenia pokojowego parametry.

- Procesor - procesor mikrokontrolera musi być zdolny wykonać wszystkie operacje związane z zadaniami urządzenia w krótkim czasie

- Dostępność - wykorzystany w projekcie układ powinien być łatwo dostępny, aby możliwe było zbudowanie zaprojektowanego urządzenia
- Łatwość programowania - moduł umożliwiający łatwe umieszczanie programu w pamięci mikrokontrolera pozwoli na szybsze uruchamianie i testowanie tworzonego oprogramowania
- Pobór mocy w stanie ciągłej pracy - należy dążyć do jak najmniejszego zużycia energii elektrycznej przez projektowane urządzenie
- Wymiary - mniejsze wymiary każdego elementu projektowanego urządzenia pozwolą zmniejszyć ostateczny rozmiar tego urządzenia
- Cena - ponieważ na system składa się wiele urządzeń pokojowych, istotnym jest, aby ich cena była niska (a co za tym idzie cena każdego elementu tego urządzenia), gdyż wpływa ona znacząco na koszt całego systemu
- Przetwornik analogowo-cyfrowy - jest konieczny w przypadku potencjalnego zastosowania analogowego czujnika temperatury

	ESP8266			ESP32	
	NodeMCU v3	ESP-WROOM02	ESP-12E	ESP-WROOM32	ESP32 Development Board
Standard WiFi	b/g/n			b/g/n	
Procesor	80-160MHz			2x 80-240MHz	
Przetwornik A/C	10 bitowy, 0-1V			12 bitowy	
Łatwość programowania	Łatwo - przez USB (wbudowany na PCB konwerter USB-UART)	Trudniej (np. przez konweter USB-UART lub programator)		Trudniej (np. przez konweter USB-UART lub programator)	Łatwo - przez USB (wbudowany na PCB konwerter USB-UART)
Dostępność	Wysoka			Wysoka	
Pobór mocy (praca ciągła)	Mniejszy			Większy	
Wymiary	49 x 25mm	20 x 18mm	24 x 16mm	25.5 x 18mm	55 x 28 mm
Cena	~10zł	~6.50zł	~6zł	~10zł	~18-20zł

Tablica 4.1: Zestawienie rozważanych modułów odpowiedzialnych za kounikację i sterowanie realizacją zadań urządzenia pokojowego.

W analizie wzięto pod uwagę moduły oparte o mikronotrolery ESP32, ESP8266 oraz moduły WFI32E01PC i Arduino Portenta H7. Te ostatnie (WFI32E01PC i Arduino Portenta H7) odrzucono od razu z uwagi na wysoką cenę i aktualnie niską popularność w zastosowaniach IoT. Resztę zestawiono w tablicy 4.1. Parametry

w niej zawarte ustalono na podstawie dokumentacji technicznych zawartych weń modułów [9, 1] i/lub ofert sprzedaży tych modułów w popularnych sklepach branżowych (np. Botland) oraz popularnych serwisach sprzedażowych tj. Allegro czy Aliexpress.

Z uwagi na prostotę realizowanych przez układ zadań, nie jest konieczny wybór modułu opartego na nowszym i bardziej zaawansowanym, ale i droższym mikrokontrolerze ESP32, zwłaszcza że pobiera on w pracy ciągłej więcej energii niż mikrokontroler ESP8266. Małe wymiary, niska cena i wysoka dostępność na rynku przesądziły o wyborze modułu ESP12E. Moduł NodeMCU v3 zostanie wykorzystany w procesie tworzenia i testowania oprogramowania.

Pomiar temperatury

Pomiaru temperatury można dokonywać przy pomocy czujnika temperatury (cyfrowego lub analogowego) lub własnego układu z termistorem. Lepszym rozwiązaniem będzie zastosowanie gotowego czujnika - jest to rozwiązanie prostsze i dokładniejsze. Podobnie jak w przypadku rozważanych w poprzedniej sekcji modułów należy sprecyzować kryteria, wedle których dokonano porównania dostępnych na rynku czujników temperatury.

- Napięcie zasilania - w celu ograniczenia ilości koniecznych na płytce drukowanej linii zasilających warto, aby czujnik był zasilany napięciem już na niej wypracowanym (np. dla mikrokontrolera).
- Zakres pomiarowy - zakres pomiarowy powinien obejmować temperatury pokojowe.
- Dokładność - czujnik temperatury powienien cechować się dobrą dokładnością.
- Rozdzielczość - przetwornik analogowo-cyfrowy czujników cyfrowych powinien charakteryzować się odpowiednią rozdzielczością.
- Czas odpowiedzi - biorąc pod uwagę parametry systemu, czujnik temperatury musi dokonywać pomiaru temperatury odpowiednio szybko.

	Cyfrowe					Analogowe	
	DS18B20	DHT22	DHT11	TMP102	AMP2320	LM35(DZ)	TMP36GT9Z
Napięcie zasilania	3 - 5.5V	3.3 - 6V	3.3 - 5.5V	1.4 - 3.6V	3.3 - 5.5V	4 - 30V	2.7 - 5.5V
Zakres pomiarowy	-55 do 125 °C	-40 do 80 °C	-20 do 60 °C	-40 do 125 °C	-40 do 80 °C	-50 do 150 °C	-40 do 120 °C
Dokładność	+/- 0.5 °C	+/- 0.5 °C	+/- 2 °C	+/- 0.5 °C	+/- 0.5 °C	+/- 0.5 °C	+/- 2 °C
Rozdzielczość	12 bitów (0.0625 °C)	8 bitów (0.1 °C)	8 bitów (1 °C)	12 bitów (0.0625 °C)	16 bitów (0.1 °C)	N/A 10.0 mV/°C	0.5 °C 10.0 mV/°C
Czas odpowiedzi	<2s	2s	6 - 15s	<2s	2s	N/A	N/A
Cena	~1.5zł	~10zł	~3.5 zł	~5zł	~10zł	~1.5zł	~4.5zł

Tablica 4.2: Zestawienie rozważanych modułów-czujników temperatury.

- Cena

Wzięte pod uwagę czujniki temperatury zestawiono w tablicy 4.2, a zawarte w niej parametry pochodzą z dokumentacji technicznych rozważanych układów, ofert sprzedaży tych układów w popularnych sklepach branżowych (np. Botland) oraz popularnych serwisach sprzedażowych tj. Allegro czy Aliexpress.

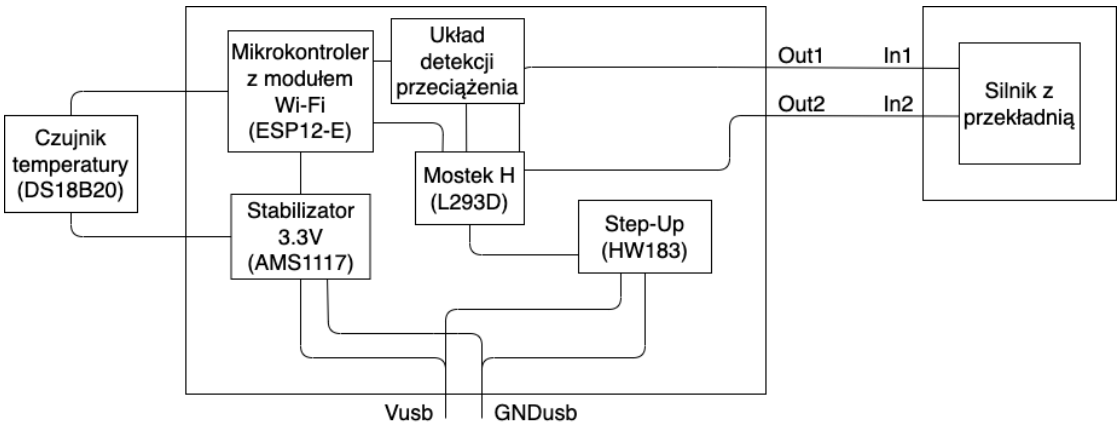
- Wszystkie zestawione czujniki posiadają odpowiedni zakres pomiarowy oraz napięcia zasilania.
- Pomiary będą dokonywane w około minutowych odstępach, więc czas odpowiedzi nie jest istotnym kryterium wyboru (wszystkie czujniki spełniają kryterium).
- Biorąc pod uwagę specyfikę projektu potrzebna jest rozdzielczość w okolicach 0.1 czy 0.2°C (lub większa). Z tego powodu odrzucono czujnik DHT11.
- Z uwagi na niezbyt dokładny przetwornik analogowy wybranego mikrokontrolera preferowanym rozwiązaniem będzie czujnik cyfrowy.
- LM35 z uwagi na swoją dokładność pomiarową oraz popularność zdaje się lepszym rozwiązaniem w swojej kategorii (przetworników analogowych).
- Najlepszym wyborem wśród czujników cyfrowych będą DS18B20 i TMP102 z uwagi na swoją niską cenę i wysoką rozdzielczość pomiarową.
- Czujniki DHT22 i AM2320 byłyby dobrym rozwiązaniem gdyby w przyszłości system miał być poszerzony o funkcjonalność pomiaru wilgotności powietrza w pomieszczeniach.

Ostatecznie zdecydowano się na wybór czujnika DS18B20 z uwagi na jego niską cenę oraz dobrą dokładność i rozdzielczość.

Inne elementy

- Mostek H - konieczny w celu sterowania pracą silnika zamykającego/otwierającego zawór grzejnika pokojowego - L293D
- Regulowana przetwornica Step-Up - daje możliwość zastosowania różnych silników z przekładnią - HW-183
- Stabilizator napięcia 3.3V - stabilizujący napięcie dla mikrokontrolera - AMS1117

Na rysunku 4.1 przedstawiono schemat blokowy urządzenia.



Rysunek 4.1: Schemat blokowy urządzenia pokojowego.

4.2.2 Język programowania i narzędzia programistyczne

Oprogramowanie urządzenia pokojowego musi obsługiwać połączenie mikrokontrolera z siecią WiFi, pobierać oraz obrabiać dane z czujnika temperatury, umożliwiać użytkownikowi wstępną konfigurację parametrów sieci i innych parametrów wymaganych do późniejszego połączenia urządzenia z resztą systemu, obsługiwać otwieranie i zamykanie zaworu grzejnika pokojowego, wymianiać informacje z urządzeniem centralnym (to jemu wysyłane są dane pomiarowe oraz na

jego żądanie otwierany jest bądź zamykany zawór) oraz implementować zachowanie urządzenia w razie utracenia połączenia z urządzeniem centralnym.

Zadecydowano, że oprogramowanie urządzenia pokojowego zostanie zrealizowane w Arduino Framework (C++). Taka decyzja podyktowana jest bardzo dużą popularnością tej platformy, szeroką dostępnością bibliotek oraz stosunkową łatwością języka C++. Ponieważ środowisko Arduino IDE jest bardzo prymitywne, oprogramowanie zostanie napisane w znacznie bardziej rozbudowanym i zaawansowanym środowisku Platform IO.

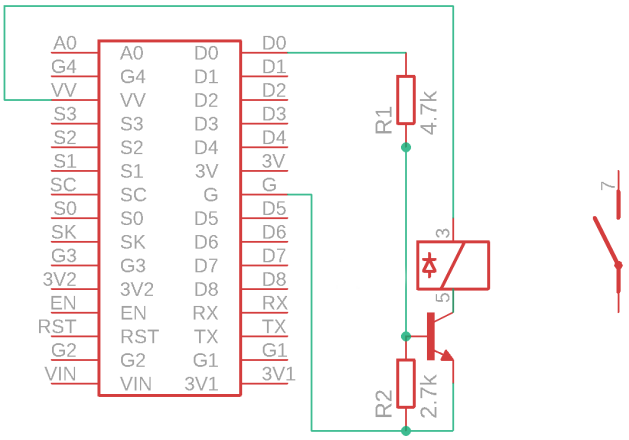
4.2.3 Schemat urządzenia i projekt płytki drukowanej

Schemat urządzenia i projekt płytki drukowanej zostaną wykonane w programie EAGLE.

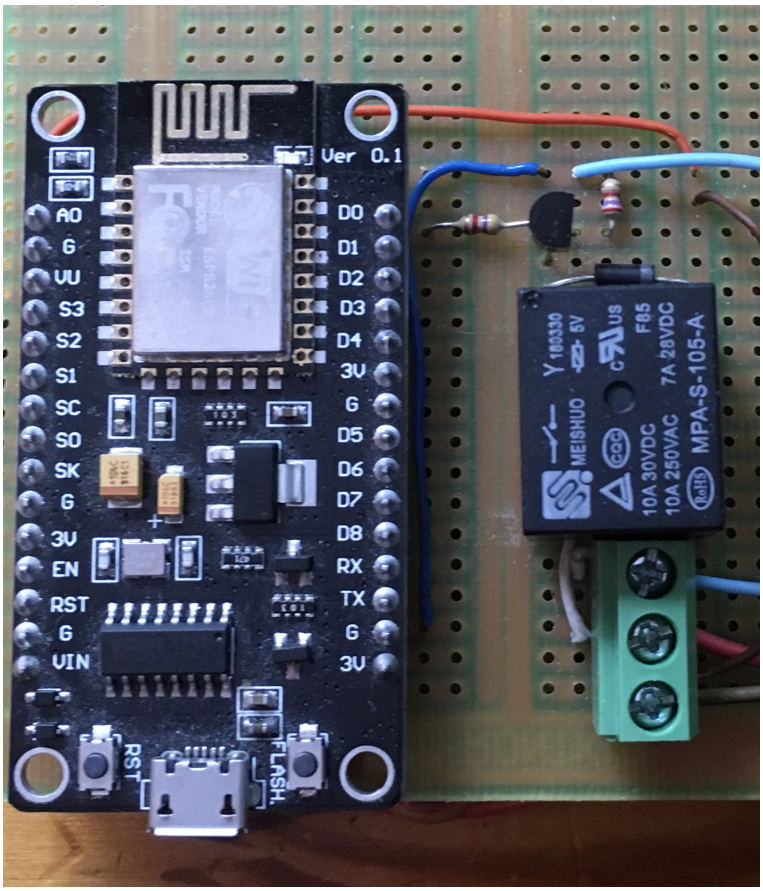
4.3 Urządzenie sterujące pracą pieca

Urządzenie sterujące pracą pieca zostało zaprojektowane i zbudowane niezależnie od niniejszej pracy. W ramach pracy zrealizowano natomiast jego oprogramowanie. Urządzenie to składa się z modułu NodeMCU v3, tranzystora, diody oraz przekaźnika. Schemat urządzenia przedstawiono na rysunku 4.2, a jego zdjęcie na rysunku 4.3.

Z uwagi na to, że urządzenie sterujące pracą pieca jest oparte o ten sam mikrokontroler co urządzenie pokojowe (ESP8266), co za tym idzie spora część kodu programu (np. odpowiedzialna za wstępną konfigurację, łączność WiFi, wymianę informacji z urządzeniem centralnym) może być wspólna z urządzeniem pokojowym, jego oprogramowanie zostanie zrealizowane z wykorzystaniem tych samych technologii co oprogramowanie urządzenia pokojowego - Arduino Framework (C++), Platform IO.



Rysunek 4.2: Schemat urządzenia sterującego pracą pieca



Rysunek 4.3: Urządzenie sterujące pracą pieca

4.4 Protokół wymiany informacji

Charakterystyka systemu wymaga możliwości wymiany informacji pomiędzy urządzeniami go stanowiącymi - urządzenia pokojowe i urządzenie sterujące piecem muszą w jakiś sposób przesyłać informacje o swoim stanie do urządzenia centralnego, urządzenie centralne musi w jakiś sposób wysyłać do urządzenia sterującego piecem żądania włączenia lub wyłączenia pieca oraz do urządzeń pokojowych żądania otwarcia lub zamknięcia zaworu grzejnika.

Idealnym do tego celu jest protokół MQTT rozwijany właśnie w celu komunikacji urządzeń IoT [17]. Protokół MQTT opiera się o mechanizm publikacja-subskrybcja [12] a jego implementacja wymaga bardzo małej ilości zasobów (moc procesora, przepustowość sieci) [17].

Fakt istnienia gotowych bibliotek implementujących obsługę protokołu MQTT dla wybranych prędzej języków programowania oznacza możliwość szybkiej implementacji wymiany informacji w oprogramowaniu urządzeń stanowiących system.

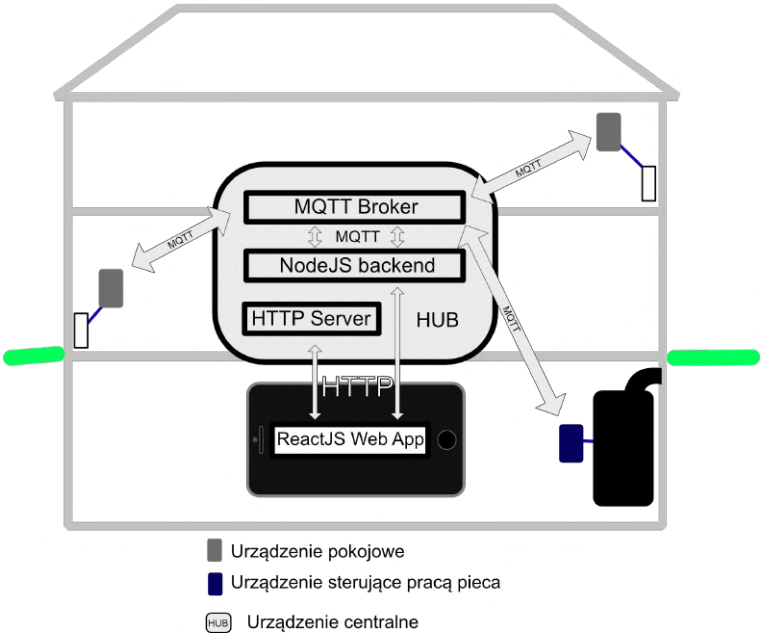
Ponieważ protokół MQTT wymaga do swojego działania tzw. brokera [12], jego funkcję będzie domyślnie pełnić urządzenie centralne systemu (rys. 4.4). Funkcję tą będzie mogło również pełnić inne urządzenie (rys. 4.5).

4.5 Interfejs użytkownika

Podjęta w innej części tego rozdziału decyzja o realizacji interfejsu API oprogramowania sterującego całością systemu zgodnie z zasadami projektowania REST daje możliwość implementacji interfejsu użytkownika na wiele sposobów.

Jednym z nich jest implementacja go jako natywnej aplikacji dla wiodących systemów mobilnych jak Android czy iOS. Takie podejście wymagałoby jednak stworzenie oddzielnej aplikacji dla każdej platformy i nie byłoby możliwości zarządzania systemem np. z poziomu komputerów osobistych.

