



AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

Wydział Elektroniki, Automatyki, Informatyki i Inżynierii Biomedycznej

Projekt dyplomowy

Inteligentny system sterowania urządzeniami elektrycznymi oparty o technologię mesh.

Intelligent control system for electrical devices based on mesh topology.

Autor:	<i>inż. Filip Żmijewski</i>
Kierunek studiów:	Automatyka i Robotyka, II stopień
Opiekun pracy:	<i>dr hab. inż. Michał Turek prof. AGH</i>

Kraków, 2023

Spis treści

1. Wprowadzenie	4
1.1. Cel i zakres pracy.....	4
2. Analiza zagadnień.....	5
2.1. Dostępne rozwiązania	5
2.1.1. free@home ABB	5
2.1.2. Grenton.....	6
2.2. Stosowane rozwiązania technologiczne	7
2.2.1. Topologie	7
2.2.2. Technologia mesh	8
2.3. Stosowane rozwiązania komunikacyjne	8
2.3.1. Interfejsy komunikacyjne	8
2.4. Urządzenia pomiarowe i wykonawcze	9
2.4.1. Czujniki	9
2.4.2. Urządzenia wykonawcze.....	10
2.4.3. Mikrokontrolr ESP-32 WiFi.....	10
2.5. Aplikacje mobilne i webowe	11
2.5.1. React.js oraz React Native.....	11
2.5.2. Node.js, REST API i websockets	12
2.6. Badanie komponentów sprzętowych	13
2.6.1. Możliwości sprzętowe mikrokontrolerów.....	13
2.6.2. Inicjacja topologii siatki z wykorzystaniem ESP-32.....	15
2.6.3. Czujnik temperatury	17
2.6.4. Przekazniki	18
3. Projekt systemu	19
3.1. Przypadki użycia systemu.	20
3.1.1. Logowanie do systemu (identyfikator: UC1.0).....	20
3.1.2. Konfiguracja urządzenia przez użytkownika (identyfikator: UC1.1).....	21
3.1.3. Połączenie urządzenia w topologię mesh (identyfikator: UC1.2).....	22
3.1.4. Przesył danych pomiędzy urządzeniami (topologia mesh) (identyfikator: UC1.3)	

3.1.5.	Sterowanie oświetleniem z wykorzystaniem fizycznych przełączników (identyfikator: UC1.4).....	24
3.1.6.	Manualne sterowanie temperaturą (identyfikator: UC1.5)	25
3.1.7.	Manualne sterowanie oświetleniem (identyfikator: UC1.6)	26
3.1.8.	Zmiana harmonogramu pracy urządzeń sterujących (identyfikator: UC1.7)	27
3.1.9.	Odbiór danych z czujnika (identyfikator: UC1.8).....	28
3.1.10.	Automatyczna kontrola temperatury. (identyfikator: UC1.9)	29
3.2.	Architektura systemu	30
3.2.1.	Urządzenia sterujące.....	30
3.2.2.	Serwer i komunikacja.....	31
3.2.3.	Aplikacja webowa i mobilna.....	32
3.3.	Zachowania systemu.....	32
3.3.1.	Logowanie użytkownika	33
3.3.2.	Konfiguracja urządzeń	33
3.3.3.	Przesył danych pomiędzy urządzeniami	34
3.3.4.	Sterowanie urządzeniami z wykorzystaniem fizycznych komponentów.....	35
3.3.5.	Sterowanie urządzeniami z wykorzystaniem aplikacji.	36
3.4.	Interfejsy użytkownika	37
3.4.1.	Okno logowania	37
3.4.2.	Okno widoku dostępnych urządzeń	37
3.4.3.	Okno konfiguracji urządzeń	38
3.4.4.	Okno sterowania urządzeniem	38

1. Wprowadzenie

Wszechobecny rozwój świata, w szczególności technologii i zagadnień z nią związanych, popycha ludzi do odkrywania rozwiązań ułatwiających codzienne życie. Z natury, każdy z nas pragnie ułatwić swoje życie w pewnym stopniu, szczególnie gdy mowa jest o codziennych, monotonnych czynnościach. Owocem takich zachowań stały się systemy smart home. Automatyzacja codziennych, potwarzanych wielokrotnie czynności to coś co w ostatnich latach stało się głównym celem wielu światowych korporacji. Ich szybki rozwój doprowadził do możliwości takich jak ograniczenie zużycia energii, ułatwienie codziennych czynności i znaczącą poprawę bezpieczeństwa.

Obecnie dostępnych jest wiele systemów smart home, które pozwalają na sterowanie różnymi urządzeniami elektronicznymi i elektrycznymi. Jednym z najbardziej obiecujących systemów są te, oparte o topologię siatki, pozwalające na tworzenie ogromnych sieci urządzeń w dużych budynkach. Takie systemy sprawiają, że coraz więcej firm i osób prywatnych decyduje się na zastosowanie takich rozwiązań w swoich inwestycjach. Zaletami takich systemów jest przede wszystkim zdalna kontrola wszystkich urządzeń i podgląd pomiarów z wielu czujników. Niestety, wygodą jaką zapewniają te systemy jest bardzo kosztowna. Często pojedyncze czujniki lub urządzenia wykawancze osiągają ceny zaczynające się od 400zł.

Poprzez zastosowanie posiadanej wiedzy, wykorzystanie dostępnych technologii i źródeł informacji pozwala na samodzielną budowę i implementację takiego projektu. Wykonanie urządzeń pozwalających na instalację takiego systemu może okazać się wyzwaniem wartym realizacji. Urządzenia te bez problemu mogą osiągać właściwości urządzeń komercyjnych, dodatkowo pozwalając na ich większą modyfikację ze względu na ich własnoręczne projektowanie.

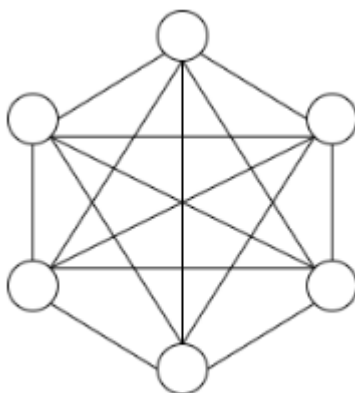
1.1. Cel i zakres pracy

Celem niniejszej pracy jest zaprojektowanie, stworzenie i implementacja fizycznych urządzeń sterujących komponentami elektrycznymi, których architektura pozwoli na podłączenie do systemu opartego o sieć mesh i obsługę ich z poziomu aplikacji webowej i mobilnej.