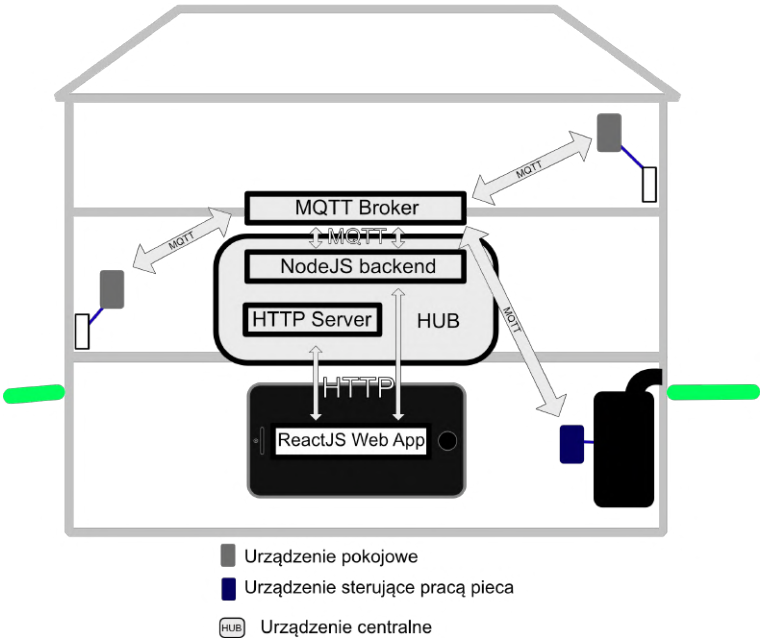
Alternatywnym rozwiązaniem jest stworzenie tzw. aplikacji webowej - interfejsu użytkownika uruchamianego w przeglądarce internetowej (lub silniku renderowania przeglądarki internetowej). To podejście gwarantuje wieloplatformowość - aplikację można uruchomić na każdym urządzeniu z przeglądarką internetową obsługującą technologie wykorzystywane w stworzonej aplikacji.



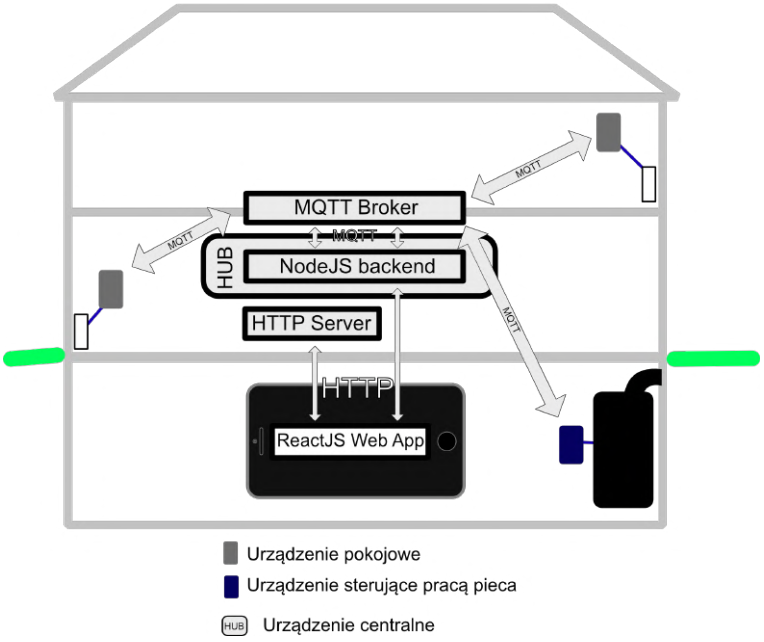
Rysunek 4.4: Domyślna konfiguracja systemu - urządzenie centralne w roli serwera WWW i brokera MQTT

Zdecydowano więc, że interfejs użytkownika zostanie zrealizowany jako aplikacja webowa wykonana przy pomocy javascriptowej biblioteki React.

Serwer WWW, na którym umieszczona zostanie aplikacja interfejsu użytkownika, będzie domyślnie pracować na urządzeniu centralym systemu (rys. 4.4). Możliwe będzie również umieszczenie aplikacji na serwerze WWW pracującym na innym urządzeniu (rys. 4.6).



Rysunek 4.5: Możliwa konfiguracja systemu - urządzenie centralne w roli serwera WWW



Rysunek 4.6: Możliwa konfiguracja systemu

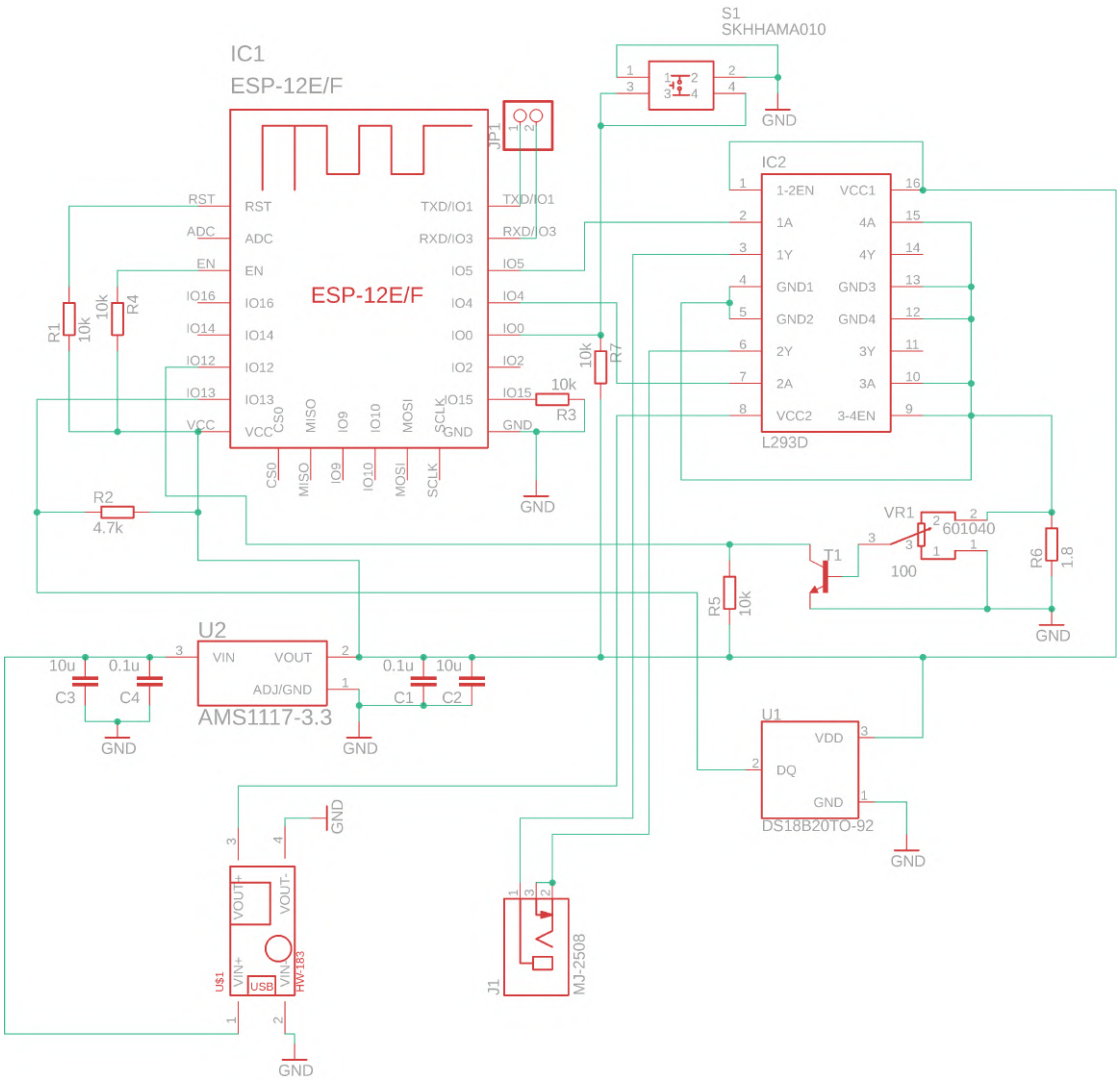
Rozdział 5

Projekt urządzenia pokojowego

5.1 Schemat ideowy urządzenia

Schemat ideowy urządzenia przedstawiono na rysunku 5.1.

Zgodnie z dokumentacją techniczną modułu ESP-12E [1] oraz dokumentacją techniczną mikrokontrolera ESP8266 [10] wymuszenie na wejściu RST (Reset) oraz EN (Enable) stanu wysokiego sprawia, że moduł ESP-12E jest cały czas aktywny, dlatego też wejścia te zostały połączone kolejno przez rezystor R1 i R4 do lini 3.3V. Zgodnie z dokumentacją techniczną modułu ESP12-E [1] w celu wprowadzenia urządzenia w stan wczytywania oprogramowania do pamięci FLASH przez interfejs UART, w chwili włączania układu na wejściu IO 15 i IO0 powinien być wymuszony stan niski. Uruchomienie na układzie programu z pamięci flash wymaga wymuszenia na tych wejściach odpowiednio stanu niskiego i wysokiego. Z tego powodu wejście IO15 zostało połączone przez rezystor R3 z masą, natomiast wejście IO0 przez rezystor R7 z linią 3.3V. Wejście to połączone zostało również z przyciskiem S1, którego przyciśnięcie wymusza na tym wejściu stan niski. Takie podłączenie zapewnia normalne uruchomienie układu po podłączniu urządzenia do zasilania gdy nie jest przyciśnięty przycisk S1, oraz wprowadzenie go w stan wczytywania oprogramowania do pamięci flash przez interfejs UART gdy w chwili podłączania go do zasilania przyciśnięty jest przycisk S1. Aby zapewnić możliwość programowania układu przez interfejs UART, wyjścia TXD(IO1) i RXD(IO3) połączono z dwupinowym gniazdem szpilkowym typu goldpin. Umiesz-

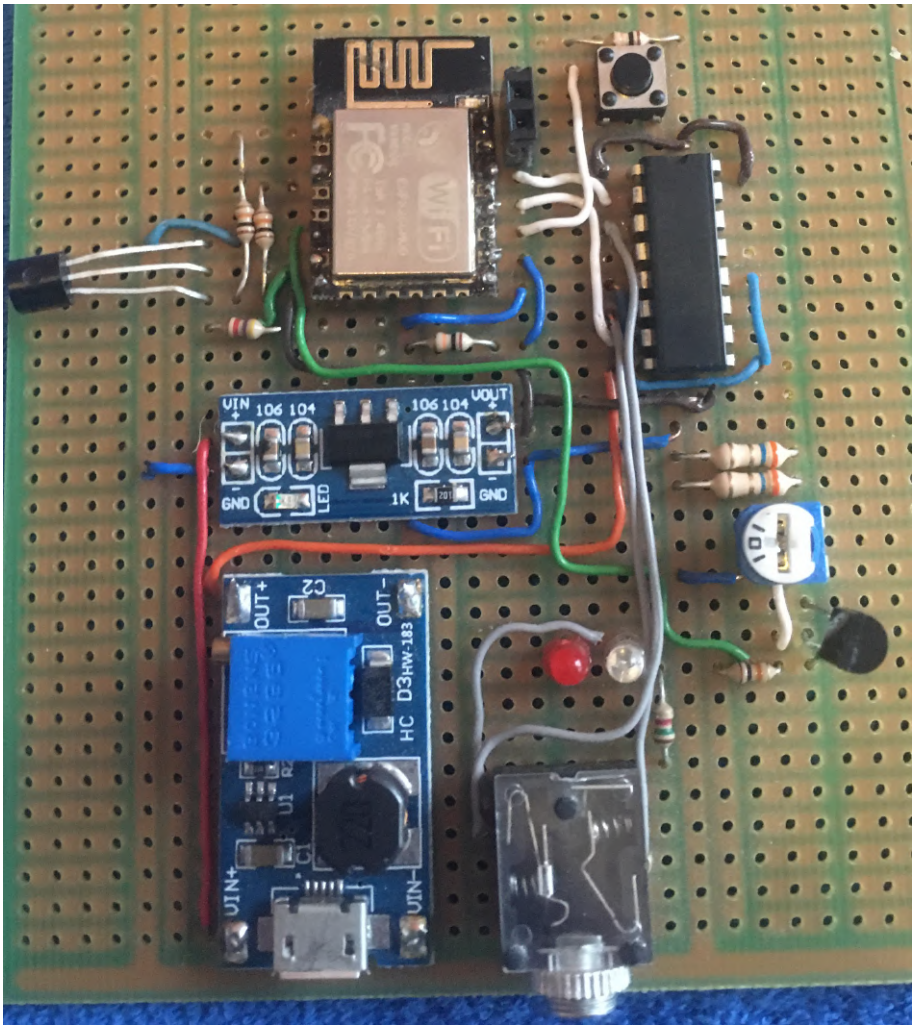


Rysunek 5.1: Schemat ideowy urządzenia pokojowego.

czenie kondensatorów C1 i C2 pomiędzy linią 3.3V a masą oraz kondensatorów C3 i C4 pomiędzy masą a linią 5V (wejście zasilające USB) ma na celu poprawienie stabilności działania układu. Wyjścia IO5 i IO4 układu ESP-12E zostały połączone z wejściami 1A i 2A układu L293D (mostek H). Zgodnie z dokumentacją techniczną tego układu [21] pojawienie się na jednym z tych wejść stanu wysokiego skutkuje pojawieniem się na odpowiadającym mu wyjściu (1Y dla 1A, 2Y dla 2A) napięcia podanego na wejście VCC2 tego układu. Wyjścia 1Y i 2Y zostały połączone z kanałami gniazda J1. Ustawienie na wyjściach IO5 oraz IO4 modułu ESP-12E odpowiednio stanów niskiego i wysokiego lub wysokiego i niskiego spowoduje obrót silnika przyłączonego do gniazda J1 w jedną lub drugą stronę. Wejście zasilania VCC1 (zasilanie logiki układu) mostka H zostało podłączone do linii 3.3V. Linia ta został również podłączona do wejścia 1-2EN w celu aktywacji wyjść 1Y i 2Y. Wejście zasilania VCC2 zostało przyłączone do linii regulowanego napięcia wyjściowego przetwornicy step-up. Wejścia masy układu (GND1, GND2, GND3, GND4) oraz nieużywane wejścia 3A i 4A, jak również wejście 3-4EN (w celu dezaktywacji wyjść 3Y oraz 4Y) zostały połączone z układem detekcji przeciążenia. Układ detekcji przeciążenia składa się z rezystora R6, potencjometru VR1 oraz tranzystora T1. Na skutek dużej wartości natężenia prądu przepływającego przez mostek H w chwili zatrzymaniem silnika (gdy zawór grzejnika pokojowego zostanie maksymalnie otwarty lub zamknięty) pomiędzy bazą a emiterem tranzystora pojawi się napięcie, które spowoduje jego przejście w stan nasycenia co z kolei doprowadzi do pojawienia się na wejściu IO12 (podłączonego przez rezystor podciągający R5 do lini 3.3V) modułu mikrokotrolera stanu niskiego. Czujnik temperatury DS18B20 został podłączony do modułu mikrokontrolera zgodnie z dokumentacją techniczną [6].

W celu weryfikacji poprawności schematu ideowego, na jego podstawie został zbudowany pierwszy prototyp urządzenia (rys. 5.2).

Prototyp potwierdził poprawność schematu ideowego - zbudowane urządzenie działa poprawnie. Wadą prototypu jest umieszczenie czujnika temperatury w zbyt bliskiej odległości do mikrokontrolera, co powoduje znacznie zawyżone odczyty temperatury. W projekcie płytki drukowanej czujnik temperatury należy umieścić jak najdalej od tego układu.

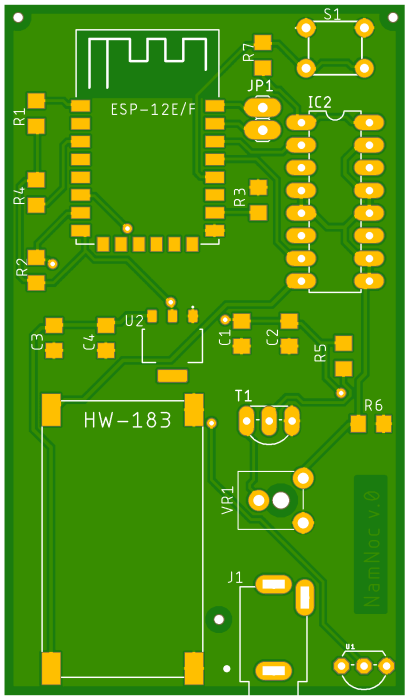


Rysunek 5.2: Pierwszy prototyp urządzenia pokojowego.

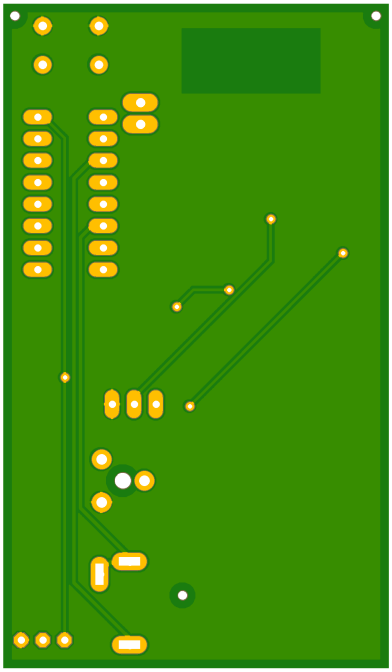
5.2 Projekt płytki drukowanej urządzenia

Projekt płytki drukowanej wykonany w programie EAGLE został umieszczony na rysunkach 5.3 i 5.4.

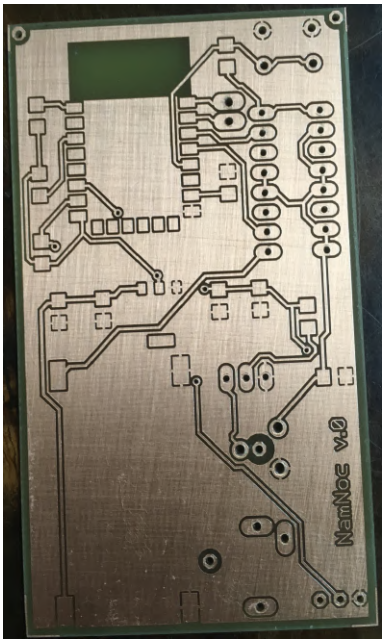
W celu weryfikacji poprawności projektu płytki zlecono jej wykonanie w kilku egzemplarzach. Zrealizowana na podstawie projektu płytka została przedstawiona na rysunkach 5.5 i 5.6.



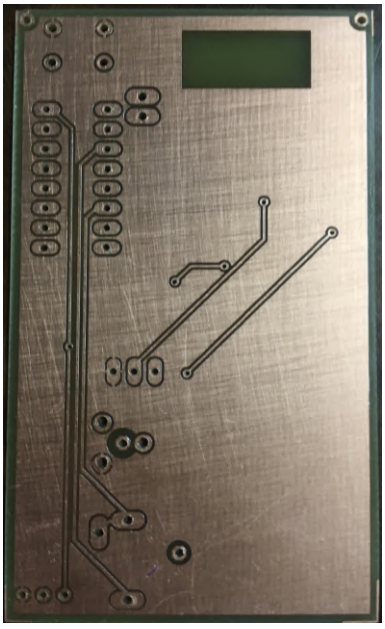
Rysunek 5.3: Projekt płytki drukowanej urządzenia pokojowego - góra.



Rysunek 5.4: Projekt płytki drukowanej urządzenia pokojowego - dół.

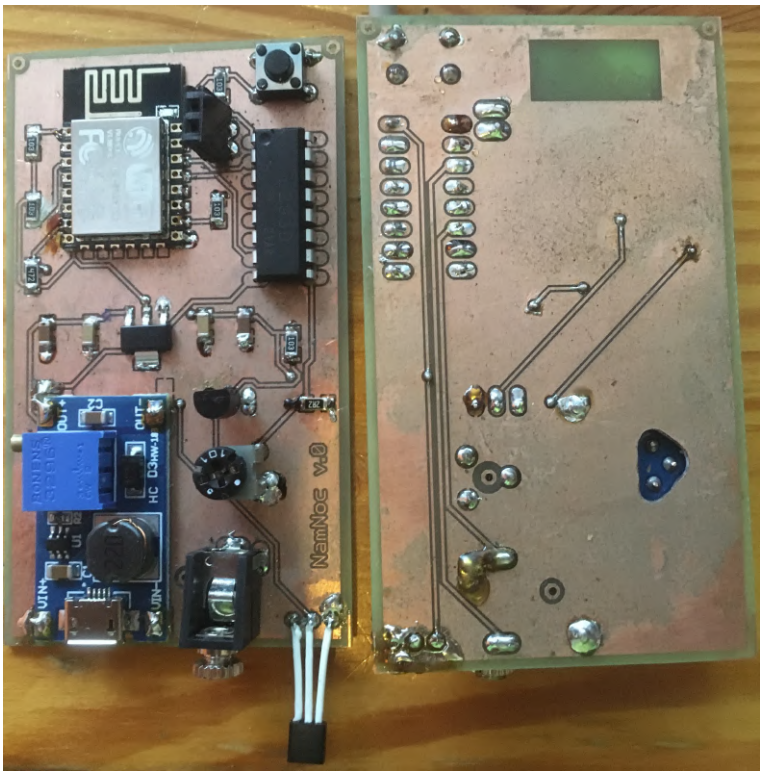


Rysunek 5.5: Płytka drukowana urządzenia pokojowego - góra.



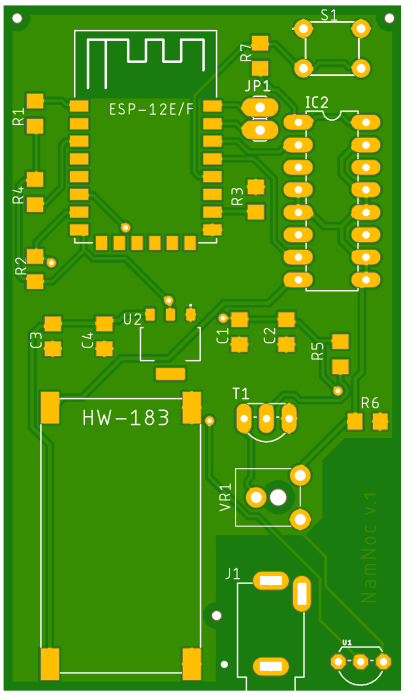
Rysunek 5.6: Płytka drukowana urządzenia pokojowego - dół.

W tym samym celu zbudowano kolejne prototypy urządzenia poprzez umieszczenie na zrealizowanych płytkach elementów elektronicznych. Góra trzeciego prototypu oraz dół drugiego prototypu urządzenia przedstawiono na rysunku 5.7.

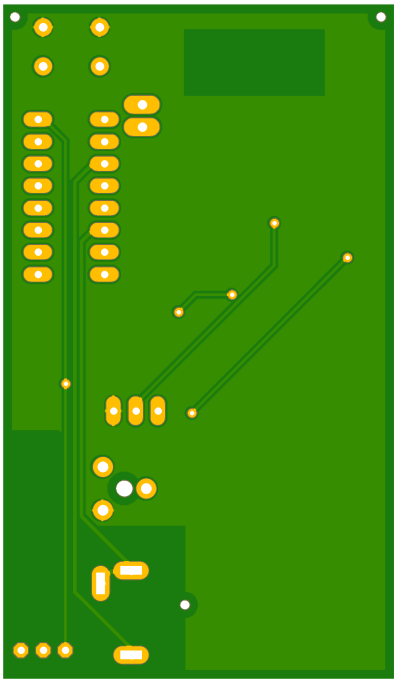


Rysunek 5.7: Trzeci prototyp (góra) oraz drugi prototyp (dół) urządzenia pokojowego.

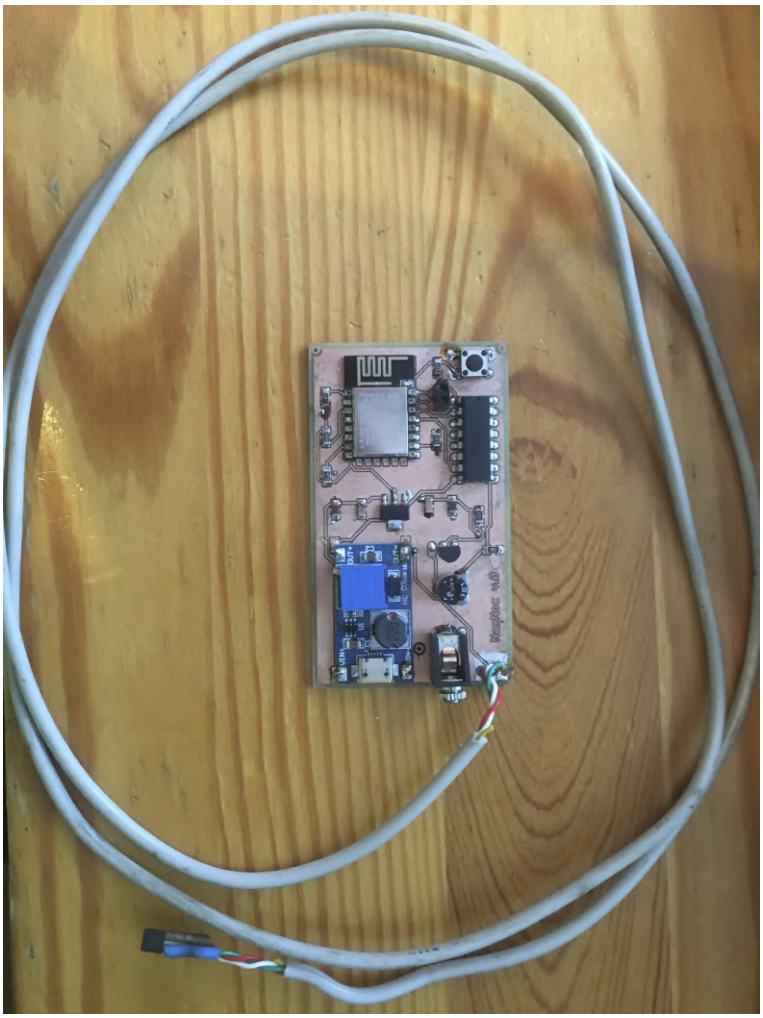
Niestety pomimo przeniesienia czujnika temperatury dalej od modułu mikrokontrolera, jego odczyty nadal były zawyżone. Jest to prawdopodobnie spowodowane faktem, że ciepło jest przewodzone przez wylaną na całym obszarze płytki masę. Należałoby zatem zmodyfikować projekt płytki w taki sposób, aby masa nie była wylana w najbliższym otoczeniu czujnika temperatury (patrz rysunki 5.8 i 5.9). Niestety ograniczona czasem możliwość zlecenia realizacji płytek wykonanych zgodnie z poprawionym projektem spowodowała, że zadecydowano rozwiązać ten problem w inny sposób - poprzez instalację czujnika temperatury na przewodzie. Zmiana ta została wprowadzona w prototypie drugim (rys. 5.10).



Rysunek 5.8: Poprawiony projekt płytki drukowanej urządzenia pokojowego - góra.



Rysunek 5.9: Poprawiony projekt płytki drukowanej urządzenia pokojowego - dół.



Rysunek 5.10: Drugi prototyp urządzenia pokojowego - czujnik temperatury na przewodzie.

Rozdział 6

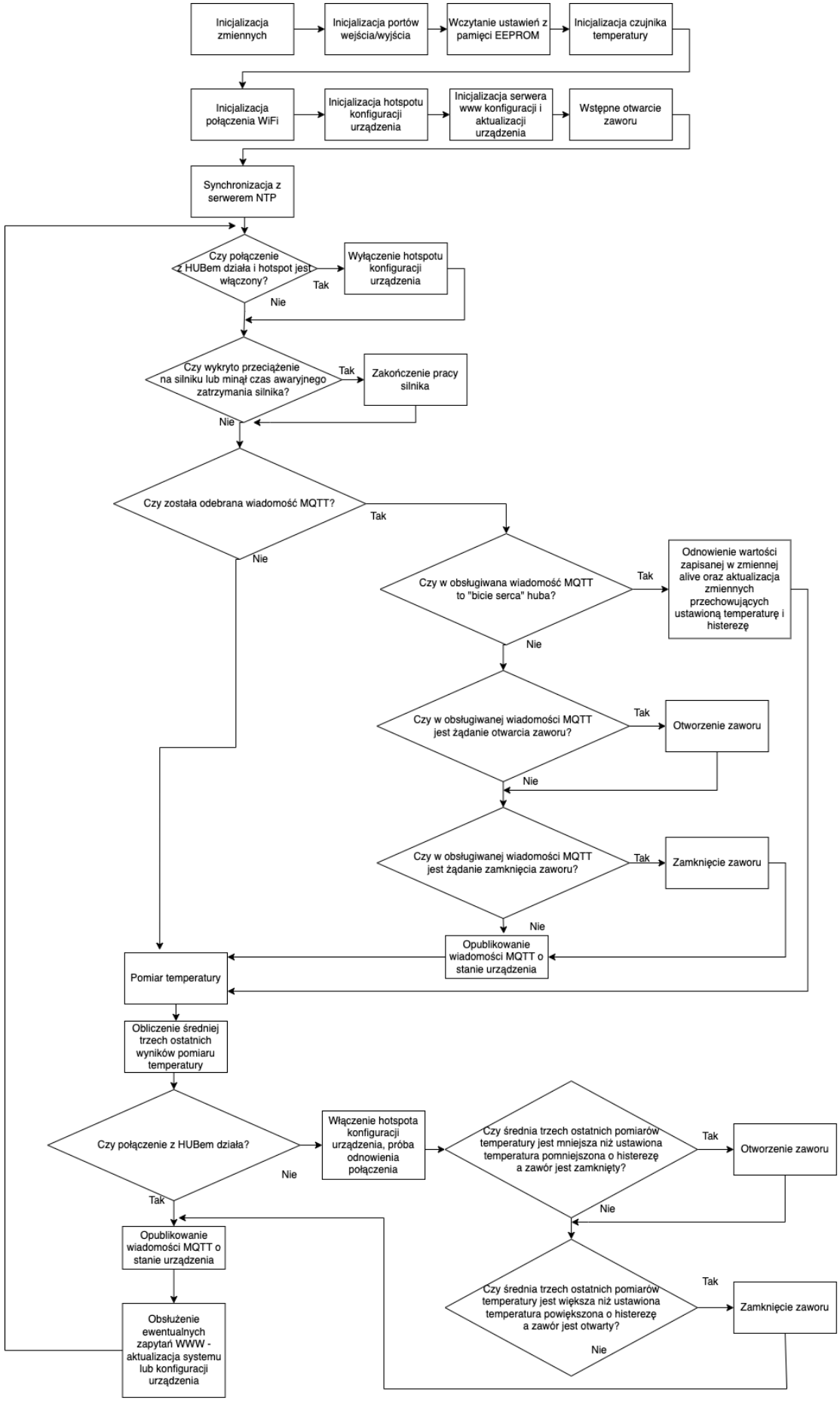
Oprogramowanie

6.1 Oprogramowanie urządzenia pokojowego

Schemat blokowy oprogramowania urządzenia pokojowego przedstawiono na rysunku 6.1.

6.1.1 Wykaz istotnych zmiennych

- *bool opened* - przechowuje stan zaworu (0/false - zawór zamknięty, 1/true - zawór otwarty)
- *int alive* - zmienna przechowująca status połączenia z urządzeniem centralnym (zakres od 0-5) - dekrementowana co minutę i ustawiana na wartość 5 w przypadku odebrania tzw. bicia serca (heartbeat) urządzenia centralnego (wysyłanego przez urządzenie centralne również w odstępach minutowych)
- *double lastThreeAvgTemp* - przechowuje średnią wartość trzech ostatnich pomiarów temperatury
- *double temps[3]* - zmienna tablicowa przechowująca wartości trzech ostatnich pomiarów temperatury
- *String timeStamp* - zmienna przechowująca znacznik czasowy ostatniego pomiaru temperatury w formacie Unix Epoch



Rysunek 6.1: Schemat blokowy oprogramowania urządzenia pokojowego.

- *double offline_temp* - zmienna przechowująca ustawioną temperaturę - parametr, według którego podejmowana jest decyzja o zamknięciu lub otwarciu zaworu grzejnika pokojowego w razie utraty połączenia z urządzeniem centralnym (wartość zmiennej *alive* jest równa 0), jej wartość jest synchronizowana z wartością ustawionej na urządzeniu centralnym, według harmonogramu, temperatury dla pomieszczenia, w którym znajduje się urządzenie

- *double offline_hist* - zmienna przechowująca ustawioną histerezę - parametr, według którego podejmowana jest decyzja o zamknięciu lub otwarciu zaworu grzejnika pokojowego w razie utraty połączenia z urządzeniem centralnym (wartość zmiennej *alive* jest równa 0), jej wartość jest synchronizowana z wartością ustawionej na urządzeniu centralnym histerezy

6.1.2 Opis ważnych procedur, funkcji oraz fragmentów kodu

- *void open()* - otwiera zawór grzejnika pokojowego poprzez wystawienie na odpowiednie wyjścia mikrokotrolera połączone z mostkiem H odpowiednich stanów logicznych (niskiego i wysokiego), ustawia zmienną opened na 1, uruchamia odliczanie awaryjnego wyłączenia silnika na 4 sekundy oraz powoduje zatrzymanie wykonywania przez mikrokontroler jakichkolwiek operacji przez 500ms (wywołanie funkcji *delay()*) w celu uniknięcia błędnego wykrycia przeciążenia związanego z wprawianiem silnika w ruch z przeciążeniem wynikłym z maksymalnego otwarcia bądź zamknięcia zaworu przez ten silnik.

- *void close()* - zamyka zawór grzejnika pokojowego poprzez wystawienie na odpowiednie wyjścia mikrokotrolera połączone z mostkiem H (te same co w procedurze open) odpowiednich stanów logicznych (wysokiego i niskiego), ustawia zmienną opened na 0, uruchamia odliczanie awaryjnego wyłączenia silnika na 5 sekundy oraz powoduje zatrzymanie wykonywania przez mikrokontroler jakichkolwiek operacji przez 500ms (wywołanie funkcji *delay()*) w celu uniknięcia błędnego wykrycia przeciążenia związanego z wprawianiem silnika w ruch jako przeciążenia wynikłego z maksymalnego otwarcia bądź zamknięcia zaworu przez ten silnik.

- *String generatePayloadString()* - generuje łańcuch znaków zawierający zako-

dowany w formacie JSON (notacja obiektów) obiekt z polami przechowującymi:

  - id urządzenia (jego adres MAC)
  - adres IP (protokołu internetowego) urządzenia
  - średnią wartość z trzech ostatnich pomiarów temperatury
  - znacznik czasowy ostatniego pomiaru
  - stan zaworu

- *void callback(char* topic, byte* payload, unsigned int length)* - wywoływana w chwili otrzymania wiadomości MQTT w celu jej przetworzenia (urządzenie centralne przesyła do urządzenia pokojowego 4 typy informacji: żądanie otwarcia zaworu - łańcuch znaków *open*; żądanie zamknięcia zaworu - łańcuch znaków *close*; tzw. bicie serca - łańcuch znaków w formacie *heartbeat:<ustawiona_temperatura>;<ustawiona_histereza>*; wymuszenie przesłania informacji o stanie urządzenia - łańcuch znaków *force*). Jeżeli w wiadomości znajduje się żądanie otwarcia zaworu, funkcja wywołuje procedurę *open()*, jeżeli żądanie zamknięcia zaworu - procedurę *close()*. Następnie w obydwu tych przypadkach publikuje informację o stanie urządzenia (łańcuch znaków generowany funkcją *generatePayloadString()*). W przypadku gdy treścią wiadomości jest bicie serca urządzenia centralnego, resetuje wartość zmiennej *alive* (ustawia ją na 5) oraz aktualizuje wartość zmiennych *offline_temp* i *offline_hist*. W przypadku gdy wartość tych zmiennych uległa zmianie względem ostatniej wiadomości tego typu, ich wartość jest również aktualizowana w pamięci EEPROM. W przypadku jakiejkolwiek innej wiadomości publikowana jest informacja o stanie urządzenia (łańcuch znaków generowany funkcją *generatePayloadString()*).

- Fragment kodu odpowiedzialny za zatrzymanie silnika po maksymalnym otwarciu/zamknięciu zaworu grzejnika pokojowego (przedstawiony na rysunku 6.2) - zatrzymuje pracę silnika poprzez wystawienie na odpowiednie wyjścia mikokontrolera stanów niskich gdy na odpowiednim wejściu mikokontrolera pojawi się stan niski (wykryto przeciążenie), lub upłynie odliczanie awaryjnego wyłączenia silnika.

```
if(!digitalRead(DETECT) || stopb)
  {
    stop.detach();
    digitalWrite(OPEN, LOW);
    digitalWrite(CLOSE, LOW);
    stopb = 0;
  }
```

Rysunek 6.2: Fragment kodu oprogramowania urządzenia pokojowego odpowiedzialny za zatrzymanie pracy silnika.

- *void handle_home_page()* - obsługuje zapytania WWW strony głównej zawierającej formularz ustawień parametrów urządzenia oraz odnośnik do podstrony aktualizacji oprogramowania.
- *void handle_settings()* - obsługuje zapytania WWW pochodzące z przesłania formularza ustawień ze strony głównej, zapisuje podane przez użytkownika parametry do pamięci EEPROM i resetuje urządzenie.
- Fragment kodu odpowiedzialny za pomiar temperatury, umieszczenie wyniku w tablicy *temps*, obliczenie średniej wartości z wyników trzech ostatnich pomiarów oraz publikujący wiadomość MQTT o stanie urządzenia (przedstawiony na rysunku 6.3).

```
    sensors.requestTemperatures();
    timeStamp = String(UTC.now());
    temps[tempIndex] = sensors.getTempCByIndex(0);
    lastThreeAvgTemp = 0;
    for(int i = 0; i<3; ++i) {
      lastThreeAvgTemp += temps[i];
    }
    lastThreeAvgTemp = lastThreeAvgTemp/3;
    payloadtosend = generatePayloadString();
    mqtt.publish(mqttt.c_str(), payloadtosend.c_str());
    tempIndex = (tempIndex+1)\%3;
```

Rysunek 6.3: Fragment kodu oprogramowania urządzenia pokojowego implementujący zachowanie urządzenia w razie utraty połączenia z urządzeniem centralnym.

- Fragment kodu implementujący zachowanie urządzenia w razie utraty połączenia z urządzeniem centralnym (przedstawiony na rysunku 6.4) - w razie utraty połączenia z urządzeniem centralnym (wartość zmiennej *alive* = 0) uruchamia hotspot w celu umożliwienia ewentualnej zmiany ustawień urządzenia, próbuje odnowić połączenie z brokerem MQTT (jeżeli zostało zerwane) oraz zamyka lub otwiera zawór w zależności od wyniku trzech ostatnich pomiarów, oraz wartości przechowywanych w zmiennych *offline_temp* oraz *offline_hist*. W przypadku gdy połączenie zostanie odzyskane, wyłącza hotspot. Dekrementuje wartość zmiennej *alive*.