Do realizacji pracy przewidziano korzystanie z ogólnodostępnych technologii takich jak mikrokontrolery, ogólnodostępne czujniki, a w celu realizacji zaplanowanego software'u wykorzystanie darmowych platform oraz bibliotek.

2. Analiza zagadnień

Ciągły rozwój technologiczny idący w parze z chęcią ułatwienia sobie życia przez człowieka, prowadzi do powstawania nowych rozwiązań dostępnych na rynku. Firmy prześcigają się wprowadzając coraz to nowsze rozwiązania w różnych dziedzinach. Jedną z tych dziedzin są inteligentne budynki. Każde z tych rozwiązań posiada swoje wady i zalety. Większość dostępnych na rynku systemów inteligentnych budynków posiada jednakową budowę, tworzą one scentralizowaną sieć urządzeń z jednym lub kilkoma urządzeniami głównymi, które następnie łączą się z serwerem. Do sterowania urządzeniami wykonawczymi służy aplikacja mobilna i przeglądarkowa. Dodatkowo ta druga pozwala najczęściej na konfigurację całego systemu. Rzadziej na rynku spotyka się rozwiązania oparte o technologie mesh, która pozwala na komunikację każdego urządzenia z każdym i dzięki temu przesyłania poleceń pomiędzy nimi bez wykorzystania centralnego serwera. Dzięki temu otrzymujemy możliwość znaczącego rozproszenia zaimplementowanych w systemie urządzeń, które tworząc siatkę połączeń (rys.1.1.) przez co zmniejszają awaryjność takiego systemu.



Rys.2.1 Schemat topologii mesh

2.1. Dostępne rozwiązania

Na rynku znajduje się obecnie szeroka gama rozwiązań pozwalająca na instalacje inteligentnego domu lub budynku, również takich o rozproszonych topologiach. Wiele zagranicznych oraz polskich firm prezentuje możliwości konstrukcji takich budowli z wykorzystaniem ich autorskich systemów, na które składa się szereg technologii.

2.1.1. free@home ABB

Taką firmą jest na przykład ABB, która prowadzi swój autorski projekt inteligentnego budynku. Okrzyknęty nazwą free@home system jest bardzo otwartym na modyfikacje



Rys.2.2 System access point

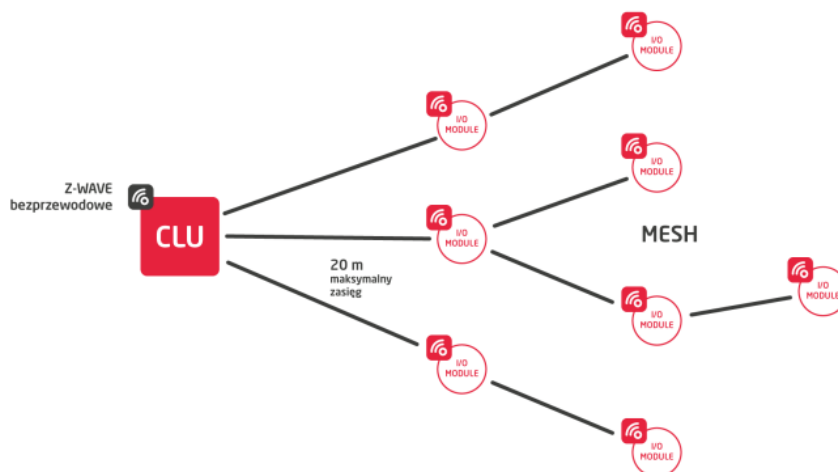
i personalne ustawienia projektem. Podstawą systemu jest urządzenie SysAP (rys.1.2.) służący jako mózg systemu zrzeszający inne urządzenia.

Urządzenie to, poza możliwością bezprzewodowej komunikacji z urządzeniami wykonawczymi, łączyć się może również z kolejnym takim samym urządzeniem. W ten sposób, w przypadku np. instalacji w dużym budynku, może stworzyć siatkę połączeń i przekazywania informacji pomiędzy nimi. Dodatkowo system ten gwarantuje bezpośrednią komunikację urządzeń wykonawczych co pozwala na rozmieszczanie ich w znacznej odległości od urządzenia matki i pośrednie przekazywanie informacji i poleceń od serwera. [1]

System free@home jest jednym z najlepiej rozwiniętych systemów na rynku. Pozwala na wdrożenie i obsługę większości potrzebnych podsystemów. Dzięki zastosowaniu częściowej topologii siatki otrzymujemy system z dużą przepustowością danych oraz o nieograniczonym polu działa ze względu na pośrednie przekazywanie danych.

2.1.2. Grenton

Również polskie firmy specjalizują się w technologiach budynków inteligentnych. Grenton, z siedzibą w Krakowie, w swojej ofercie prezentuje wiele popularnych na rynku rozwiązań w wielu interesujących konfiguracjach. Projekt ten wykorzystuje w swojej architekturze znany wśród producentów smart home protokół Z-wave. Zamyśl protokołu opiera się o komunikację urządzeń w topologii mesh(rys.1.3.), gdzie urządzenia łączą się między sobą rozprzestrzeniając siatkę połączeń. Pozwala to na bezawaryjny system o niskim zużyciu energii. Na stronie producenta dostępnych jest wiele ofert przedstawiających dostępne schematy połączeń oraz kombinacje pomiędzy sieciami przewodowymi i bezprzewodowymi.



Rys.2.3 Schemat przykładowej sieci opartej o Z-wave oraz topologie mesh [2]

2.2. Stosowane rozwiązania technologiczne

Niestety, oferowane przez producentów systemy są zazwyczaj bardzo drogie w instalacji szczególnie w przypadku, gdy są skomplikowane i bardzo rozproszone, a budynki posiadają dziesiątki albo nawet setki pomieszczeń. Z wykorzystaniem wiedzy inżynierskiej oraz ogólnodostępnych, tanich technologii istnieje możliwość stworzenia własnej instalacji budynku inteligentnego, również takiej bardzo zaawansowanej jak te o rozproszonej topologii. W trakcie tworzenia projektu składającego się na część praktyczną pracy, wykorzystane zostaną różne technologie jakie dostarcza nam świat techniczny. Temat pracy w głównej mierze opiera się na stworzeniu systemu, który ma pozwalać na implementacje inteligentnego budynku. Głównymi założeniami początkowym są: oparcie systemu połączeń urządzeń elektronicznych o technologie mesh, stworzenie aplikacji webowej i mobilnej oraz komunikacja całości. Do stworzenia aplikacji wykorzystane zostaną dostępne rozwiązania takie jak React.js, React native, Node.js, REST Api oraz websockets.

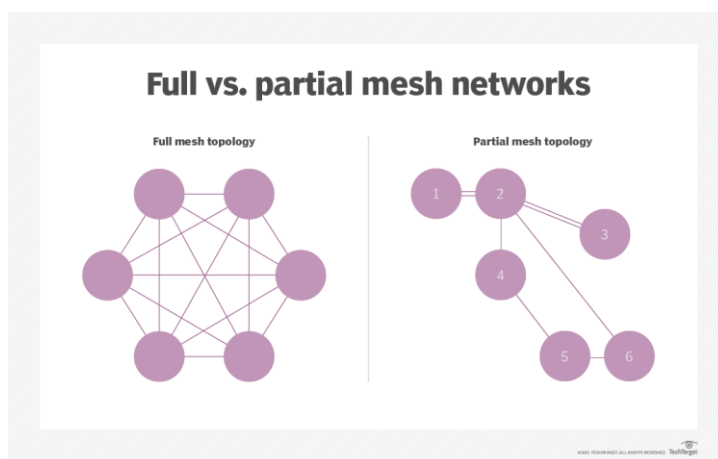
2.2.1. Topologie

Nawiązując do tematu pracy, jakim jest inteligentny system sterowania, należy rozważyć temat jakim jest topologia. System, na który składa się wiele urządzeń oparty jest w większości przypadków o sieć, która rozróżnia kilka rodzajów nazywanych topologiami. Obecnie producenci starają się być jak najlepiej dostosowanym pod potrzeby klienta i tworzą systemy oparte o sieci przewodowe oraz bezprzewodowe. Najczęściej stosowanym standardem przy tworzeniu inteligentnych budynków jest KNX. Pozwala on na obsługę ogromnej ilości urządzeń przy wykorzystaniu dwużyłowej magistrali. Posiada bardzo uniwersalny standard i jest wspierane przez ogromną ilość światowych korporacji. KNX również pozostawia wybór po stronie klienta, wspiera topologie takie jak gwiazda, drzewo oraz linia.

Gdy mówimy o sieciach bezprzewodowych sytuacja wygląda nieco inaczej. Często spotykamy topologię gwiazdy, gdzie konkretne urządzenia łączą się z serwerem, a nie następuje komunikacja pomiędzy nimi. Jest to rozwiązanie łatwe w implementacji, jednakże oparte w 100% o jeden serwer co może powodować niechciane awarie. Rozwiązaniem tego problemu może być wspomniana już wcześniej topologia mesh (siatki).[3]

2.2.2. Technologia mesh

Topologia mesh jest rodzajem lokalnej sieci, w której komunikacja polega na łączeniu urządzeń należących do sieci na zasadzie „każdy z każdym” tworząc przy tym siatkę połączeń. Ten typ komunikacji pozwala na bardzo efektywny przesył danych pomiędzy urządzeniami, bez konieczności korzystania z jednego głównego serwera. Wyróżnia się dwa typy połączeń mesh, pełnej siatki (rys. 1.3.) gdzie każde z urządzeń łączy się z każdym oraz niepełnej siatki (rys. 1.3.) gdzie nie musi występować połączenie każdego urządzenia z każdym [4]



Rys. 2.4. Pełna topologia mesh (po lewej), częściowa topologia mesh(po prawej)[5]

Ze względu na pewne ograniczenia systemowe, konieczność zastosowania zcentralizowanego serwera, który pozwoli na obsługę całości systemu, przy tworzeniu projektu systemu zastosuję częściową topologię mesh.

2.3. Stosowane rozwiązania komunikacyjne

Projekt zostanie oparty o ogólnodostępne technologie i niskobudżetowe urządzenia. Kluczem jest dopasowanie istniejących czujników, mikrokontrolerów tak, aby istniało realne i proste połączenie ich ze sobą. Z pomocą przychodzą nam ogólnodostępne interfejsy komunikacyjne.

2.3.1. Interfejsy komunikacyjne

Obecnie w świecie technologii korzysta się z różnorodnych protokołów komunikacyjnych, przewodowych jak i bezprzewodowych. Pozwalają one na przesył danych pomiędzy urządzeniami, często bardzo różnych od siebie. Zapis podstawowych zasad pozwala na odczyt informacji bez ich uszkodzenia.

Najpowszechniejszymi interfejsami komunikacji sieciowej jest Ethernet bądź WiFi. Ten pierwszy stanowi pewnego rodzaju ograniczenie ze względu na konieczność wykorzystania przewodów, co może stanowić kłopot gdy oczekujemy instalacji systemu w istniejącym już budynku. Jeżeli chodzi o drugi interfejs, WiFi, może obudzić pewne wątpliwości ze względu na charakter bezprzewodowy. Na szczęście dzięki rozwijającej się technologii i powstaniu 5G systemy oparte o sieć bezprzewodową cechują się porównywalną stabilnością do sieci przewodowych.

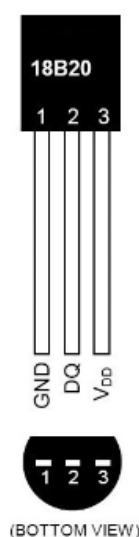
Drugą kwestią jest dobór protokołów do komunikacji z czujnikami. Tutaj sprawa wygląda nieco inaczej, w tym przypadku oczekujemy często podłączenia do jednej magistrali komunikacyjnej. Obecnie powszechne są takie protokoły jak I²C, SPI czy np. One-wire. Wszystkie charakteryzują się komunikacją po jednej linii magistrali z wykorzystaniem miejsc w kolejności lub adresów urządzeń. Pozwala to na zajęcie jedynie jednego pinu na płycie procesora z zachowaniem możliwości uzyskania takich samych wyników. Przy wyborze płytki mikroprocesorowej do obsługi urządzeń fizycznych należy zadbać o to aby była zdolna do obsługi tych protokołów. Podobnie sytuacja wygląda przy wyborze czujników gdzie musimy zadbać, aby dało się je komunikować z wykorzystaniem tych magistral.

2.4. Urządzenia pomiarowe i wykonawcze

Kluczem do stworzenia elektrycznych urządzeń systemu Smart Home jest dobranie czujników i urządzeń wykonawczych jakie mogą być stosowane do ich budowy.

2.4.1. Czujniki

Jednym z podstawowych czujników w takim systemie są czujniki temperatury. Jednymi z najbardziej popularnych na rynku są cyfrowe czujniki DS18B20. Są one bardzo małych rozmiarów, dostosowane do pracy w temperaturach, które uzyskujemy w domu.