- *void factoryReset()* - implementuje przywrócenie urządzenia do ustawień fabrycznych (wyczyszczenie wykorzystywanego przez oprogramowanie obszaru pamięci EEPROM).

```
1     if(alive > 0){
2       --alive;
3       if(softAP) {
4           WiFi.softAPdisconnect(1);
5           blinking.detach();
6           digitalWrite(LED_BUILTIN, HIGH);
7           softAP = 0;
8         }
9     } else {
10      if(!softAP){
11          WiFi.softAP(mac);
12          softAP = 1;
13        }
14      blinking.attach(0.5, blinkingcb);
15      if(mqtt.state() != 0)
16      {
17        blinking.attach(0.2, blinkingcb);
18        mqtt.connect(macchar, mqttu.c_str(), mqttp.c_str())
              ;
19        sub = mqtt.subscribe(macchar);
20      } else if(!sub) sub = mqtt.subscribe(macchar);
21
22      if(opened && lastThreeAvgTemp > (offline_temp +
           offline_hist))
23      {
24        close();
25      } else if(!opened && lastThreeAvgTemp < (offline_temp
            - offline_hist))
26      {
27        open();
28      }
29    }
30  }
```

Rysunek 6.4: Fragment kodu oprogramowania urządzenia pokojowego implementujący zachowanie urządzenia w razie utraty połączenia z urządzeniem centralnym.

6.2 Oprogramowanie urządzenia sterującego pracą pieca

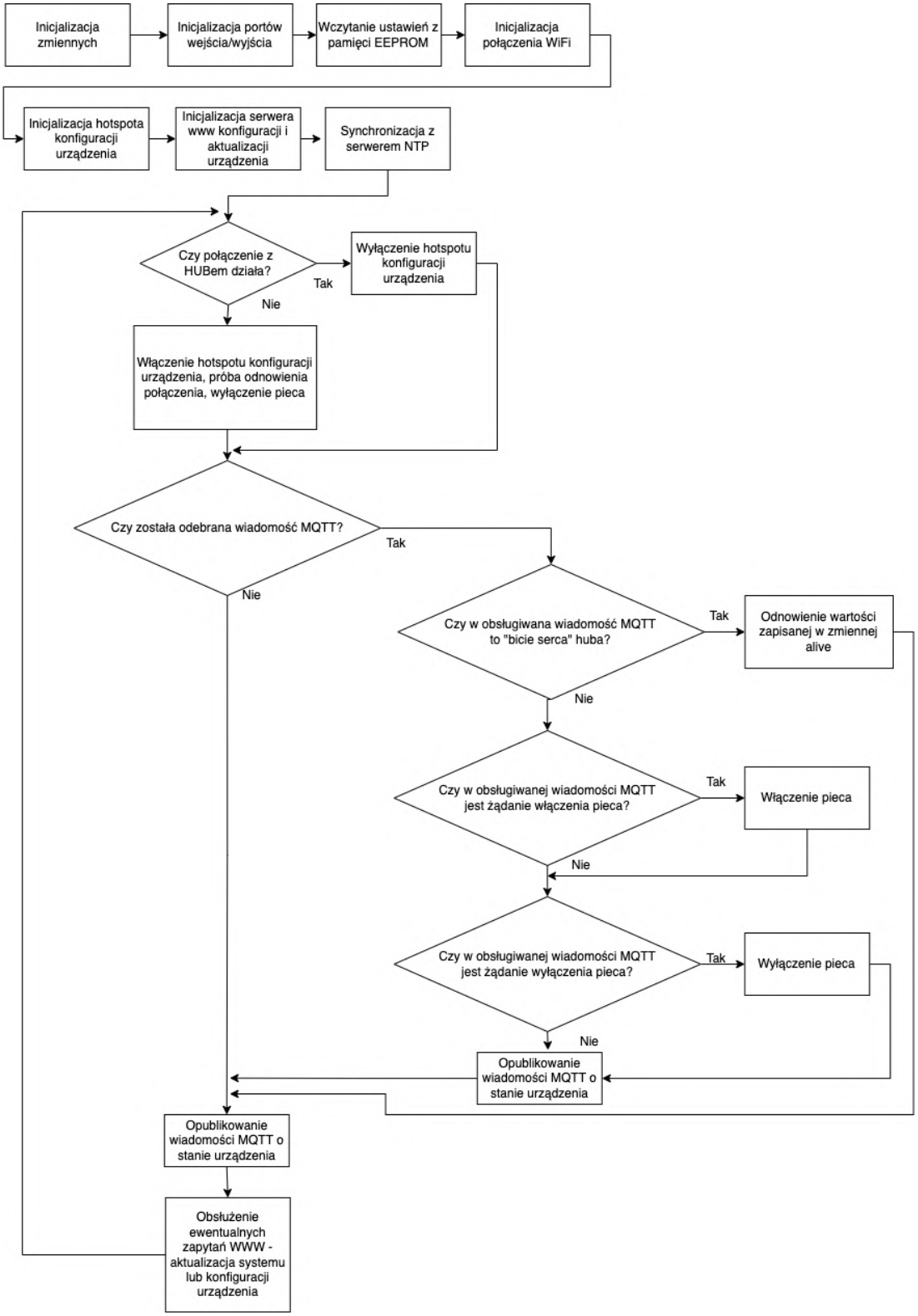
Schemat blokowy oprogramowania urządzenia sterującego pracą pieca przedstawiono na rysunku 6.5.

6.2.1 Wykaz istotnych zmiennych

- *bool on* - przechowuje stan pieca (0/false - wyłączony, 1/true - włączony)
- *int alive* - zmienna przechowująca status połączenia z urządzeniem centralnym (zakres od 0-5) - dekrementowana co minutę i ustawiana na wartość 5 w przypadku odebrania tzw. bicia serca (heartbeat) urządzenia centralnego (wysyłanego przez urządzenie centralne również w odstępach minutowych)
- *String timeStamp* - zmienna przechowująca znacznik czasowy ostatniej zmiany stanu pieca (ostatniego włączenia/wyłączenia) w formacie Unix Epoch

6.2.2 Opis ważnych procedur, funkcji oraz fragmentów kodu

- void *turnOn()* - włącza piec poprzez wystawienie na odpowiednie wyjście mikrokotrolera, sterujące przekaźnikiem (przez tranzystor), stanu wysokiego.
- *void turnOff()* - wyłącza piec poprzez wystawienie na odpowiednie wyjście mikrokotrolera, sterujące przekaźnikiem (przez tranzystor), stanu niskiego.
- *String generatePayloadString()* - generuje łańcuch znaków zawierający zakodowany w formacie JSON obiekt z polami przechowującymi:
  - adres IP urządzenia
  - znacznik czasowy ostatniej zmiany stanu pieca
  - stan pieca
- *void callback(char* topic, byte* payload, unsigned int length)* - wywoływana w chwili otrzymania wiadomości MQTT w celu jej przetworzenia (urządzenie



Rysunek 6.5: Schemat blokowy oprogramowania urządzenia sterującego pracą pieca.

centralne przesyła do urządzenia sterującego piecem 4 typy informacji: żądanie włączenia pieca - łańcuch znaków *on*; żądanie wyłączenia pieca - łańcuch znaków *off*; tzw. bicie serca - łańcuch znaków *heartbeat*; wymuszenie przesłania informacji o stanie urządzenia - łańcuch znaków *force*). Jeżeli w wiadomości znajduje się żądanie włączenia pieca, funkcja wywołuje procedurę *turnOn()*, jeżeli żądanie wyłączenia pieca procedurę *turnOff()*. Następnie w obydwu tych przypadkach publikuje informację o stanie urządzenia (łańcuch znaków generowany funkcją *generatePayloadString()*). W przypadku gdy treścią wiadomości jest bicie serca urządzenia centralnego, resetuje wartość zmiennej *alive* (ustawia ją na 5). W przypadku jakiejkolwiek innej wiadomości publikowana jest informacja o stanie urządzenia (łańcuch znaków generowany funkcją *generatePayloadString()*).

- *void factoryReset()* - implementuje przywrócenie urządzenia do ustawień fabrycznych (wyczyszczenie wykorzystywanego przez oprogramowanie obszaru pamięci EEPROM).

- *void handle_home_page()* - obsługuje zapytania WWW strony głównej zawierającej formularz ustawień parametrów urządzenia oraz odnośnik do podstrony aktualizacji oprogramowania.

- *void handle_settings()* - obsługuje zapytania WWW pochodzące z przesłania formularza ustawień ze strony głównej, zapisuje podane przez użytkownika parametry do pamięci EEPROM i resetuje urządzenie.

- Fragment kodu implementujący zachowanie urządzenia w razie utraty połączenia z urządzeniem centralnym (przedstawiony na rysunku 6.6) - w razie utraty połączenia z urządzeniem centralnym (wartość zmiennej $alive = 0$) uruchamia hotspot w celu umożliwienia ewentualnej zmiany ustawień urządzenia, próbuje odnowić połączenie z brokerem MQTT (jeżeli zostało zerwane) oraz wyłącza piec (wywołuje procedurę *turnOff()*). W przypadku gdy połączenie zostanie odzyskane, wyłącza hotspot. Dekrementuje wartość zmiennej *alive*.

```
1  if(alive > 0)
2     {
3       alive -=1;
4       if(softAP) {
5           WiFi.softAPdisconnect(1);
6           blinking.detach();
7           digitalWrite(LED_BUILTIN, !on);
8           softAP = 0;
9         }
10    } else {
11      if(on) turnOff();
12      if(!softAP){
13          WiFi.softAP(mac);
14          softAP = 1;
15        }
16      blinking.attach(0.5, blinkingcb);
17      if(mqtt.state() != 0)
18      {
19        blinking.attach(0.2, blinkingcb);
20        mqtt.connect(mac.c_str(), mqttu.c_str(), mqttp.
              c_str());
21        sub = mqtt.subscribe((mqttt+"/receive").c_str());
22      } else if(!sub) sub = mqtt.subscribe((mqttt+"/receive
           ").c_str());
23    }
```

Rysunek 6.6: Fragment kodu oprogramowania urządzenia sterującego pracą pieca implementujący zachowanie urządzenia w razie utraty połączenia z urządzeniem centralnym.

6.3 Oprogramowanie sterujące całością systemu

6.3.1 Klasy i obiekty

Klasa Device

Obiekty klasy *Device* reprezentują skonfigurowane w systemie urządzenia pokojowe.

Pola klasy *Device*:

- *name* - przechowuje nazwę urządzenia (pomieszczenia)
- *id* - przechowuje id urządzenia (jego adres MAC)
- *temp* - przechowuje temperaturę panującą w pomieszczeniu, w którym znajduje się urządzenie
- *timeStamp* - przechowuje znacznik czasowy ostatniego pomiaru temperatury przez urządzenie
- *schedule* - obiekt przechowujący harmonogram ustawionych temperatur
- *opened* - przechowuje stan zaworu, którym steruje urządzenie (0 - zamknięty, 1 - otwarty)
- *alive* - przechowuje stan połączenia z urządzeniem (w zakresie od 5 do 0, gdzie 0 oznacza utratę połączenia, dekrementowane co minutę, ustawiane na wartość maksymalną przy odebraniu wiadomości MQTT ze stanem urządzenia)

Obiekt *schedule*:

- posiada pola-obiekty reprezentujące każdy dzień tygodnia
- każdy obiekt reprezentujący dzień tygodnia posiada pola:
  - *times* - pole przechwujące tablicę obiektów reprezentujących przedziały czasowe

    - obiekty reprezentujące przedziały czasowe, posiadają pole *end* przechowujące godzinę końca przedziału, oraz pole *temp* przechowujące temperaturę ustawioną dla danego przedziału
    - pierwszy przedział obejmuje okres od godziny 00:00 do godziny przechowywanej w polu *end*, kolejny przedział okres od godziny końca poprzedniego przedziału do godziny przechowywanej w polu *end* itd.
  - *lastTemp* - pole przechowywujące temperaturę dla przedziału czasowego obejmującego okres od godziny ostatniego przedziału zawartego w tablicy obiektów *times* do końca dnia (godziny 23:59).

Klasa newDevice

Obiekty tej klasy reprezentują urządzenia pokojowe oczekujące na konfigurację w systemie.

Pola klasy *newDevice*:

- *id* - przechowuje id urządzenia (jego adres MAC)
- *temp* - przechowuje temperaturę panującą w pomieszczeniu, w którym znajduje się urządzenie
- *timeStamp* - przechowuje znacznik czasowy ostatniego pomiaru temperatury przez urządzenie
- *opened* - przechowuje stan zaworu, którym steruje urządzenie (0 - zamknięty, 1 - otwarty)
- *alive* - przechowuje stan urządzenia (w zakresie od 5 do 0, gdzie 5 oznacza, że niedawno otrzymano wiadomość MQTT od tego urządzenia)

Obiekt options

Obiekt *options* przechowuje podstawowe ustawienia systemu.

Struktura obiektu *options*:

- *hysteresis* - przechowuje wartość histerezy
- *mqttaddress* - przechowuje adres brokera MQTT w formacie:

  mqtt(s)://adres_ip_lub_domena:port
- *mqtttopic* - przechowuje temat urządzenia centralnego systemu
- *mqttuser* - przechowuje nazwę użytkownika umożliwiającą autoryzację podczas podłączania do brokera MQTT
- *mqttpassword* - przechowuje hasło użytkownika umożliwiające autoryzację podczas podłączania do brokera MQTT
- *mqttfurnacetopic* - przechowuje temat MQTT urządzenia sterującego pracą pieca
- *usedb* - przechowuje informację, czy skonfigurowano okresowy zapis stanu systemu do bazy danych
- *influxdb* - przechowuje obiekt zawierający parametry bazy danych
  - *url* - przechowuje adres bazy danych
  - *organisation* - przechowuje nazwę organizacji (parametr bazy danych influxdb)
  - *bucket* - przechowuje nazwę koszyka, w którym umieszczane są dane
  - *key* - przechowuje token autoryzacji do bazy danych
- *day* - przechowuje godzinę początku dnia
- *night* - przechowuje godzinę początku nocy

Obiekt furnace

Obiekt *furnace* reprezentuje piec.

Struktura obiektu *furnace*:

- *on* - przechowuje informację o stanie pieca (0 - wyłączony, 1 - włączony)
- *ip* - przechowuje adres IP urządzenia sterującego pracą pieca
- *alive* - przechowuje stan połączenia z urządzeniem sterującym pracą pieca (w zakresie od 0 do 5, gdzie 0 oznacza utratę połączenia)

6.3.2 Struktury danych

Dane przetwarzane przez program przechowywane są w podstawowych strukturach danych takich jak tablice - w szczególności tablica *devices*, przechowująca obiekty klasy *Device* (skonfigurowane w systemie urządzenia pokojowe), oraz tablica *newdevices*, przechowująca obiekty klasy *newDevice* (urządzenia pokojowe oczekujące na konfigurację) - oraz obiekty (w szczególności obiekt *options* przechowujący konfigurację systemu).

W celu zapewnienia trwałości tych struktur danych (tablica *devices* oraz obiekt *options*) - na wypadek zatrzymania programu - w sytuacji zmiany przez użytkownika parametrów w nich zawartych są one kodowane do formatu JSON i zapisywane w systemie plików (pliki *data/options.json*, *data/devices.json*).

6.3.3 Opis ważnych procedur, funkcji oraz fragmentów kodu

- funkcja *getSetTemp(device)* - zwraca ustawioną według harmonogramu temperaturę dla danego pomieszczenia
- funkcja *howMany()* - zwraca ilość pomieszczeń w których otwarty jest zawór grzejnika pokojowego
- funkcja odpowiedzialna za zapis stanu systemu do bazy danych przedstawiona na rysunku 6.7 - w pierwszej kolejności wysyła do każdego skonfigurowanego w systemie urządzenia pokojowego żądanie publikacji wiadomości zawierającej jego stan, następnie po okresie ok. 10 sekund tworzy dla każdego urządzenia punkt (rekord) zawierający stan zaworu oraz wartość temperatury panującej w pomieszczeniu i umieszcza go w bazie danych. Na koniec tworzy i umieszcza w bazie danych punkt zawierający stan pieca

```
function () {
    //Send "force" to force publication of latest
       measurements
    devices.forEach((device) => {
      client.publish(device.id, "force");
    });

    //Wait for responses
    setTimeout(() => {
      devices.forEach((device) => {
        const tempPoint = new Point("Devices")
          .tag("Name", device.name)
          .tag("ID", device.id)
          .intField("Opened", device.opened)
          .timestamp(device.timeStamp)
          .floatField("Temperature", device.temp);
        writeApi.writePoint(tempPoint);
      });

      const tempPoint = new Point("Furnace")
        .timestamp(furnace.timeStamp)
        .intField("On", furnace.on);
      writeApi.writePoint(tempPoint);
    }, 10000);
  });
```

Rysunek 6.7: Funkcja odpowiedzialna za zapisywanie stanu systemu do bazy danych (oprogramowanie sterujące całością systemu)

- fragment kodu odpowiedzialny za wyłączenie pieca w sytuacji, gdy utracono połączenie ze wszystkimi urządzeniami pokojowymi pomieszczeń w których były otwarte zawory grzejników pokojowych przedstawiono na rysunku 6.8

- fragment kodu odpowiadający za przetwarzanie wiadomości MQTT przedstawiony na rysunku 6.9 - jeśli wiadomość pochodzi od urządzenia pokojowego, wyszukuje, czy w tablicy *devices* znajduje się obiekt reprezentujący urządzenie od którego pochodzi przetwarzana wiadomość MQTT. Jeśli tak, aktualizuje pola tego obiektu otrzymanymi danymi, jeśli nie wyszukuje czy

```
1 if (howMany() == 0 && furnace.on) {
2     client.publish(options.mqttfurnacetopic + "/receive", "
          off");
3     furnace.on = 0;
4  }
```

Rysunek 6.8: Fragment kodu oprogramowania sterującego całością systemu odpowiedzialny za wyłączenie pieca w sytuacji utraty połączenia z urządzeniami pokojowymi.

obiekt reprezentujący to urządzenie znajduje się w tablicy *newdevices*. Jeśli tak, aktualizuje pola tego obiektu otrzymanymi danymi, jeśli nie tworzy obiekt klasy *NewDevice* i umieszcza go w tablicy *newdevices*. Jeśli otrzymana wiadomość pochodzi od innego urządzenia, oznacza to, że została wysłana przez urządzenie sterujące pracą pieca - wówczas aktualizowany jest obiekt reprezentujący piec. Źródło wiadomości jest określane na podstawie tematu z którego pochodzi - urządzenia pokojowe publikują wiadomości z tematem zawartym w polu *mqttthread* obiektu *options*, natomiast piec z podtematem *send* tematu zawartego w polu *mqttfurnacetopic* tego samego obiektu.