Rys. 2.5. Czujnik temperatury DS18B20 [6]

Czujnik ten zasilany jest napięciem 5V, jest przystosowany do użytku z protokołem One-wire, który opisany został wcześniej. Każdy z termometrów posiada swój unikalny adres co pozwala na ich bardzo prostą identyfikację. Te czujniki zostaną wykorzystane w projekcie do wizualizacji działania urządzeń.

2.4.2. Urządzenia wykonawcze

Jeżeli chodzi o urządzenia wykonawcze tutaj sytuacja staje się znacznie prostsza ze względu, iż większość urządzeń możemy obsłużyć zwykłym przekaźnikiem. Tak też stanie się w tym przypadku, do systemu zostanie wdrożony przekaźnik SRD-5VDC-SL-C sterowany napięciem 5V ze zwykłego pinu procesora. Pozwala on na sterowanie urządzeniami do napięcia 250V. Pozwala na podłączenie takich urządzeń jak rolety, światła, drzwi lub bramy. Co zdecydowanie rozszerza stosowalność projektowanych urządzeń.



Rys. 2.6. Przekaźnik SRD-5VDC-SL-C [7]

2.4.3. Mikrokontroler ESP-32 WiFi

Kluczem do sukcesu przy konstruowaniu wspomnianej w poprzednim podrozdziale topologii mesh jest odpowiedni wybór urządzeń, które sprawować będą rolę węzłów takiego systemu. Obecnie wybór takich urządzeń jest olbrzymi, a ceny zaczynają się już od kilku złotych. Kontrolery wyposażone w moduły wifi lub bluetooth zalewają rynek sprawiając, że tworzenie własnych systemów w oparciu o sieci urządzeń bądź IoT stało się bardzo proste. Jednym z najpopularniejszych chipów, o którego oparte są mikrokontrolery jest ESP32. Jak wspomniano wcześniej jego funkcjonalności są wzbogacone o możliwość komunikacji poprzez sieć bezprzewodową, dzięki czemu może łączyć się z serwerem oraz innymi urządzeniami w tej sieci. Chip ten dodatkowo obsługuje protokoły bezpieczeństwa np. WPA2. Posiada wyprowadzenia magistrali co pozwala na obsługę przekaźników i tranzystorów, a w rezultacie na obsługę urządzeń elektrycznych.[6]

Powyższy mikrokontroler to tylko jeden z wielu dostępnych na rynku. Zauważyć można, iż dostępne rozwiązania są na prawdę bogate i pozwalają na wiele sposobów poprowadzić do

rozwiązania postawionych sobie problemów. Dzięki dostępności odpowiednich bibliotek oraz interfejsów istnieje możliwość bardzo prostego sterowania takimi mikrokontrolerami poprzez informacje wysyłane z serwera, choćby w formacie danych JSON. A to sprawia, że jesteśmy już o krok od obsługi takiego urządzenia aplikacją w telefonie bądź komputerze.

2.5. Aplikacje mobilne i webowe

Inteligentny system sterowania urządzeniami w budynku nie mógłby w pełni funkcjonować bez wykorzystania wspomnianej wcześniej aplikacji służącej jego konfiguracji i obsłudze. Obecnie najpopularniejszym narzędziem stosowanym jako aplikacja, a w szczególności, jeżeli chodzi o sektor inteligentnych budynków, są aplikacje webowe. Obsługiwane z poziomu przeglądarki strony, dzięki którym możemy sterować rzeczywistymi urządzeniami to już dla każdego z nas codzienność. Zgodnie z raportem portalu StackOverflow w roku 2020 najczęściej stosowanym językiem do tworzenia tych właśnie aplikacji wraz ze wszystkimi otaczającymi go technologiami jest JavaScript.[7]

Kolejnym, niemalże obowiązkowym elementem takiego systemu jest aplikacja mobilna. Obecnie na rynku przy tworzeniu aplikacji mobilnych stosuje się szeroką gamę języków programowania w zależności od chociażby systemu operacyjnego bądź ich zastosowania.[8] W ostatnim czasie język programowania JavaScript, który w głównej mierze jest kojarzony z aplikacjami webowymi zaczął być rozwijany na polu programowania aplikacji mobilnych. Najnowsze osiągnięcia świata programistycznego pozwalają na tworzenie tychże aplikacji z wykorzystaniem wspomnianego języka. Z pomocą tutaj przychodzą technologie takie jak React Native.

2.5.1. React.js oraz React Native

React.js początkowo został stworzony na wewnętrzne potrzeby korporacji Meta, która korzystała z tego narzędzia w celu dewelopmentu takich platform jak Facebook czy Instagram, następnie udostępniona jako freesourcowa biblioteka języka JavaScript z każdym rokiem zyskuje coraz większą popularność na rynku. Na ten moment prawdopodobnie większość firm stosuje to narzędzie przy jednym ze swoich projektów.[9] Biblioteka ta, pozwala na tworzenie zaawansowanych interfejsów użytkownika w bardzo przystępny sposób. Wykorzystanie porównanie pomiędzy stanami aplikacji i aktualizuje dzięki temu zmiany. Zastosowany w tej bibliotece język JSX pozwala na „wstrzyknięcie” html wprost do aplikacji javascript dzięki czemu zarządzanie DOM stało się bardzo proste i szybkie. Tworzone aplikacje są bardzo szybkie ze względu na to, iż re-renderowane są jedynie elementy, które zostały zmienione. Do biblioteki React.js dołączony został również pakiet React Native, który pozwala na tworzenie aplikacji mobilnych z wykorzystaniem dokładnie tych samych języków programowania i

minimalnej znajomości architektury urządzeń mobilnych. Dzięki wykorzystaniu gotowych, opensourcowych bibliotek komponentów takich jak MUI oraz bootstrap, podczas tworzenia tych aplikacji można pominąć często powtarzający się schemat pisania w języku html³. Tworzone aplikacje mają charakter jednej strony. Nie następuje tutaj przekierowanie do innych adresów, wszystko generowane jest pod jednym z nich przez co są bardzo intuicyjne i świetnie spisują się jako aplikacje do sterowania różnymi systemami. Korzystając z technologii React mamy możliwość korzystania z niezliczonych pakietów i bibliotek npm. Mowa tutaj o bibliotekach takich jak socket.io, które dzięki wykorzystaniu websocketów pozwala na komunikację z serwerem i przesył danych w czasie rzeczywistym. Axios, dzięki którym możemy korzystać z API oraz wiele wydanych na licencji MIT bibliotek ułatwiających pracę. Podczas tworzenia interfejsu użytkownika możemy na bieżąco podglądać dokonane zmiany i ewentualne błędy dzięki wbudowanemu w bibliotekę React serwerowi.

2.5.2. Node.js, REST API i websockets

Konieczny do stworzenia projektu budynku inteligentego jest serwer, który ponosi odpowiedzialność za przesył danych pomiędzy wszystkimi podzespołami systemu. Obecnie prawie każdy język programowania jest używany do tworzenia programów typu serwer. Stosując w swoim projekcie technologie biblioteki React.js programiści uciekają do wykorzystania narzędzi podobnych do niego. Obecne realia przyniosły rozwiązania pozwalające ograniczyć się do tworzenia całych systemów z wykorzystaniem jednego języka. Jedną z takich technologii rozszerzających działanie jednego z języków jest Node.js. Rozwiązanie to jest środowiskiem programistycznym służącym do tworzenia aplikacji typu serwerowego. Narzędzie to pozwala na tworzenie aplikacji w języku JavaScript poza przeglądarką. Możliwe jest to dzięki przeniesieniu silnika przeglądarki V8 stworzonego przez firmę Google do systemu. Dzięki wykorzystaniu tego narzędzia istnieje możliwość tworzenia aplikacji webowych oraz mobilnych typu client-server z wykorzystaniem jednego języka jakim jest JavaScript.[10]

Opensourcowy charakter oraz manager pakietów npm pozwala na bardzo proste stworzenie aplikacji typu serwer. Z pomocą przychodzą tutaj takie biblioteki jak express.js lub socket.io. Ta pierwsza, udostępniona na licencji MIT, służy do tworzenia REST API, które pozwala na przesył danych pomiędzy serwerem, a interfejsem użytkownika. Omawiane API składa się z metod http takich jak POST, GET itp. Dzięki im możliwe jest przesłanie i odbiór danych. Korzystanie z takiego rozwiązania niesie ze sobą wiele korzyści, dane przesyłane są w formacie json, który jest stosowany przy tworzeniu wszelakich aplikacji, w których to strona kliencka wysyła zapytania i przesyła dane do serwera. Jeżeli przy tworzeniu aplikacji konieczna jest reakcja interfejsu użytkownika na zapytania serwera skorzystanie z tej technologii nie będzie miało zastosowania. Z pomocą przychodzi tutaj biblioteka socket.io. Dzięki niej istnieje możliwość stworzenia tzw. komunikacji w czasie rzeczywistym pomiędzy serwerem, a interfejsem użytkownika. Rozwiązanie to oparte jest o protokół websocketów. Pozwala na

wysyłanie zapytań jednocześnie ze strony serwera oraz strony klienta, a nawet wielu różnych klientów, ponieważ istnieje możliwość komunikacji poprzez wiele portów. Protokół websocket oparty jest o architekturę TCP/IP co również w tym wypadku pozwala wykorzystać serwer w komunikacji z rzeczywistymi urządzeniami jakimi są np. mikrokontrolery, sterowniki itp.

2.6. Badanie komponentów sprzętowych

Aby w odpowiedni sposób zaprojektować inteligentny system sterowania urządzeniami konieczna jest szeroka znajomość posiadanych komponentów sprzętowych. W poprzednich podrozdziałach podjęta została decyzja o tym jakie części będą stosowane, w tym podrozdziale przeprowadzone zostaną badania sprzętu pozwalające pokazać jego zastosowanie, możliwości i doprowadzić do sytuacji nieprzewidzianych przez producenta. W przypadku płytek mikrokontrolera przeprowadzona zostanie próbna integracja i implementacja topologii.

2.6.1. Możliwości sprzętowe mikrokontrolerów

Główną rolę w całym systemie odgrywają mikrokontrolery. To one odpowiadają za sterowanie i służą jako węzły systemu odpowiedzialne za przesył informacji. W tym podrozdziale zajmę się eksperymentami na tym komponentcie.

Aby urządzenie spełniało swoje zastosowanie konieczne jest odpowiednie działanie modułu WiFi, który znajduje się w urządzeniu. Założenia projektu zakładają, że mikrokontroler będzie działał w formie Access Point tzn. będzie generował własną sieć w celu stworzenia topologii siatki opartej o tę sieć oraz musi mieć możliwość łączenia się do tej sieci. Dlatego w celu zbadania pracy urządzenia w tych trybach wgrane zostały kolejno dwa programy, jeden do łączenia się z istniejącą siecią, oraz drugi umożliwiający tworzenie Access Point. W obu przypadkach korzystam z popularnej open source biblioteki WiFi.h. W pierwszym przypadku zaimplementowany został następujący kod:

```
//Uzupełnienie danych sieci domowej
const char* ssid = "Nazwa_sieci";
const char* password = "Haslo";




void setup()
{
  WiFi.begin(ssid, password); //Rozpoczęcie połączenia z siecią
  //Poniższa funkcja jest wykonywana gdy połączenie jest ustalone
  if (WiFi.status() == WL_CONNECTED){
    Serial.println("[WiFi] WiFi is connected!");
    Serial.print("[WiFi] IP address: ");
    Serial.println(WiFi.localIP());
  }
}
```

Rys. Informacja na temat połączenia i adresu wysłana przez płytkę.

Dzięki czemu po uzupełnieniu danych logowania mojej domowej sieci WiFi nastąpiło połączenie, które możemy tutaj obserwować.

```
[WiFi] WiFi is connected!  
[WiFi] IP address: 192.168.50.57
```

Rys. Informacja na temat połączenia i adresu wysłana przez płytkę.

		esp32-C89300	192.168.50.57	DHCP	48:E7:29:C8:93:00		65	6	00:00:29
---	---	--------------	---------------	------	-------------------	---	----	---	----------

Rys. Informacja o urządzeniu zaczerpnięta ze strony routera i listy urządzeń do niego podłączonych.

W drugim przypadku wytworzony został Access Point, do którego możemy podłączyć się innymi urządzeniami. Na płytkę wgrany został poniższy kod:

```
//Stworzenie danych sieci  
const char *ssid = "MyAP";  
const char *password = "MyAP12345";  
  
void setup() {  
    //Inicjalizacja AP  
    if (!WiFi.softAP(ssid, password)) {  
        log_e("Soft AP creation failed.");  
        while(1);  
    }  
}  
  
void loop() {  
    //odczyt liczby klientów w interwałach czasowych  
    int numClients = WiFi.softAPgetStationNum();  
    Serial.print("Number of connected clients: ");  
    Serial.println(numClients);  
    delay(2000);  
}
```

Rys. Uproszczony kod zaimplementowany na płytkę

Dzięki temu możemy zaobserwować następujące wyniki:

```
12:36:11.443 -> Configuring access point...  
12:36:11.521 -> AP IP address: 192.168.4.1  
12:36:11.568 -> Server started  
12:36:11.568 -> Number of connected clients: 0  
12:36:13.560 -> Number of connected clients: 0  
12:36:15.555 -> Number of connected clients: 0
```

Rys. Informacja z płytki po uruchomieniu serwera

Dzięki przeprowadzonym eksperymentom zauważamy, patrząc na timestamp'y z obu logów. Iż w momencie łączenia się jednej płytki z AP ilość klientów podłączonych i rozpoznawanych przez AP się zwiększyła.