- fragment kodu odpowiedzialny za podejmowanie decyzji o konieczności otwarcia bądź zamknięcia zaworu grzejnika pokojowego w którymś pomieszczeniu oraz włączeniu lub wyłączeniu pieca przedstawiony na rysunku 6.10 - decyzje są podejmowane zgodnie ze schematem blokowym przedstawionym na rysunku 3.11 - kiedy piec jest włączony wysyła żądanie otwarcia zaworu grzejnika pokojowego do każdego pomieszczenia w którym nie jest on jeszcze otwarty, a temperatura jest mniejsza niż ustawiona temperatura. Jeżeli temperatura w danym pomieszczeniu jest większa niż ustawiona powiększona o histerezę, a zawór jest otwarty - wysyła żądanie zamknięcia zaworu. W przypadku gdy pomieszczenie które zostało dogrzane do tej temperatury jest ostatnim urządzeniem z otwartym zaworem, zamiast żądania zamknięcia w nim zaworu wysyła żądanie wyłączenia pieca. W przypadku gdy piec jest wyłączony, a temperatura w którymś pomieszczeniu spadnie poniżej ustawionej pomniejszonej o histerezę wysyła żądanie otwarcia w tym pomieszczeniu zaworu oraz żądanie włączenia pieca.

```
client.on("message", (topic, message) => {
  if (topic == options.mqtttopic) {
    const received = JSON.parse(message.toString());
    const current = devices.find((c) => c.id === received.
       id);
    if (!current) {
      const current = newdevices.find((c) => c.id ===
         received.id);
      if (!current) {
        newdevices.push(new NewDevice(received));
      } else {
        current.ip = received.ip;
        current.temp = received.temp;
        current.opened = received.opened;
        current.timeStamp = received.timestamp;
        current.alive = 5;
      }
    } else {
      current.ip = received.ip;
      current.temp = received.temp;
      current.opened = received.opened;
      current.timeStamp = received.timestamp;
      current.alive = 5;
    }
  } else {
    const received = JSON.parse(message.toString());
    furnace.on = parseInt(received.on);
    furnace.ip = received.ip;
    furnace.timeStamp = received.timestamp;
    furnace.alive = 5;
  }
});
```

Rysunek 6.9: Fragment kodu oprogramowania sterującego całością systemu odpowiedzialny za przetwarzanie wiadomości MQTT.

```
settemp = getSetTemp(devices[i]);
if (furnace.on) {
  if (
    parseFloat(devices[i].temp) < parseFloat(settemp) &&
    devices[i].opened == 0
  ) {
    client.publish(devices[i].id, "open");
    devices[i].opened = 1;
  }
  else if (
    parseFloat(devices[i].temp) >
    parseFloat(settemp) + parseFloat(options.hysteresis) &&
    devices[i].opened == 1
  ) {
    if (howMany() == 1) {
      client.publish(options.mqttfurnacetopic + "/receive",
          "off");
      furnace.on = 0;
    } else {
      client.publish(devices[i].id, "close");
      devices[i].opened = 0;
    }
  }
} else {
  if (
    parseFloat(devices[i].temp) <
    parseFloat(settemp) - parseFloat(options.hysteresis)
  ) {
    if (devices[i].opened == 0) {
      client.publish(devices[i].id, "open");
    }
    client.publish(options.mqttfurnacetopic + "/receive", "
       on");
  }
}
```

Rysunek 6.10: Fragment kodu odpowiedzialny za podejmowanie decyzji o konieczności otwarcia bądź zamknięcia zaworu grzejnika pokojowego w którymś pomieszczeniu oraz włączeniu lub wyłączeniu pieca (oprogramowanie sterujące całością systemu)

6.3.4 API

Interfejs REST API w celu odczytania, utworzenia, modyfikacji lub usunięcia rekordów w zasobie wykorzystuje żądania HTTP [13]. Najczęściej w celu pobrania danych wykorzystuje się żądanie GET, w celu utworzenia rekordu żądania POST, w celu modyfikacji - PUT, natomiast w celu usunięcia - żądania DELETE.

API omawianego w tym podrozdziale oprogramowania musi umożliwiać:

- pobranie oraz modyfikację podstawowych ustawień systemu
- pobranie oczekujących na konfigurację w systemie urządzeń i ich parametrów
- pobranie urządzeń skonfigurowanych w systemie oraz ich parametrów, modyfikację tych parametrów, dodanie nowego urządzenia i jego parametrów do systemu oraz usunięcie urządzenia wraz z jego konfiguracją z systemu
- pobranie aktualnego stanu pieca

Zasób *`/options`*

- Żądanie GET

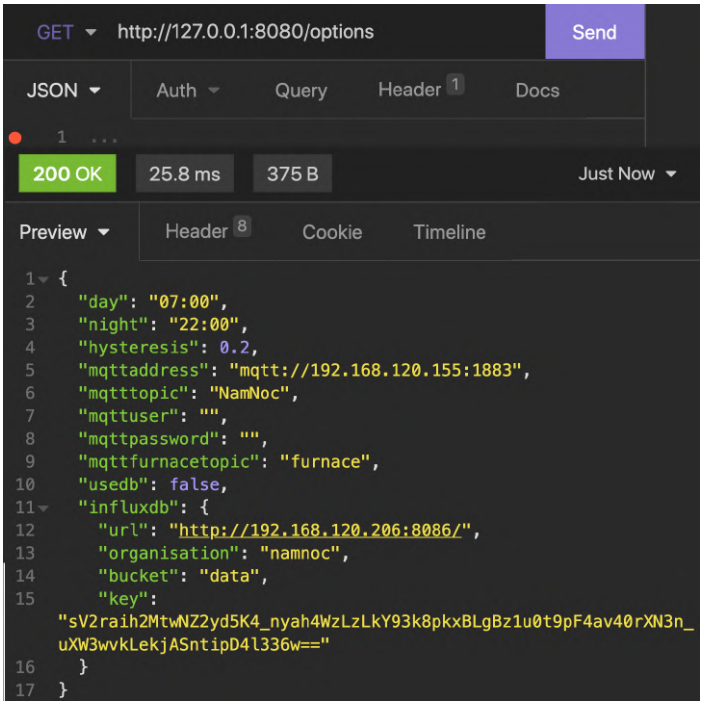
  Ciało żądania: puste

  Odpowiedź: Obiekt zakodowany w formacie JSON zawierający podstawowe ustawienia systemu (struktura obiektu - patrz rysunek 6.11).

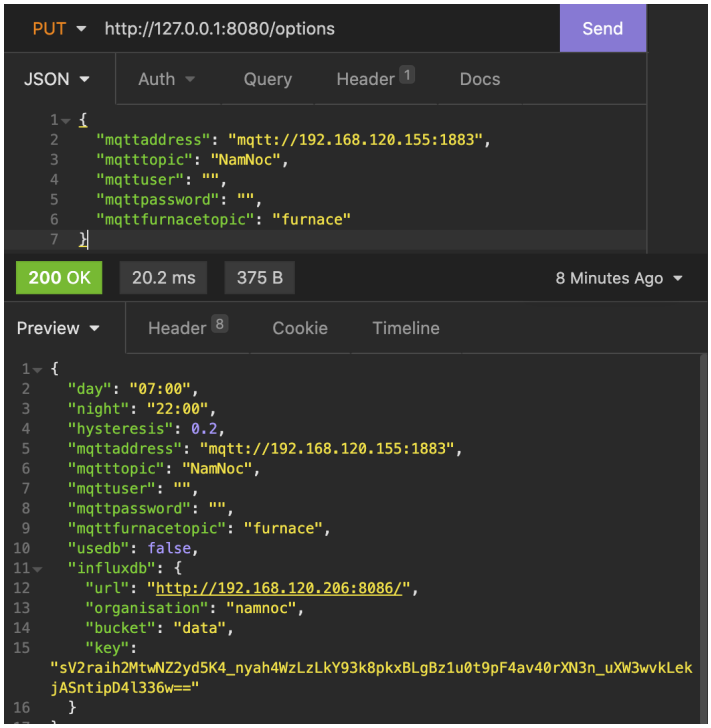
- Żądanie PUT

  Ciało żądania: obiekt zakodowany w formacie JSON zawierający modyfikowane pola obiektu reprezentującego podstawowe ustawienia systemu (nazwy pól powinny być identyczne jak w przypadku obiektu zwracanego w odpowiedzi na żądanie GET - patrz obrazek 6.11)

  Odpowiedź: Obiekt zakodowany w formacje JSON zawierający podstawowe ustawienia systemu po modyfikacji (patrz rysunek 6.12).



Rysunek 6.11: Wykonanie żądania GET do API dla zasobu /options.



Rysunek 6.12: Wykonanie żądania PUT do API dla zasobu /options.

Zasób */devices*

- Żądanie GET

  Ciało żądania: puste

  Odpowiedź: Tablica zakodowana w formacie JSON zawierająca obiekty reprezentujące skonfigurowane w systemie urządzenia pokojowe (patrz rysunek 6.13).

- Żądanie DELETE

  Ciało żądania: Obiekt zakodowany w formacie JSON z polem *id* zawierającym id usuwanego urządzenia (patrz rysunek 6.14).

  Odpowiedź: Obiekt zakodowany w formacie JSON reprezentujący urządzenie usunięte z systemu (patrz rysunek 6.14).

- Żądanie POST

  Ciało żądania: Obiekt zakodowany w formacie JSON reprezentujący dodawane do systemu urządzenie pokojowe (patrz rysunek 6.15).

  Odpowiedź: Obiekt zakodowany w formacie JSON reprezentujący urządzenie dodane do systemu (patrz rysunek 6.15).

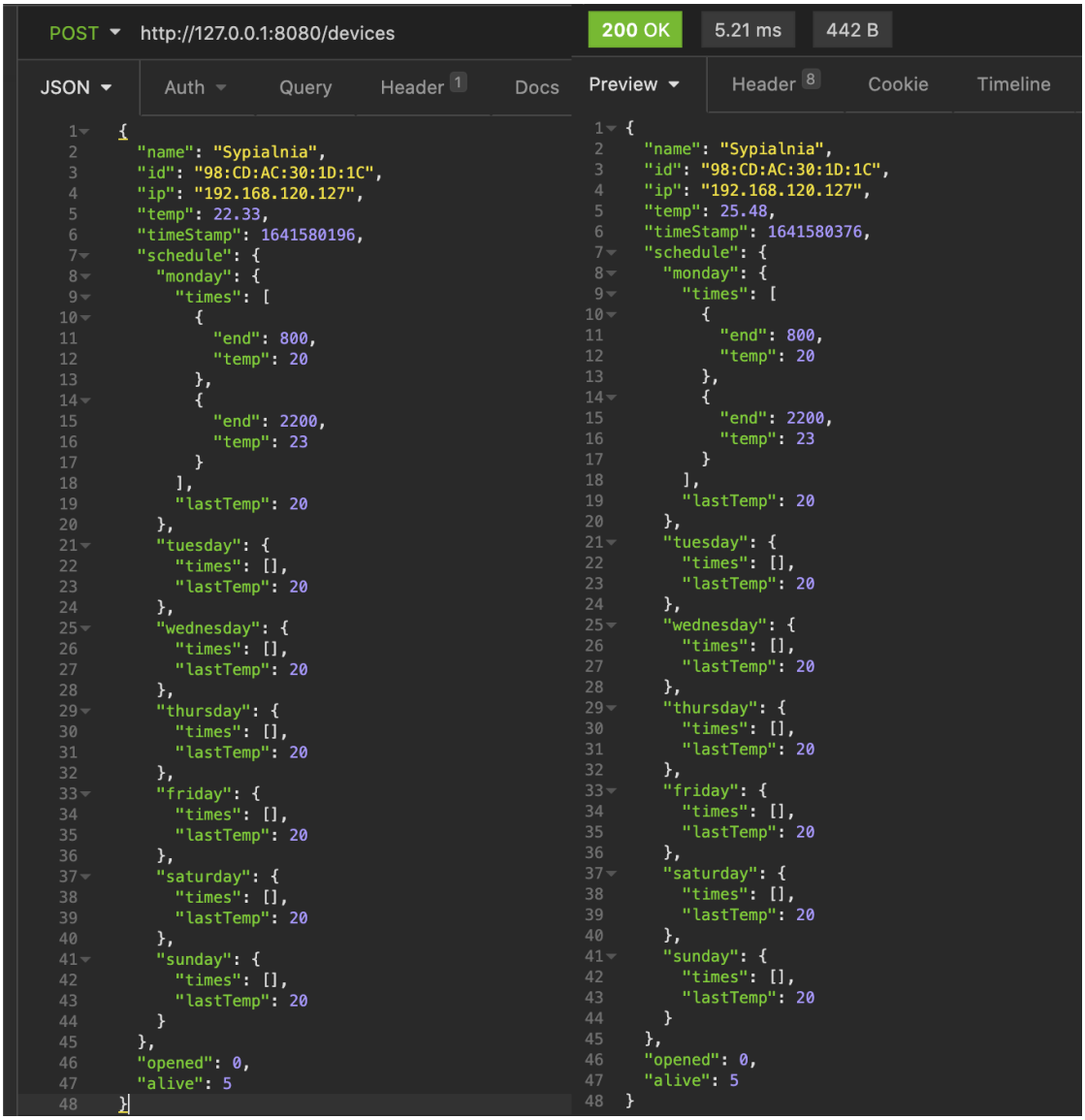
- Żądanie PUT

  Ciało żądania: Obiekt zakodowany w formacie JSON z polem *id* zawierającym id modyfikowanego urządzenia, polem *name* zawierającym nową nazwę urządzenie oraz obiektem *schedule* przechowującym nowy harmonogram (patrz rysunek 6.16).

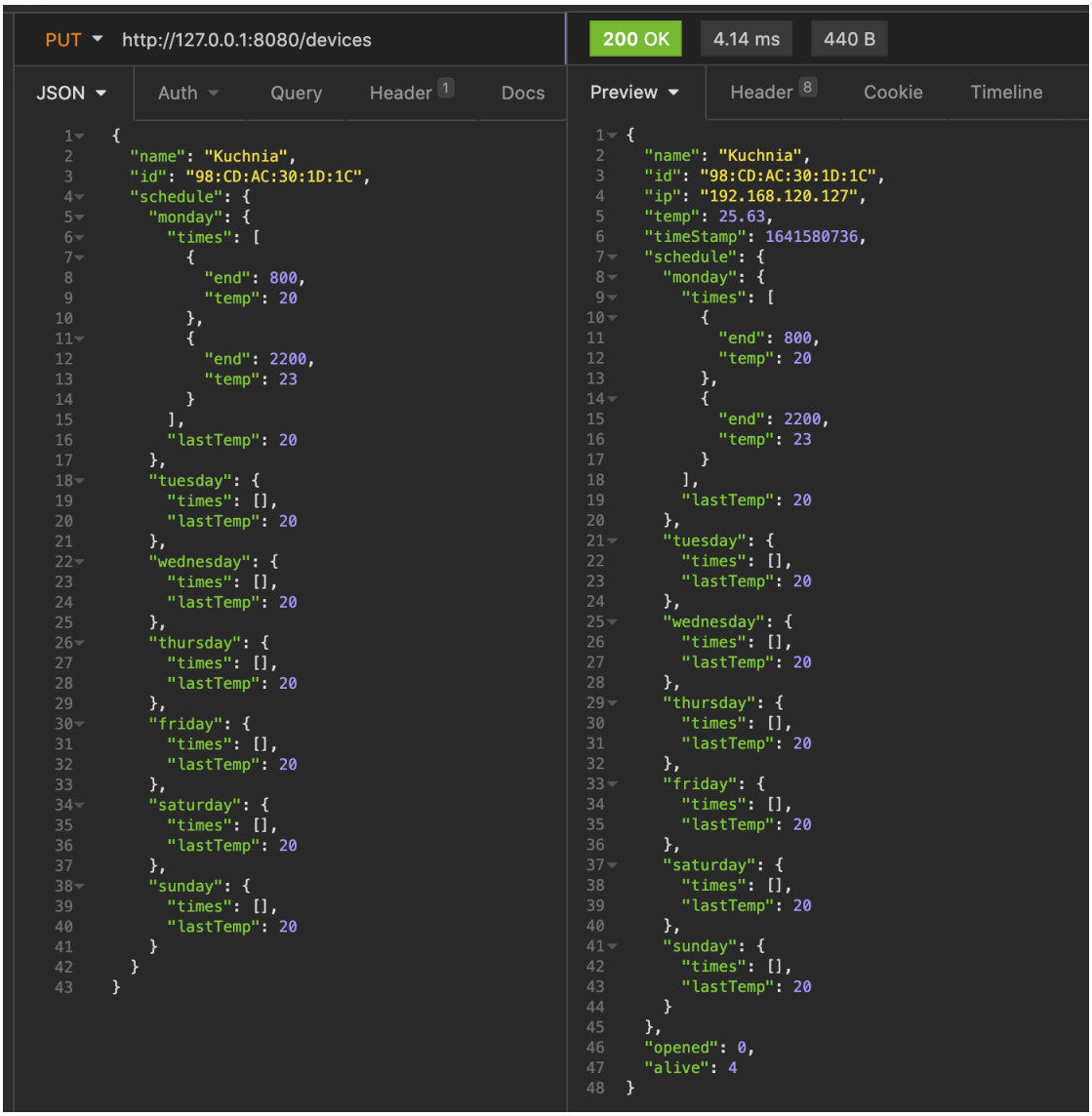
  Odpowiedź: Obiekt zakodowany w formacie JSON reprezentujący zmodyfikowane urządzenie (patrz rysunek 6.16).

Rysunek 6.13: Wykonanie żądania GET do API dla zasobu /devices.

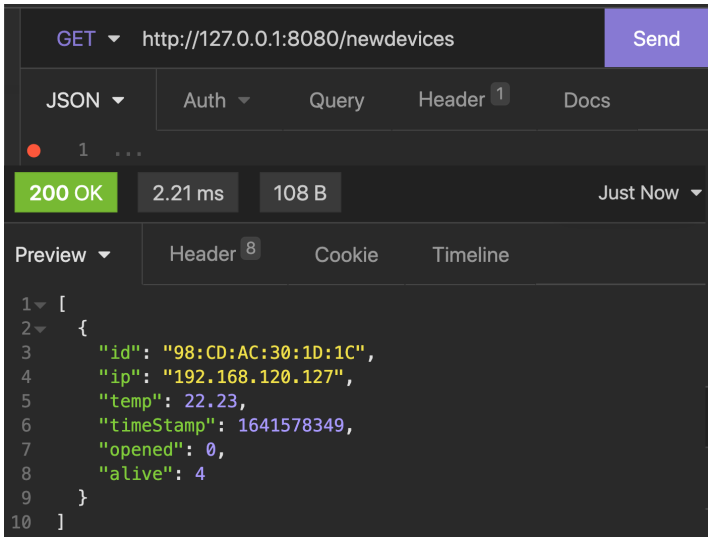
Rysunek 6.14: Wykonanie żądania DELETE do API dla zasobu /devices.



Rysunek 6.15: Wykonanie żądania POST do API dla zasobu /devices.



Rysunek 6.16: Wykonanie żądania PUT do API dla zasobu /devices.



Rysunek 6.17: Wykonanie żądania GET do API dla zasobu /newdevices.

Zasób */newdevices*

- Żądanie GET

  Ciało żądania: puste

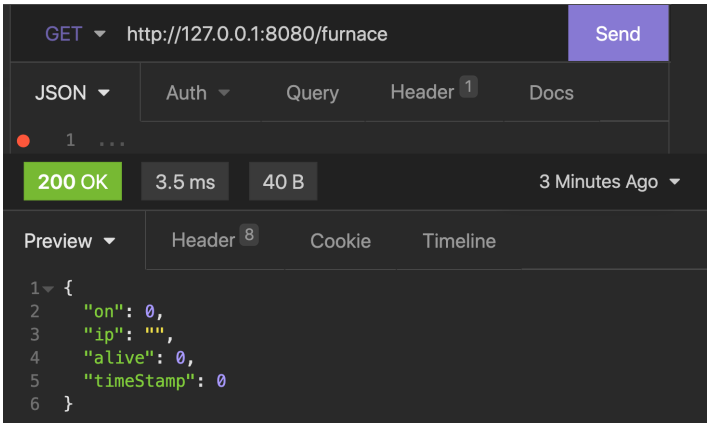
  Odpowiedź: Tablica zakodowana w formacie JSON zawierająca obiekty reprezentujące urządzenia pokojowe oczekujące na konfigurację (patrz rysunek 6.17).

Zasób */furnace*

- Żądanie GET

  Ciało żądania: puste

  Odpowiedź: Obiekt zakodowany w formacie JSON zawierający parametry stanu pieca (struktura obiektu - patrz rysunek 6.18).



Rysunek 6.18: Wykonanie żądania GET do API dla zasobu /furnace.

Zasób */options/set*

- Żądanie GET

  Ciało żądania: puste

  Odpowiedź: *true* - jeśli wczytano ustawienia z pliku, *false* - jeśli wczytano ustawienia domyślne.

Zasób */mqttConnected*

- Żądanie GET

  Ciało żądania: puste

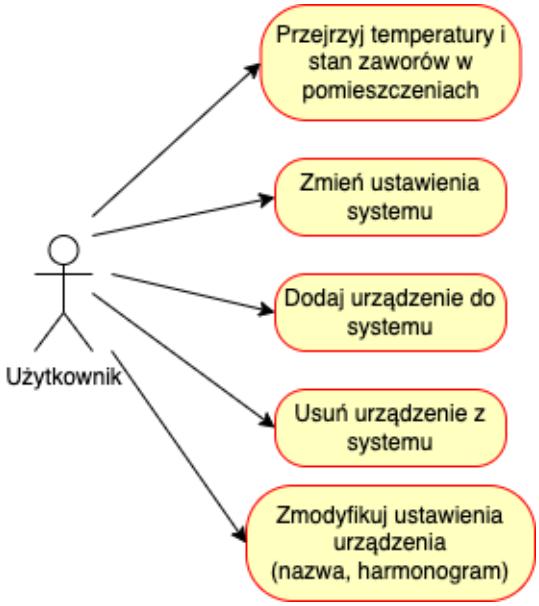
  Odpowiedź: *true* - jeśli połączono do brokera MQTT, *flase* - jeśli brak połączenia do brokera MQTT.

Zasób */time*

- Żądanie GET

  Ciało żądania: puste

  Odpowiedź: Obiekt zakodowany w formacie JSON zawierający czas systemu - pole *time* zawierające aktualną godzinę w formacie HH:MM oraz pole *isNight* zawierające wartość *true* jeśli noc lub *false* jeśli dzień.



Rysunek 6.19: Przypadki użycia - interfejs użytkownika.

6.4 Interfejs użytkownika

Interfejs użytkownika powinien umożliwić użytkownikowi zarządzanie systemem, czyli realizację zadań wyszczególnionych w przypadkach użycia (rys. 6.19).

W tym podroździale, przez serwer rozumie się urządzenie centralne (oprogramowanie sterujące całością systemu).

6.4.1 Wykaz istotnych komponentów

- *settings.jsx* - umożliwia zmianę podstawowych ustawień systemu
- *devices.jsx* - wyświetla parametry skonfigurowanych w systemie urządzeń
- *newDevices.jsx* - wyświetla parametry urządzeń oczekujących na konfigurację
- *device.jsx* - umożliwia modyfikację konfiguracji urządzenia
- *newDevice.jsx* - umożliwia dodanie nowego urządzenia do systemu
- *furnace.jsx* - pokazuje stan pieca
- *home.jsx* - implementuje główny ekran interfejsu użytkownika

6.4.2 Opis ważnych metod

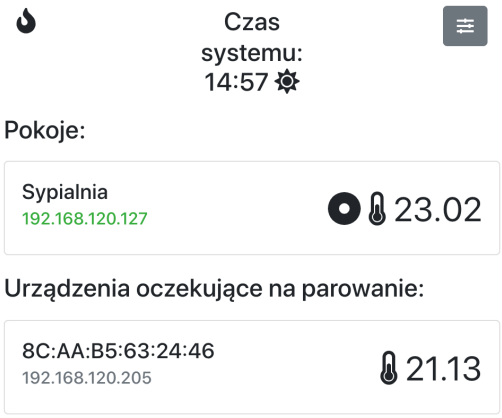
- Komponent *settings.jsx*
  - metoda *getSettings()* - pobiera z serwera podstawowe ustawienia systemu i umieszcza je na ekranie.
  - metoda *submitForm()* - zapisuje podstawowe ustawienia systemu na serwerze
- Komponent *devices.jsx*
  - metoda *getDevices()* - pobiera z serwera skonfigurowane w systemie urządzenia i ich parametry i umieszcza je na ekranie.
- Komponent *newDevices.jsx*
  - metoda *getDevices()* - pobiera z serwera urządzenia oczekujące na konfigurację i umieszcza je na ekranie.
- Komponent *device.jsx*
  - metoda *submitForm()* - modyfikuje konfigurację urządzenia na serwerze
  - metoda *deleteDevice()* - usuwa urządzenie i jego konfigurację z systemu
  - metoda *divide()* - dzieli przedział czasowy w harmonogramie na dwie części
  - metoda *copySchedule()* - kopiuje harmonogram z innego dnia
  - metoda *deleteScheduleEntry()* - usuwa przedział czasowy z harmonogramu
- Komponent *newDevice.jsx*
  - metoda *submitForm()* - dodaje nowe urządzenie do systemu
  - metoda *divide()* - dzieli przedział czasowy w harmonogramie na dwie części
  - metoda *copySchedule()* - kopiuje harmonogram z innego dnia

  - metoda *deleteScheduleEntry()* - usuwa przedział czasowy z harmonogramu

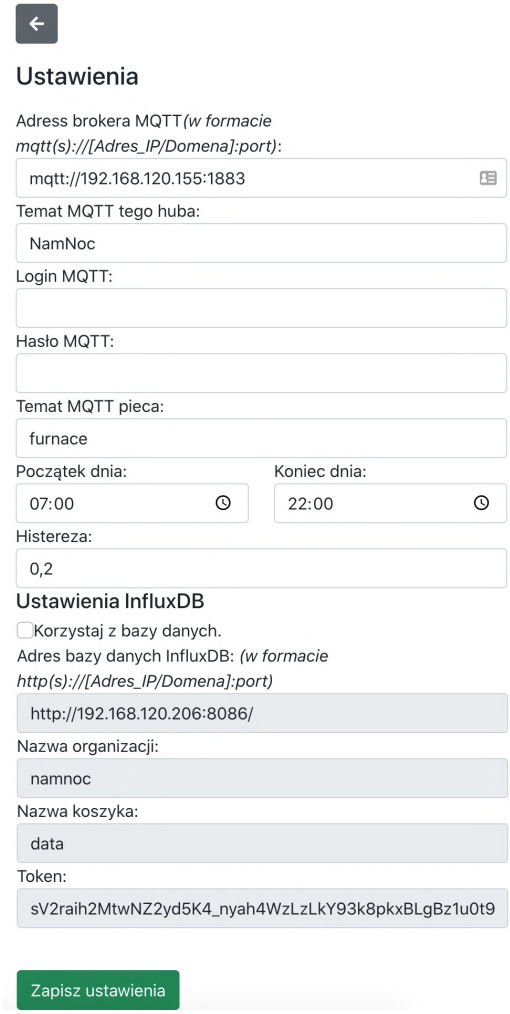
- Komponent *furnace.jsx*
  - metoda *getData()* - pobiera stan pieca z serwera i umieszcza go na ekranie

6.4.3 Główne ekrany interfejsu graficznego

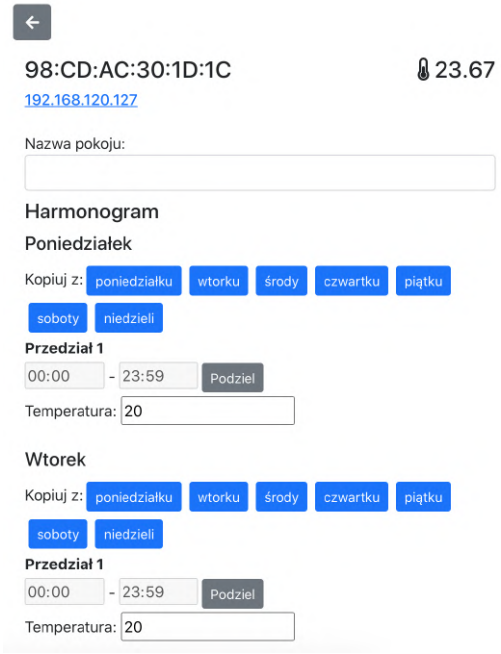
Główne ekrany interfejsu graficznego przedstawiono na rysunkach 6.20, 6.21, 6.22 i 6.23.



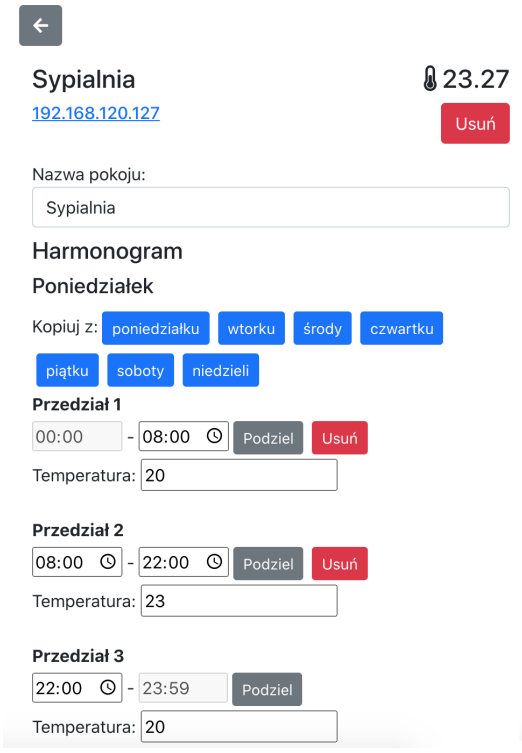
Rysunek 6.20: Interfejs użytkownika - Główny ekran aplikacji



Rysunek 6.21: Interfejs użytkownika - Ekran zmiany podstawowych ustawień systemu



Rysunek 6.22: Interfejs użytkownika - Ekran konfiguracji nowego urządzenia



Rysunek 6.23: Interfejs użytkownika - Ekran zmiany konfiguracji urządzenia

Rozdział 7

Uruchomienie

7.1 Urządzenie pokojowe

Wymagania

- konwerter USB-UART
- Oprogramowanie Tasmotizer w wersji 1.2

Uruchomienie

1. Połącz wyjście RXD konwertera USB-UART z górnym wejściem gniazda typu goldpin urządzenia pokojowego (wejście TXD modułu ESP-12E, patrz rys. 5.1).
2. Połącz wyjście TXD konwertera USB-UART z dolnym wejściem gniazda typu goldpin urządzenia pokojowego (wejście RXD modułu ESP-12E, patrz rys. 5.1).
3. Podłącz konwerter USB-UART do komputera.
4. Podłącz urządzenie pokojowe do zasilania trzymając przyciśnięty przycisk znajdujący się na płytce drukowanej urządzenia (przycisk S1, patrz rys. 5.1). UWAGA! Urządzenie pokojowe musi być zasilone z gniazda USB tego samego komputera co konwerter USB-UART (wspólna masa).

5. Uruchom oprogramowanie Tasmotizer.
6. Wybierz odpowieni port transmisji szeregowej (konwerter USB-UART).
7. Wybierz opcję *Open* w celu wskazania ścieżki do oprogramowania mikrokontrolera urządzenia pokojowego (Urządzenie pokojowe/Oprogramowanie-/firmware.bin).
8. Wybierz opcję *Tasmotize!* w celu umieszczenia programu w pamięci mikrokontrolera urządzenia pokojowego.
9. Odłącz urządzenie pokojowe od zasilania oraz konwertera USB-UART.
10. Podłącz urządzenie pokojowe do zasilania. Jeżeli proces ładowania oprogramowania do pamięci mikrokontrolera przebiegł pomyślnie, wbudowana w moduł ESP-12E dioda powinna mrugać. W przypadku gdy oprogramowanie zostało wgrane na dany moduł ESP-12E po raz pierwszy, zaleca się przywrócenie ustawień fabrycznych urządzenia - przytrzymaj przycisk aż wbudowana w moduł ESP-12E dioda zacznie świecić światłem ciągłym.

Ustawienie napięcia dla silnika z przekładnią

W celu zmiany napięcia wyjściowego przetwornicy step-up (zasilającej silnik przyłączony do gniazda J1 - patrz rys. 5.1) należy kręcić śrubką potencjometru umieszczonego na module tej przetwornicy (przetwornica HW-183 - patrz rys. 5.1) w jedną lub drugą stronę. Napięcie wyjściowe przetwornicy można mierzyć pomiędzy wyjściem $OUT+$ tego modułu a masą urządzenia pokojowego (np. wyjście $OUT-$ przetwornicy step-up). Należy uwzględnić spadek napięcia na mostku H wynoszący ok. 2V (zatem należy ustawić napięcie wyższe o około 2V). Zalecanym maksymalnym napięciem na wyjściu przetwornicy jest wartość nieprzekraczająca 26V. Zmiany napięcia należy dokonywać z silnikiem odłączonym od gniazda J1.

Kalibracja układu detekcji przeciążeń

W celu zwiększenia lub zmniejszenia czułości układu detekcji przeciążeń należy zmienić nastawienie potencjometru VR1 (patrz rys. 5.1) umieszczonego bezpośrednio na płytce drukowanej urządzenia pokojowego. W procesie kalibracji pomocne może okazać się wysyłanie spreparowanych wiadomości MQTT (temat tożsamy z adresem MAC urządzenia którego układ detekcji przeciążeń jest kalibrowany) - *open* i *close* (żądanie otwarcia/zamknięcia zaworu). Tak spreparowane wiadomości można wysyłać np. z poziomu oprogramowania NodeRED, urządzenie pokojowe musi być oczywiście podłączone do brokera MQTT.

7.2 Urządzenie sterujące pracą pieca

Wymagania

- Oprogramowanie Tasmotizer w wersji 1.2

Uruchomienie

1. Podłącz urządzenie sterujące pracą pieca przewodem USB do komputera (gniazdo microUSB modułu NodeMCU)

2. Uruchom oprogramowanie Tasmotizer.

3. Wybierz odpowieni port transmisji szeregowej (wbudowany na płytce NodeMCU konwerter USB-UART).

4. Wybierz opcję *Open* w celu wskazania ścieżki do oprogramowania mikrokontrolera urządzenia sterującego pracą pieca (Urządzenie sterujące pracą pieca/Oprogramowanie/firmware.bin).

5. Wybierz opcję *Self-resetting device (NodeMCU, Wemos).*

6. Wybierz opcję *Tasmotize!* w celu umieszczenia programu w pamieci mikrokontrolera.

7. Urządzenie uruchomi się ponownie, jeśli proces przebiegł pomyślnie, dioda wbudowana w moduł ESP-12E modułu NodeMCU powinna mrugać. W przypadku gdy oprogramowanie zostało wgrane na dany moduł NodeMCU po raz pierwszy, zaleca się przywrócenie ustawień fabrycznych urządzenia - przytrzymaj przycisk na płytce modułu NodeMCU opisany jako *FLASH* aż do momentu gdy wspomniana wcześniej dioda zacznie świecić światłem ciągłym.

7.3 Urządzenie centralne

Wymagania (na urządzeniu pełniącym rolę urządzenia centralnego)

- Środowisko uruchomieniowe *node* w wersji 14.16.0
- Manager pakietów *npm*
- Zainstalowany globalnie pakiet *serve* (*https://www.npmjs.com/package/serve*)
- Oprogramowanie brokera MQTT (np. mosquitto)

Uruchomienie

1. Przenieś zawartość katalogu *Urządzenie centralne/Production* do dowolnego katalogu (urządzenia pełniącego rolę urzadzenia centralnego).
2. Przejdź do podkatalogu *server* i zainstaluj wymagane zależności poleceniem:

   npm install
3. Powróć do katalogu nadrzędnego. Przejdź do podkatalogu *App*.
4. Otwórz plik *config.json* w dowolnym edytorze tekstowym.
5. Zmodyfikuj zawartość pola *hubIp* umieszczając w nim adres IP urządzenia centralnego w sieci lokalnej.
6. Powróć do katalogu nadrzędnego.

7. Uruchom oprogramowanie sterujące całością systemu, oraz serwer WWW interfejsu użytkownika poleceniem:

   node start.js

8. Jeśli oprogramowanie zostało uruchomione poprawnie, interfejs użytkownika powinien być dostępny z poziomu dowolnego urządzenia w sieci lokalnej pod adresem:

   http://<adres-ip-urządzenia-centralnego>:5000

   Interfejs API powinien być dostępny na porcie 8080.

Rozdział 8

Instrukcja obsługi

8.1 Urządzenie pokojowe

8.1.1 Wstępna konfiguracja

1. Podłącz urządzenie do zasilania. Niebieska dioda powinna szybko mrugać.
2. Na dowolnym urządzeniu z przeglądarką internetową oraz możliwością komunikacji WiFi przejdź do ustawień połączenia WiFi. Podłącz to urządzenie do hotspotu WiFi uruchomionego na konfigurowanym urządzeniu pokojowym (nazwą sieci jest adres MAC urządzenia).
3. Otwórz przeglądarkę internetową i przejdź pod adres:

   http://192.168.4.1
4. Wypełnij formularz konfiguracji urządzenia:
   - SSID - nazwa sieci WiFi do której ma się przyłączyć urządzenie
   - WiFi Password - hasło sieci WiFi
   - MQTT Broker address - adres IP brokera MQTT
   - MQTT Broker port - port na którym pracuje broker MQTT (domyślnie *1883*)
   - MQTT NamNoc's Hub Topic - temat MQTT urządzenia centralnego (domyślnie *NamNoc*)

- MQTT Username - nazwa użytkownika do uwierzytelniania połączenia z brokerem MQTT (domyślnie puste)
- MQTT Password - hasło użytkownika do uwierzytelniania połączenia z brokerem MQTT (domyślnie puste)
- Temperature - temperatura według której zawór jest otwierany/zamykany w razie utraty połączenia z urządzeniem centralnym (jest też synchronizowana z urządzeniem centralnym).
- Hysteresis - histereza według której zawór jest otwierany/zamykany w razie utraty połączenia z urządzeniem centralnym (jest też synchronizowana z urządzeniem centralnym).

5. Zapisz ustawienia wybierając opcję *Submit*

6. Urządzenie uruchomi się ponownie. Podczas nawiązywania połączenia do sieci WiFi i brokera MQTT niebieska dioda powinna szybko mrugać. Po nawiązaniu połączenia do sieci WiFi i brokera MQTT dioda powinna zmniejszyć częstotliwość mrugania (powolne mruganie niebieskiej diody oznacza oczekiwanie na konfigurację w systemie/brak połączenia z urządzeniem centralnym).

8.1.2 Zmiana parametrów wstępnej konfiguracji

W przypadku, gdy urządzenie nie nawiąże połączenie do WiFi lub brokera MQTT albo je utraci (niebieska dioda szybko mruga) lub gdy nie ma połączenia z urządzeniem centralnym albo oczekuje na konfigurację w systemie (niebieska dioda powolnie mruga) urządzenie uruchamia hotspot WiFi i możliwa jest zmiana parametrów wstępnej konfiguracji w sposób identyczny jak jej przeprowadzenie.

W przypadku gdy urządzenie ma połączenie z urządzeniem centralnym, formularz zmiany parametrów wstępnej konfiguracji jest dostępny z poziomu przeglądarki pod adresem tożsamym z adresem IP urządzenia, którego parametry wstępnej konfiguracji chcemy zmienić (adres ten można łatwo znaleźć z poziomu interfejsu użytkownika).