Rozszerzając eksperyment, korzystamy z pierwszego kodu implementując go w sposób identyczny jak w pierwszym przypadku na dodatkowym urządzeniu, co pozwala nam zaobserwować połączenie jednego urządzenia do AP, który został utworzony na drugim urządzeniu.

12:29:09.938 -> [WiFi] WiFi is disconnected	12:29:01.980 -> Number of connected clients: 1
12:29:10.435 -> [WiFi] WiFi is connected!	12:29:04.012 -> Number of connected clients: 1
12:29:10.435 -> [WiFi] IP address: 192.168.4.2	12:29:06.006 -> Number of connected clients: 0
	12:29:07.999 -> Number of connected clients: 0
	12:29:10.013 -> Number of connected clients: 1
	12:29:11.980 -> Number of connected clients: 1

Rys. Po lewej informacja z płytki, która podłączała się do AP, po prawej stronie informacja z AP.

2.6.2. Inicjacja topologii siatki z wykorzystaniem ESP-32

Wyniki uzyskane w poprzednim podrozdziale świadczą o tym, iż urządzenia posiadają odpowiednie możliwości do zainicjalizowania topologii mesh. W kolejnym kroku dokonamy pierwszej implementacji takiej topologii, która pozwoli na oparcie całego projektu o to rozwiązanie. W celu budowy takiej topologii skorzystam z opensourcowej biblioteki `painlessMesh`.

Do implementacji wykorzystam 4 urządzenia ESP-32, które zostaną połączone w sieć opartą o architekturę siatki. Dzięki bardzo uniwersalnej bibliotece, na każdą ze stosowanych płytek, wgrane zostanie to samo oprogramowanie. Po uruchomieniu urządzeń będziemy mogli obserwować wymianę podstawowych informacji pomiędzy urządzeniami. Poniżej opisana zostanie struktura wykorzystanego kodu.

```
//Definiowanie informacji o sieci
#define MESH_PREFIX "MyMesh"
#define MESH_PASSWORD "MyMesh123"
#define MESH_PORT 5555

painlessMesh mesh; //definiowanie sieci mesh

Task taskSendMessage( TASK_SECOND * 1, TASK_FOREVER, &sendMessage ); //Ustawienie cyklicznego wysyłania wiadomości

//Definicja funkcji przesyłającej wiadomości
void sendMessage() {
    String msg = "Hi from node id: 1";
    msg += mesh.getNodeId(); // Pozyskanie ID węzła/urządzenia
    mesh.sendBroadcast( msg ); //Przesłanie wiadomości do wszystkich członków sieci
    taskSendMessage.setInterval( TASK_SECOND * 5 );
}

void setup() {
    mesh.init( MESH_PREFIX, MESH_PASSWORD, &userScheduler, MESH_PORT ); //inicjalizacja sieci mesh z wykorzystaniem
}

void loop() {
    mesh.update(); //Uruchomienie sieci
}
```

Rys. Uproszczony kod odpowiadający za inicjalizację i działanie urządzenia w sieci mesh

Dzięki implementacji można zaobserwować poniższe wyniki świadczące o poprawnej wymianie informacji.

```
startHere: Received from 2808636797 msg=Hi from node id: 32808636797
startHere: Received from 701096317 msg=Hi from node id: 4701096317
startHere: Received from 2808732869 msg=Hi from node id: 22808732869
```

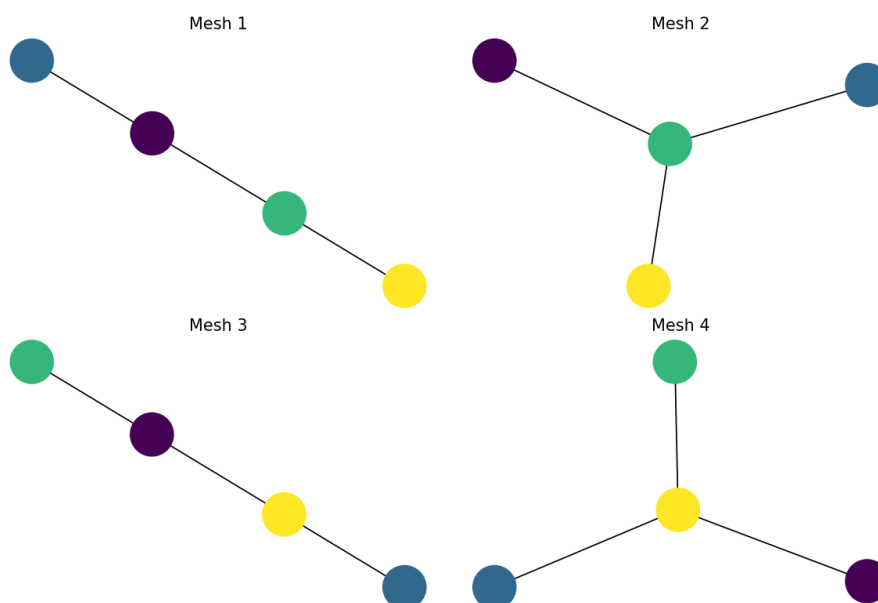
Rys. Wiadomości odebrane przez jeden z węzłów od innych członków sieci.

Dodatkowo stosowana biblioteka posiada funkcję, która zwraca architekturę zbudowanej siatki w formacie JSON, co możemy obserwować poniżej.

```
{
  "nodeId": 701010689,
  "subs": [
    {"nodeId": 2808636797, "subs": [{"nodeId": 2808732869}]},
    {"nodeId": 701096317}
  ]
},
```

Rys. Architektura siatki w formacie JSON

Otrzymanie takiego formatu może dodatkowo pozwolić na obrazowanie połączeń i ich dynamiczną zmianę w zależności od przemieszczania się urządzeń. Taka wizualizacja dobrze obrazuje moment, w którym dwa urządzenia oddalają się od siebie na tyle, aby stracić połączenie ze sobą, lecz mimo tego dalej otrzymują od siebie informację co świadczy o poprawności działania topologii mesh. W ramach badania stworzony został skrypt, który zbierał informacje z jednego z urządzeń na temat obecnego przesyłu danych w sieci. W celu wizualizacji zmieniających się dróg przesyłu urządzenia zostały przemieszczone kilkakrotnie co skutkowało zmianami w połączeniach. Urządzenia przesyłały dane do najbliższych sobie węzłów, a te przekazywały je dalej.



Rys. Wizualizacja przesyłu danych w topologii mesh

Zobrazowane sieci na powyższych schematach przedstawiają 4 urządzenia, kolejno każdy kolor reprezentuje to samo urządzenie. Dzięki temu w łatwy sposób możemy zaobserwować jak zmieniała się komunikacja w systemie w zależności od przemieszczenia się węzłów.

2.6.3. Czujnik temperatury

Jak wcześniej wspomniałem do systemu zostaną wdrożone czujniki temperatury. Są one bardzo proste w połączeniu i obsłudze. W celu sprawdzenia możliwości i działania tych czujników zaimplementuje prosty program na jedną z płytek Esp-32, który pozwoli na odczytywanie danych z czujników. W tym celu skorzystam z biblioteki Onewire oraz DallasTemperature.

Po wgraniu programu i uruchomieniu płytki możemy zauważyć działanie czujnika, który odbiera informacje i przesyła je na serial port.

```
12:42:02.222 -> Requesting temperatures...DONE
12:42:02.286 -> Temp C: 27.50 Temp F: 81.50
12:42:04.295 -> Requesting temperatures...DONE
12:42:04.371 -> Temp C: 27.50 Temp F: 81.50
```

Rys. Uzyskane wyniki pomiarów

W celu sprawdzenia zachowania czujnika w skrajnych sytuacjach ten został podłączony do źródła zasilania w sposób odwrotny co powoduje natychmiastowe przegrzanie się czujnika. Należy zadbać, aby taka sytuacja nie miała możliwości wystąpienia ze względu na prawdopodobne uszkodzenie czujnika.

W trakcie testów przystąpiłem do implementacji dodatkowych funkcjonalności na płytce ESP-32 takich jak jednocześnie uruchomienie klienta WiFi. Dzięki tym eksperymentom

zaobserwowałem znaczące opóźnienie działania programu poprzez ciągły odczyt danych z czujnika, którego czas trwania zbadałem eksperymentalnie poprzez umieszczenie timera w programie. Poniżej prezentuje uzyskany wynik:

```
12:47:51.731 -> Requesting temperatures...DONE
12:47:51.843 -> Temp C: 27.50 Temp F: 81.50
12:47:51.886 -> Invoking time: 591
```

Rys. Uzyskane wyniki pomiarów

Ze względu na znaczący, dla działania systemu, czas odczytu danych z czujnika i jednowątkowość stosowanych procesorów konieczne jest wdrożenie odczytu danych w pewnych odstępach czasowych. Skorzystanie z wbudowanej funkcji millis() i ustawienie odpowiednich odstępów czasowych pozwoli na uzyskanie tzw. pseudowielowątkowości. Dzięki takim działaniom odczyt danych odbywa się w pewnych interwałach co pozwala na sprawne działanie innych części programu, które nie są już spowalniane.

```
//definicja zmiennych czasowych
unsigned long currentTime = 0;
unsigned long timeDiff = 0;
unsigned long previousTime = 0;

void loop() {
    currentTime = millis(); //pobranie aktualnego czasu
    timeDiff = currentTime - previousTime; //wyliczenie różnicy

    if (timeDiff >= 1000UL) {
        Serial.print("Requesting temperatures...");
        sensors.requestTemperatures(); // Pozyskiwanie temperatury z czujników
        Serial.println("DONE");
        printTemperature(insideThermometer); // funkcja wypisująca w konsoli dane
        previousTime = currentTime; //nadpisanie czasu poprzedniego w celu wyznaczenia różnicy
    }
}
```

Rys. Przykład wdrożonych zmian pozwalających na sprawne działanie programu.

2.6.4. Przekazniki

Ostatnim komponentem sprzętowym jaki zostanie zastosowany przeze mnie podczas tworzenia tego projektu jest przekaznik. W celu sprawdzenia możliwości takiego przekaznika wykonant zostanie test sprawdzający szybkość jego reakcji i działanie podczas częstego przełączania. W tym celu przekaznik został podłączony zgodnie z kartą katalogową do źródła zasilania oraz płytki ESP-32. Dodatkowo na płytkę został wgrany program, który w konkretnych interwałach czasowych zmienia wartość prądu na jednym z pinów, do którego podłączony jest przekaznik. Eksperymenty wykazały, iż przekaznik reaguje natychmiastowo i jest bardzo odporny na szybkie zmiany wartości.

3. Projekt systemu

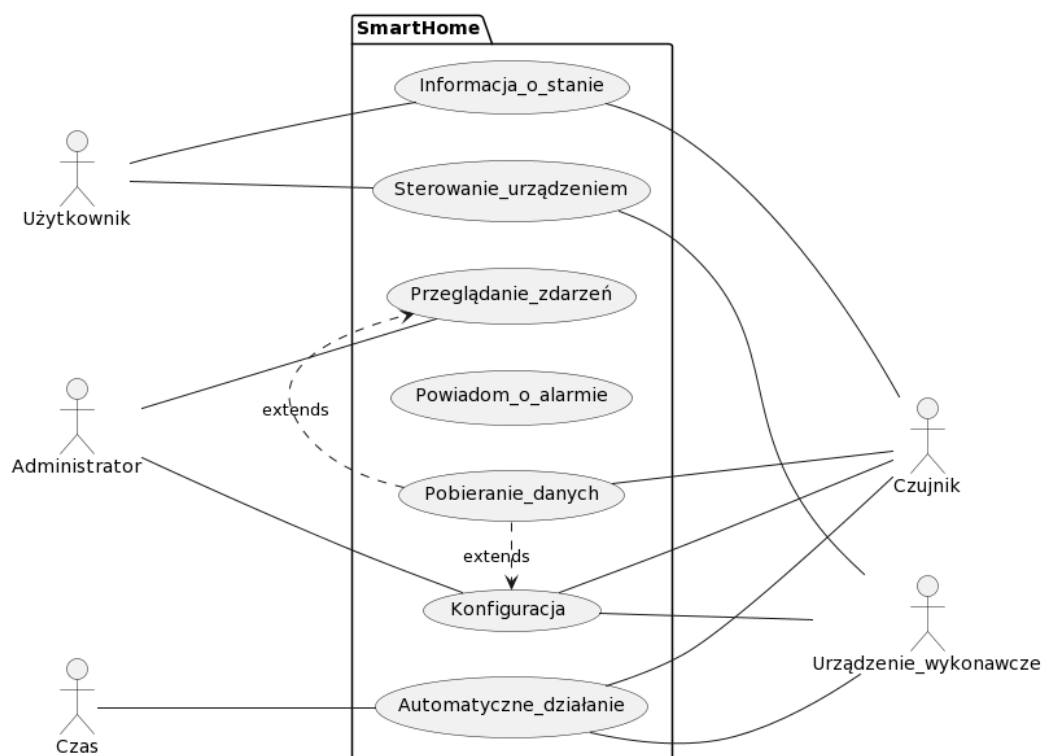
W poprzednim rozdziale omówione zostały dostępne na rynku rozwiązania i możliwość zastosowania ich podczas budowania systemu zarządzania urządzeniami elektrycznymi. Aby poprawnie dostosować projekt do jego zastosowań należy rozpatrzyć przypadki użycia, uwzględniając w nich ogólne zachowania użytkowników i komponentów w systemie. W pierwszej kolejności należy zastanowić się jakcy aktorzy obecni są w systemie.