8.1.3 Aktualizacja oprogramowania

1. - W przypadku gdy urządzenie ma połączenie z urządzeniem centralnym:
     (a) W dowolnej przeglądarce otwórz interfejs użytkownika systemu.
     (b) Znajdź i wybierz urządzenie z listy.
     (c) Kliknij w niebieski odnośnik (adres IP urządzenia)
   - W przypadku gdy urządzenie nie ma połączenia z urządzeniem centralnym:
     (a) Na dowolnym urządzeniu z przeglądarką internetową oraz możliwością komunikacji WiFi przejdź do ustawień połączenia WiFi. Podłącz to urządzenie do hotspotu WiFi uruchomionego na urządzeniu pokojowym którego oprogramowanie chcesz zaktualizować (nazwą sieci jest adres MAC urządzenia).
     (b) Otwórz przeglądarkę internetową i przejdź pod adres: *http://192.168.4.1*

2. Kliknij w odnośnik *Firmware upgrade*.

3. W sekcji *Firmware* wybierz plik aktualizacji (plik zawierający oprogramowanie urządzenia pokojowego).

4. Wybierz opcję *Update Firmware*.

5. Urządzenie uruchomi się ponownie z nowym oprogramowaniem.

8.1.4 Przywrócenie urządzenia do ustawień fabrycznych

1. Naciśnij i przytrzymaj znajdujący się na płytce drukowanej przycisk aż do momentu gdy niebieska dioda zacznie świecić światłem ciągłym.

2. Po okresie ok. 5 sekund urządzenie uruchomi się ponownie.

8.2 Urządzenie sterujące pracą pieca

8.2.1 Wstępna konfiguracja

1. Podłącz urządzenie do zasilania. Niebieska dioda powinna szybko mrugać.

2. Na dowolnym urządzeniu z przeglądarką internetową oraz możliwością komunikacji WiFi przejdź do ustawień połączenia WiFi. Podłącz to urządzenie do hotspotu WiFi uruchomionego na konfigurowanym urządzeniu (nazwą sieci jest adres MAC urządzenia).

3. Otwórz przeglądarkę internetową i przejdź pod adres:

   http://192.168.4.1

4. Wypełnij formularz konfiguracji urządzenia:

   - SSID - nazwa sieci WiFi do której ma się przyłączyć urządzenie
   - WiFi Password - hasło sieci WiFi
   - MQTT Broker address - adres IP brokera MQTT
   - MQTT Broker port - port na którym pracuje broker MQTT (domyślnie *1883*)
   - MQTT topic - temat MQTT tego urządzenia (domyślnie *furnace*)
   - MQTT Username - nazwa użytkownika do uwierzytelniania połączenia z brokerem MQTT (domyślnie puste)
   - MQTT Password - hasło użytkownika do uwierzytelniania połączenia z brokerem MQTT (domyślnie puste)

5. Zapisz ustawienia wybierając opcję *Submit*

6. Urządzenie uruchomi się ponownie. Podczas nawiązywania połączenia do sieci WiFi i brokera MQTT niebieska dioda powinna szybko mrugać. Po nawiązaniu połączenia do sieci WiFi i brokera MQTT dioda powinna zmniejszyć częstotliwość mrugania (powolne mruganie niebieskiej diody oznacza

brak połączenia z urządzeniem centralnym) a następnie po nawiązaniu połączenia z urządzeniem centralnym zgasnąć (piec wyłączony) lub zaświecić się światłem ciągłym (piec włączony).

8.2.2 Zmiana parametrów wstępnej konfiguracji

W przypadku, gdy urządzenie nie nawiąże połączenie do WiFi lub brokera MQTT albo je utraci (niebieska dioda szybko mruga) lub gdy nie ma połączenia z urządzeniem centralnym (niebieska dioda powolnie mruga) urządzenie uruchamia hotspot WiFi i możliwa jest zmiana parametrów wstępnej konfiguracji w sposób identyczny jak jej przeprowadzenie.

W przypadku gdy urządzenie ma połączenie z urządzeniem centralnym, formularz zmiany parametrów wstępnej konfiguracji jest dostępny z poziomu przeglądarki pod adresem tożsamym z adresem IP urządzenia sterującego pracą pieca.

8.2.3 Aktualizacja oprogramowania

1. - W przypadku gdy urządzenie ma połączenie z urządzeniem centralnym:
     (a) W dowolnej przeglądarce przejdź pod adres tożsamy z adresem IP urządzenia sterującego pracą pieca.
   - W przypadku gdy urządzenie nie ma połączenia z urządzeniem centralnym:
     (a) Na dowolnym urządzeniu z przeglądarką internetową oraz możliwością komunikacji WiFi przejdź do ustawień połączenia WiFi. Podłącz to urządzenie do hotspotu WiFi uruchomionego na urządzeniu sterującym pracą pieca (nazwą sieci jest adres MAC urządzenia).
     (b) Otwórz przeglądarkę internetową i przejdź pod adres: *http://192.168.4.1*

2. Kliknij w odnośnik *Firmware upgrade*.

3. W sekcji *Firmware* wybierz plik aktualizacji (plik zawierający oprogramowanie urządzenia sterującego pracą pieca).

4. Wybierz opcję *Update Firmware*.

5. Urządzenie uruchomi się ponownie z nowym oprogramowaniem.

8.2.4 Przywrócenie urządzenia do ustawień fabrycznych

1. Naciśnij i przytrzymaj znajdujący się na płytce NodeMCU przycisk oznaczony *FLASH* przez okres ok. 15 sekund.

2. Po okresie ok. 5 sekund urządzenie uruchomi się ponownie.

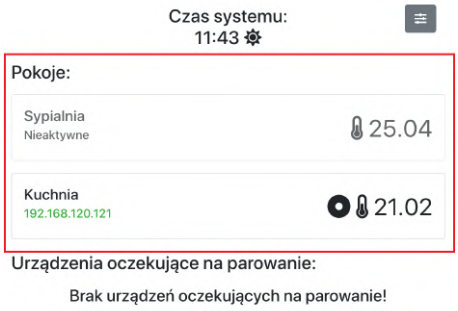
8.3 Elementy interfejsu graficznego

- Sekcja *Pokoje* (rys. 8.1) - W tej sekcji wyświetlana jest lista urządzeń skonfigurowanych w systemie. Jeżeli na liście któreś z urządzeń jest poszarzone, a pod nazwą pomieszczenia zamiast adresu IP widnieje napis *Nieaktywne* oznacza to, że urządzenie utraciło połączenie z urządzeniem centralnym lub jest wyłączone.

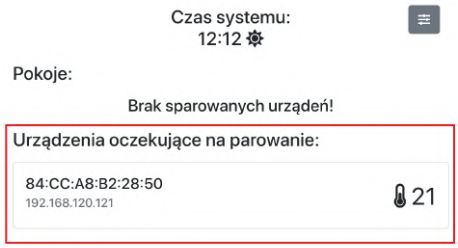
- Sekcja *Urządzenia oczekujące na parowanie* (rys. 8.2) - W tej sekcji wyświetlane są urządzenia które oczekują na konfigurację w systemie.

- Sekcja stanu pieca (rys. 8.3) - w tej sekcji wyświetlony jest stan pieca: czarny płomień - piec włączony; brak symbolu - piec wyłączony; czerwony symbol ostrzegawczy - brak połączenia z urządzeniem sterującym piecem.

- Symbol otwartego zaworu (rys. 8.4) - Symbol wyświetlający się przy wartości zmierzonej w danym pomieszczeniu temperatury, gdy w pomieszczeniu tym zawór grzejnika pokojowego jest otwarty.



Rysunek 8.1: Interfejs użytkownika - Sekcja "Pokoje"



Rysunek 8.2: Interfejs użytkownika - Sekcja "Urządzenia oczekujące na parowanie"

Rysunek 8.3: Interfejs użytkownika - Sekcja stanu pieca

Rysunek 8.4: Interfejs użytkownika - Symbol otwartego zaworu

8.4 Zarządzanie systemem

8.4.1 Zmiana podstawowych ustawień systemu

1. W przeglądarce internetowej na dowolnym urządzeniu w sieci lokalnej otwórz interfejs użytkownika.

2. Przejdź do ekranu podstawowych ustawień systemu klikając w ikonkę ustawień znajdującą się z prawej strony górnego paska ekranu głównego.

3. Wypełnij formularz podstawowej konfiguracji systemu.

   - Adres brokera MQTT - adres brokera MQTT w formacie:
     mqtt(s)://[Adres_IP/Domena]: port
     (domyślnie *mqtt://127.0.0.1:1883*)
   - Temat MQTT tego huba - Temat MQTT urządzenia centralnego systemu (domyślnie *NamNoc*)
   - Login MQTT - nazwa użytkownika do uwierzytelniania połączenia z brokerem MQTT (domyślnie puste)
   - Hasło MQTT - hasło użytkownika do uwierzytelniania połączenia z brokerem MQTT (domyślnie puste)
   - Temat MQTT pieca - temat MQTT urządzenia sterującego pracą pieca (taki sam jak ustawiony przy wstępnej konfiguracji urządzenia sterującego pracą pieca, domyślnie *furnace*)
   - Początek dnia - godzina początku dnia
   - Koniec dnia - godzina początku nocy
   - Ustawienia InfluxDB
     - Korzystaj z bazy danych - wybierz tę opcję, jeśli chcesz aby stan systemu był okresowo zapisywany do bazy danych
     - Adres bazy danych - adres bazy danych w formacie:
       http(s)://[Adres_IP/Domena]: port
     - Nazwa organizacji - nazwa organizacji

- Nazwa koszyka - nazwa koszyka, do którego mają być umieszczane dane
- Token - klucz uwierzytelniający połączenia z bazą danych

4. Zatwierdź zmienę ustawień wybierając opcję *Zapisz ustawienia.*

8.4.2 Dodawanie nowego urządzenia pokojowego do systemu

1. W przeglądarce internetowej na dowolnym urządzeniu w sieci lokalnej otwórz interfejs użytkownika.

2. W sekcji *Urządzenia oczekujące na parowanie* wybierz z listy urządzenie które chcesz dodać do systemu.

3. Wypełnij pole *Nazwa pokoju* oraz ustal harmonogram temperatur.

4. Wybierz opcję *Zapisz ustawienia* w celu dodania urządzenia z jego konfiguracją do systemu.

8.4.3 Usuwanie urządzenia pokojowego z systemu

1. W przeglądarce internetowej na dowolnym urządzeniu w sieci lokalnej otwórz interfejs użytkownika.

2. W sekcji *Pokoje* wybierz z listy urządzenie które chcesz usunąć z systemu.

3. Wybierz opcję *Usuń.*

8.4.4 Zmiana konfiguracji urządzenia pokojowego

1. W przeglądarce internetowej na dowolnym urządzeniu w sieci lokalnej otwórz interfejs użytkownika.

2. W sekcji *Pokoje* wybierz z listy urządzenie którego ustawienia chcesz zmodyfikować.

3. Zmodyfikuj wybrane parametry (nazwa pokoju lub harmonogram temperatur).

4. Wybierz opcję *Zapisz ustawienia* w celu zapisania zmian w systemie.

Rozdział 9

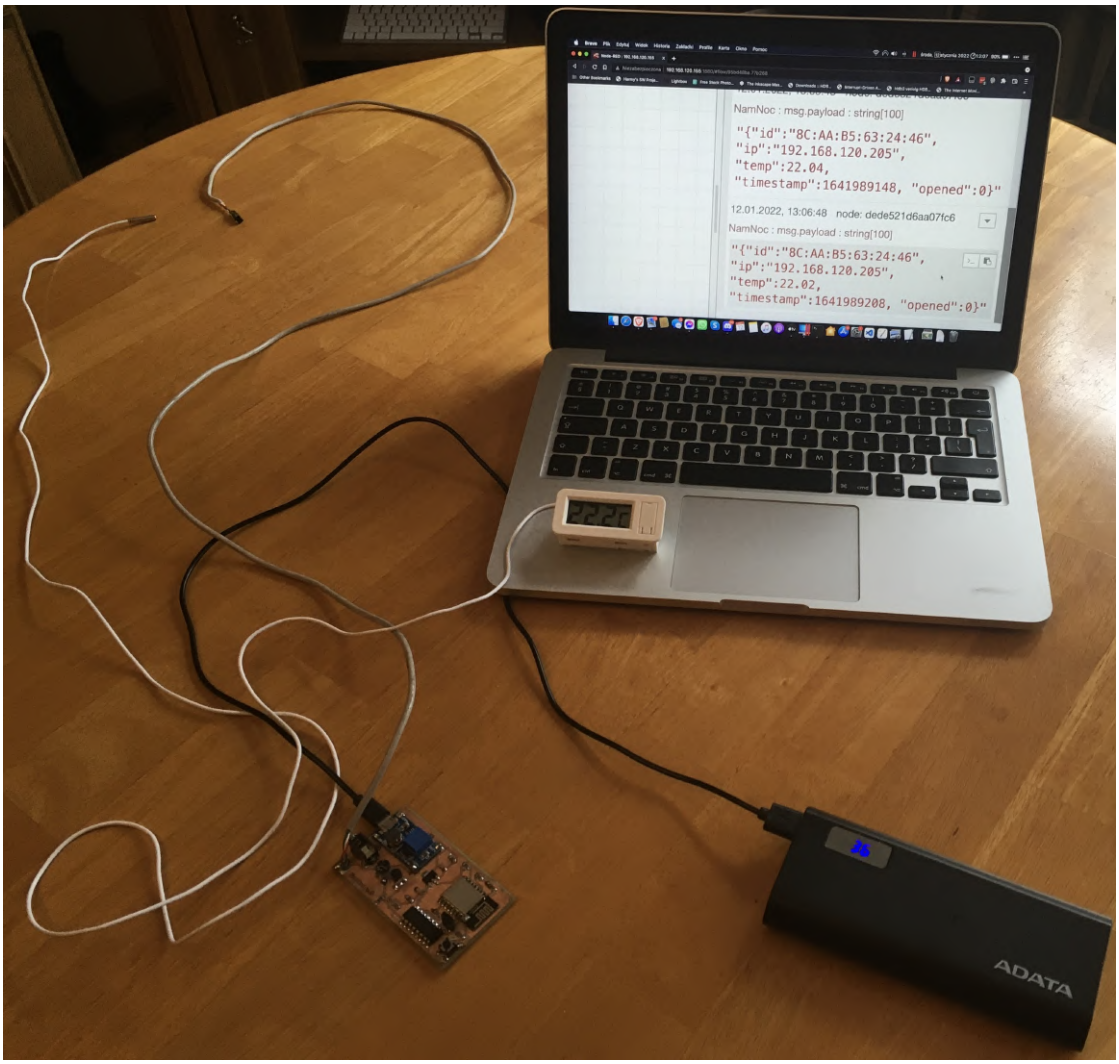
Weryfikacja i walidacja

9.1 Urządzenie pokojowe

9.1.1 Test poprawności pomiarów temperatury wykonywanych przez urządzenie

W celu weryfikacji poprawności wyników pomiarów temperatury wykonywanych przez urządzenie dokonano porównania wyników pomiaru z dostępnym na rynku termometrem *TMP-30*. Czujniki temperatur obydwu urządzeń umieszczono w bliskiej odległości (patrz rys. 9.1), następnie przez okres ok. jednej godziny w odstępach trzyminutowych spisywano wyniki pomiarów z obydwu urządzeń. Wyniki pomiarów umieszczono w tablicy 9.1.

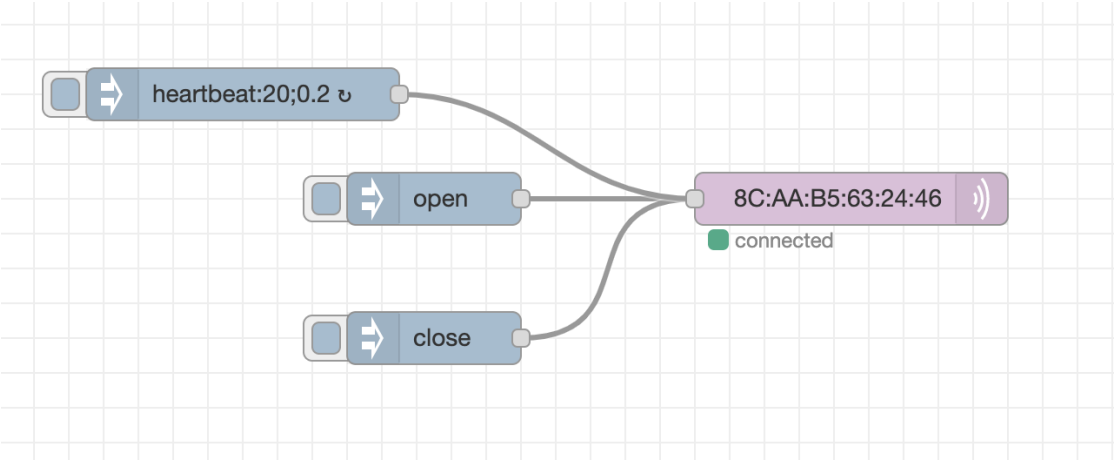
Wyniki pomiarów temperatury obydwu czujników przez cały okres eksperymentu pozostawały w granicach błędu pomiarowego urządzeń.



Rysunek 9.1: Test poprawności pomiarów temperatury wykonywanych przez urządzenie pokojowe

ESP8266		
Godzina	Urządzenie pokojowe	Termometr TMP-30
13:06	22.04	22.2
13:09	21.98	22.2
13:12	22	22.2
13:15	22.02	22.2
13:18	22	22.1
13:21	21.9	22.1
13:24	21.88	22.1
13:27	21.88	22.1
13:30	21.85	22.1
13:33	21.88	22.1
13:36	21.83	22
13:39	21.83	22
13:42	21.81	22
13:45	21.81	22
13:48	21.79	21.9
13:51	21.65	21.9
13:54	21.63	21.8
13:57	21.67	21.9

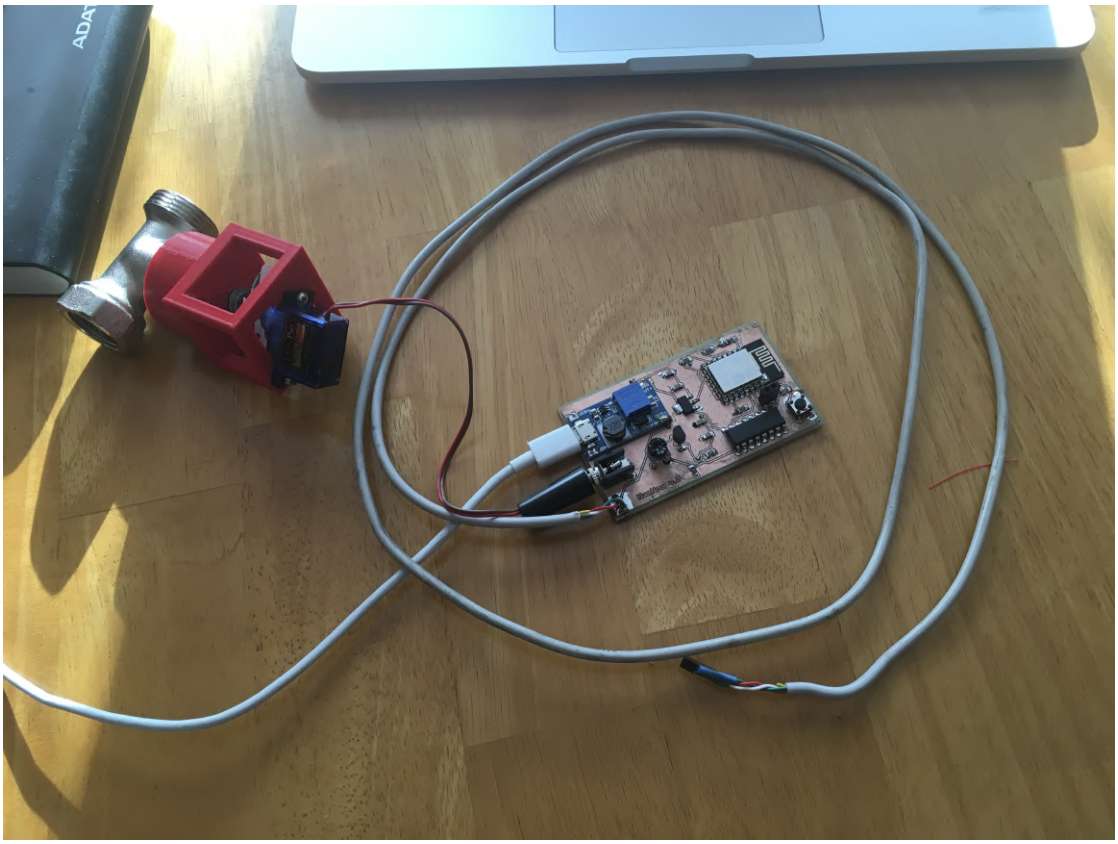
Tablica 9.1: Wyniki pomiarów temperatury dokonanych przez urządzenie pokojowe i termometr TMP-30.



Rysunek 9.2: Wysyłanie spreparowanych ramek bicia serca oraz żądań otwarcia i zamknięcia zaworu z poziomu oprogramowania Node-RED

9.1.2 Test poprawności realizowanego przez urządzenie mechanizmu otwierania i zamykania zaworu grzejnika pokojowego na żądanie urządzenia centralnego

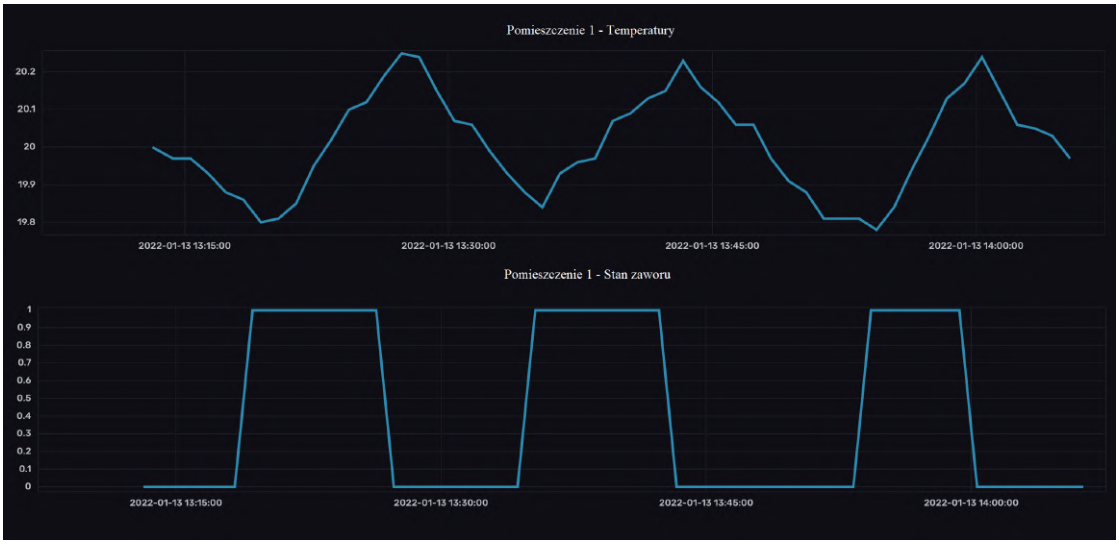
W celu przetestowania poprawności działania mechanizmu otwierania i zamykania zaworu grzejnika pokojowego na żądanie urządzenia centralnego, do urządzenia został przyłączony (do gniazda J1 - patrz rys. 5.1) zainstalowany na zaworze silnik z przekładnią (przerobiony serwomechanizm SG-90) (rys. 9.3) . Następnie poprzez protokół MQTT przesyłano do urządzenia spreparowane (identyczne jak wysyłane przez urządzenie centralne) wiadomości zawierające żądanie otwarcia lub zamknięcia zaworu (rys. 9.2). Weryfikowano czy po maksymalnym otwarciu lub zamknięciu zaworu, silnik jest zatrzymywany na skutek działania detekcji przeciążenia. Weryfikowano również, czy po zatrzymaniu silnika zawór faktycznie jest otwarty (widać czubek grzybka) bądź zamknięty (widać lekkie rozpłaszczenie uszczelki).



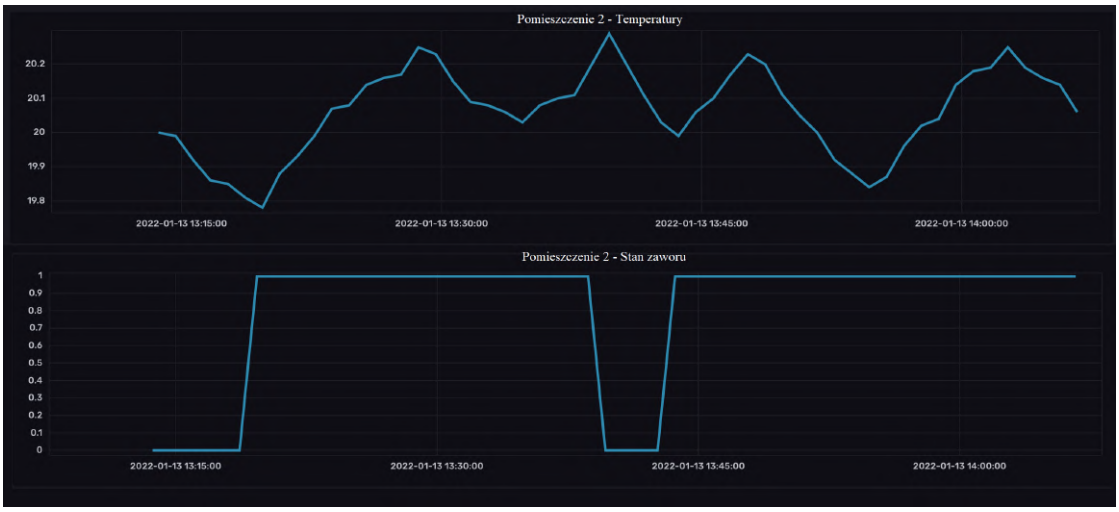
Rysunek 9.3: Silnik z przekładnią zamontowany na zaworze grzejnika pokojowego, podłączony do urządzenia pokojowego

9.2 Urządzenie centralne oraz urządzenie sterujące pracą pieca

W celu weryfikacji poprawności działania oprogramowania urządzenia sterującego pracą pieca oraz najważniejszej części oprogramowania urządzenia centralnego (odpowiedzialnej za podejmowanie decyzji o konieczności włączenia lub wyłączenia pieca i otwarcia lub zamknięcia zaworu grzejnika pokojowego w którymś z pomieszczeń) w języku JavaScript (NodeJS) napisano oprogramowanie, które symuluje zachowanie urządzenia pokojowego i warunki panujące w pomieszczeniu (wysyła wiadomości MQTT o swoim stanie, symuluje zmianę temperatury w pomieszczeniu - gdy piec jest włączony i zawór otwarty temperatura rośnie, w innym wypadku spada). Następnie uruchomiono trzy instancje tego oprogramowania. Symulowane urządzenia skonfigurowano w systemie (ustawiona temperatura - 20 stopni, histereza - 0,2 stopnia). Przez okres około jednej godziny parametry systemu - informacje o stanie pieca oraz symulowanych urządzeniach - były zapisywane w bazie danych. Wykresy przedstawiające temperaturę panującą w symulowanych pomieszczeniach oraz stan zaworów grzejników znajdujących się w nich przedstawiono na rysunkach 9.4, 9.5 i 9.6, a godziny włączeń i wyłączeń pieca przedstawiono w tablicy 9.2. Na ich podstawie można stwierdzić, że urządzenie centralne poprawnie steruje całością systemu i temperatura w pomieszczeniach jest utrzymywana zgodnie z wypracowanymi w tej pracy zasadami. W międzyczasie weryfikowano również, czy urządzenie sterujące pracą pieca poprawnie steruje przekaźnikiem (gdy piec miał być włączony, zamykał on obwód z żarówką, która świeciła). Ten test również nie wykazał żadnych błędów, potwierdzając tym samym poprawne działanie oprogramowania sterującego pracą pieca.



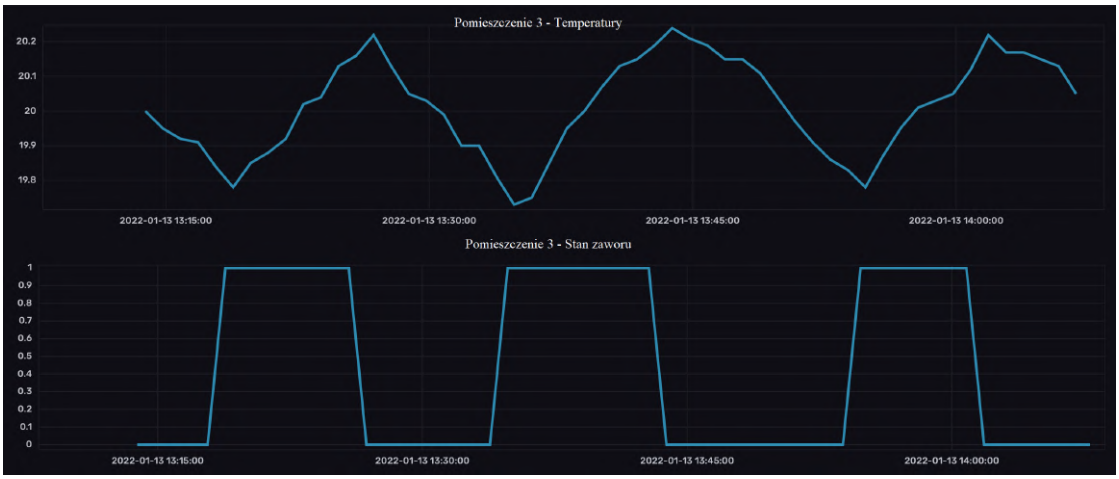
Rysunek 9.4: Warunki w pierwszym symulowanym pomieszczeniu



Rysunek 9.5: Warunki w drugim symulowanym pomieszczeniu

_time	_value	_field	_measurement
2022-01-13 13:19:10	1	On	Furnace
2022-01-13 13:29:10	0	On	Furnace
2022-01-13 13:35:10	1	On	Furnace
2022-01-13 13:48:10	0	On	Furnace
2022-01-13 13:55:10	1	On	Furnace
2022-01-13 14:03:10	0	On	Furnace

Tablica 9.2: Godziny włączeń i wyłączeń pieca.



Rysunek 9.6: Warunki w trzecim symulowanym pomieszczeniu

Rozdział 10

Podsumowanie i wnioski

- Zaprojektowane i stworzone w ramach projektu urządzenie pokojowe, jego oprogramowanie, oprogramowanie sterujące całością systemu, oraz oprogramowanie przykładowego urządzenia sterującego pracą pieca działają poprawnie. Umożliwiają one stworzenie systemu działającego zgodnie z założeniami projektu - systemu którego stworzenie było celem pracy.

- Wybrane w procesie analizy narzędzia programistyczne i języki programowania okazały się odpowiednie i umożliwiły szybką i łatwą realizację oprogramowania urządzeń stanowiących system. Czynnikiem szczególnie przyspieszającym i ułatwiającym proces tworzenia oprogramowania były mnogość bibliotek dla wybranych języków programowania (minimalizująca konieczność pisania długich fragmentów kodu w celu implementacji danej funkcjonalności, jak np. obsługa czujnika temperatury, komunikacji MQTT i http itd.) oraz spora społeczność zawiązana wokół tych języków (łatwość naprawy ewentualnych błędów w kodzie, mnogość przykładów umożliwiających naukę języka).

- Dobrane elementy elektroniczne (urządzenie pokojowe) zgodnie z wnioskami analizy okazały się ostatecznie odpowiednie i umożliwiły budowę urządzenia działającego zgodnie z założeniami projektowymi.

- Największym napotkanym problemem okazały się skutki zlekceważenia na etapie projektowania płytki drukowanej urządzenia pokojowego zarówno ilo-

ści ciepła emitowanego przez układy umieszczone na tej płytce jak i przewodnictwa cieplnego wylanej na płytce masy. Przyjęte w ramach tej pracy rozwiązanie tego problemu (umieszczenie czujnika temperatury na przewodzie), choć skuteczne, raczej nie jest rozwiązaniem estetycznym i zadowalającym. Rozsądnym będzie zatem w przyszłości poszukanie innego rozwiązania tego problemu.

- W przyszłości warto pomyśleć o realizacji dodatków dla najbardziej popularnych narzędzi umożliwiających lub ułatwiających zarządzenie inteligentnym domem tj. HomeAssistant, Domoticz czy Homebridge, aby możliwa była integracja systemu z tymi narzędziami.

Bibliografia

[1] ESP-12E WiFi Module Version 1.0. https://docs.ai-thinker.com/_media/esp8266/docs/esp12e_datasheet.pdf. dostęp 03.01.2021.

[2] Głowica Auraton TRA. https://www.auraton.pl/produkty/auraton-tra/. dostęp 30.12.2021.

[3] Learn about bluetooth - Topology options. https://www.bluetooth.com/learn-about-bluetooth/topology-options/. dostęp 30.12.2021.

[4] Moduły WiFi. https://botland.com.pl/377-moduly-wifi. dostęp 30.12.2021.

[5] XBee. https://botland.com.pl/322-xbee. dostęp 30.12.2021.

[6] DS18B20 - Dokumentacja Techniczna. https://pdf1.alldatasheet.com/datasheet-pdf/view/58557/DALLAS/DS18B20.html. dostęp 05.01.2021.

[7] Zigbee Network Topology. https://www.e-spincorp.com/zigbee-network-topology/. dostęp 30.12.2021.

[8] ESP-WIFI-MESH. https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/esp-wifi-mesh.html. dostęp 28.01.2021.

[9] ESP32-WROOM-32 Datasheet Version 2.5. https://circuits4you.com/wp-content/uploads/2018/12/esp32-wroom-32_datasheet_en.pdf. dostęp 03.01.2022.

[10] ESP8266EX Datasheet. https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf. dostęp 04.01.2021.

[11] Intro to painlessMesh. https://gitlab.com/painlessMesh/painlessMesh. dostęp 28.01.2021.

[12] Publish & Subscribe - MQTT Essentials: Part 2. https://www.hivemq.com/blog/mqtt-essentials-part2-publish-subscribe/. dostęp 03.01.2021.

[13] Interfejsy API REST. https://www.ibm.com/pl-pl/cloud/learn/rest-apis. dostęp 30.12.2021.

[14] Olaide O. Kazeem, Olubiyi Akintade, L. O. Kehinde. Comparative study of communication interfaces for sensors and actuators in the cloud of internet of things. https://www.researchgate.net/publication/318054538. dostęp 30.12.2021.

[15] Mariola Kwiatkowska. WiFi. https://stat.gov.pl/metainformacje/slownik-pojec/pojecia-stosowane-w-statystyce-publicznej/1911,pojecie.html. dostęp 30.12.2021.

[16] Łukasz Lewczuk. Inteligentne sterowanie ogrzewaniem – przegląd wybranych rozwiązań. https://www.fachowyinstalator.pl/inteligentne-sterowanie-ogrzewaniem-przeglad-wybranych-rozwiazan/. dostęp 30.12.2021.

[17] MQTT. https://mqtt.org/. dostęp 03.01.2021.

[18] Marcin Olejniczak. RECENZJA: Inteligentny Termostat SmartHome FRITZ!DECT 310. https://cyfrowyja.pl/recenzje/recenzja-inteligentny-termostat-smarthome-fritz-dect-310.html. dostęp 30.12.2021.

[19] Jak działa głowica termostatyczna. https://pec.gliwice.pl/jak-dziala-glowica-termostatyczna. dostęp 28.01.2021.

[20] Marcin Połowianiuk. Ty też możesz mieć inteligentny dom. Nest 3 w końcu jest na wyciągnięcie ręki. https://spidersweb.pl/2015/11/nest-3-w-europie.html. dostęp 30.12.2021.

[21] L293D - Dokumentacja techniczna https://pdf1.alldatasheet.com/datasheet-pdf/view/22432/STMICROELECTRONICS/L293D.html. dostęp 05.01.2021.

[22] Zawór termostatyczny danfoss ra-n 15 prosty z nastawą wstępną. http://trelka.com.pl/produkt/zawor-termostatyczny-danfoss-ra-n-15-prosty-z-nastawa-wstepna/. dostęp 28.01.2021.

[23] Wireless local area network IEE 802.11. http://www.cs.uccs.edu/~gsc/pub/master/pjfong/UCCS%20Project/Articles/IE%20802_11%20Network%20Topology.htm. dostęp 30.12.2021.

[24] Bluetooth. https://en.wikipedia.org/wiki/Bluetooth. dostęp 30.12.2021.

[25] IEEE 802.11. https://pl.wikipedia.org/wiki/IEEE_802.11. dostęp 30.12.2021.

[26] ZigBee. https://pl.wikipedia.org/wiki/ZigBee. dostęp 30.12.2021.

Dodatki

Spis skrótów i symboli

IoT	internet rzeczy (ang. Internet of Things)
API	interfejs programistyczny aplikacji (ang. application programming interface)
ISO OSI RM	model odniesienia łączenia systemów otwartych (ang. ISO Open Systems Interconnection Reference Model)
REST	zmiana stanu poprzez reprezentacje (ang. Representational state transfer)
IP	protokół internetowy (ang. internet protocol)
MAC	warstwa sterowania dostępem do medium transmisyjnego (ang. medium access control)
JSON	notacja obiektów (ang. JavaScript Object Notation)

Lista dodatkowych plików, uzupełniających tekst pracy

W systemie do pracy dołączono dodatkowe pliki zawierające:

- pliki projektu płytki drukowanej urządzenia pokojowego (EAGLE) (w katalogu *Urządzenie pokojowe/Projekt urządzenia*)
- oprogramowanie urządzenia pokojowego (*Urządzenie pokojowe/Oprogramowanie/firmware.bin*) i jego kod źródłowy (*Urządzenie pokojowe/Oprogramowanie/source*)
- oprogramowanie urządzenia sterującego pracą pieca (*Urządzenie sterujące pracą pieca/Oprogramowanie/firmware.bin*) i jego kod źródłowy (*Urządzenie sterujące pracą pieca/Oprogramowanie/source*)
- oprogramowanie urządzenia centralnego i interfejs użytkownika (*Urządzenie centralne/Production*) i ich kody źródłowe (*Urządzenie centralne/source*)
- oprogramowanie wykorzystane w procesie weryfikacji i walidacji projektu (w katalogu *Symulacja*)
- film przedstawiający działanie systemu

Spis rysunków

Spis tablic