Rys. 3.1. Aktorzy obecni w systemie

Na podstawie wstępnej analizy projektu wyróżnieni zostali powyżsi aktorzy składających się na system. Danym aktorom można przypisać następujące opisy:

- Administrator – osoba odpowiedzialna za zarządzanie systemem, zajmuje się konfiguracją urządzeń i ich obsługą w aplikacji webowej
- Użytkownik –
- Czujniki –
- Urządzenia wykonawcze –
- Czas –

W kolejnym kroku należy rozważyć przypadki użycia systemu tak, aby w odpowiedni sposób można było przystąpić do realizacji projektu. UseCase diagram jest kluczowym diagramem podczas tworzenia dokumentacji UML.



Rys. 3.2. UseCase diagram

Powyższy diagram jest ogólnym zaprezentowaniem procesów występujących w systemie. Posłuży on do projektowania scenariuszy przypadków użycia oraz przedstawieniu działania całego systemu. Podczas projektowania oprogramowania i fizycznych urządzeń konieczne będzie uwzględnienie tych przypadków i oparcie całego systemu o powyższy diagram.

3.1. Przypadki użycia systemu.

Aby lepiej zrozumieć idee systemu i powstające w nim procesy stworzone zostaną przypadki użycia. Przypadek użycia zwraca uwagę na przebieg konkretnej sytuacji w systemie. Pozwala na dogłębną analizę procesu zachodzącego w systemie i odpowiednie zaplanowanie oprogramowania. Składa się on z opisu aktorów, miejsca oraz scenariusza głównego jak i tych alternatywnych.

3.1.1. Logowanie do systemu (identyfikator: UC1.0)

Aktorzy:	Użytkownik
Zakres:	Aplikacja
Poziom:	Systemowy
Udziałowcy i ich cele:	Użytkownik chce zalogować się do systemu.
Zdarzenie wyzwalające (trigger):	Użytkownik otwiera aplikację
Warunki wstępne:	Użytkownik jest zarejestrowany w systemie.
Warunki końcowe dla sukcesu:	Użytkownik loguje się do systemu.
Warunki końcowe dla niepowodzenia:	Użytkownik nie może zalogować się do systemu.

Scenariusz główny:

1. Użytkownik otwiera aplikację.
2. Użytkownik wybiera użytkownika i wprowadza hasło.
3. Aplikacja komunikuje się z serwerem w celu weryfikacji.
4. Serwer pozytywnie weryfikuje użytkownika i pobiera uprawnienia.
5. Serwer przesyła informację do aplikacji.
6. Aplikacja wyświetla stronę główną.

Scenariusz alternatywny:

- 4.a.1. Serwer nie może zeweryfikować użytkownika.
- 4.a.2. Serwer przesyła informację o niepowodzeniu do aplikacji.
- 4.a.3. Aplikacja wyświetla informację o niepowodzeniu.

3.1.2. Konfiguracja urządzenia przez użytkownika (identyfikator: UC1.1)

Aktorzy:	Administrator, czujnik/urządzenie wykonawcze
Zakres:	Aplikacja
Poziom:	Administracji budynku
Udziałowcy i ich cele:	Administrator chce dodać skonfigurować urządzenie
Zdarzenie wyzwalające (trigger):	Administrator otwiera listę urządzeń
Warunki wstępne:	Administrator musi być zalogowany i posiadać uprawnienia
Warunki końcowe dla sukcesu:	Ustawienia urządzenia zostaną zmienione.
Warunki końcowe dla niepowodzenia:	Ustawienia urządzenia nie zostaną zmienione, urządzenie wyświetlać się będzie jako nieaktywne

Scenariusz główny:

1. Administrator przechodzi do zakładki "Urządzenia" w menu głównym.
2. System wyświetla dostępne urządzenia i opcje dodania nowych.
3. Administrator wybiera opcję dodania urządzenia.
4. System dodaje urządzenie i kończy konfigurację.
5. Administrator wprowadza IP urządzenia.
6. System pobiera właściwości konkretnego urządzenia i przypisuje mu typ.
7. System wyświetla formularz dla danego typu urządzenia.
8. Administrator uzupełnia brakujące dane.
9. System zapisuje konfigurację i konfiguruje urządzenie w topologię systemu.

Scenariusz alternatywny:

- 3.a.1. Administrator wybiera urządzenie z listy
- 3.a.2. System pobiera dane o typie urządzenia.
- 3.a.3. System wyświetla odpowiedni formularz pozwalający na edycję urządzenia.

Scenariusz alternatywny:

- 8.a. Urządzenie nie odpowiada na zapytania systemu
- 8.a.1 System wyświetla informację o niepowodzeniu lub błędach w komunikacji.

3.1.3. Połączenie urządzenia w topologię mesh (identyfikator: UC1.2)

Aktorzy:	Urządzenie wykonawcze/czujnik, serwer
Zakres:	Aplikacja, system
Poziom:	Administracji budynku
Udziałowcy i ich cele:	System chce wdrożyć urządzenie do topologii mesh
Zdarzenie wyzwajające (trigger):	Administrator dodaje urządzenie w aplikacji.
Warunki wstępne:	Urządzenie jest dodane do systemu.
Warunki końcowe dla sukcesu:	Urządzenie zostaje zintegrowane w siatce urządzeń.
Warunki końcowe dla niepowodzenia:	Urządzenie nie mogło zostać zintegrowane.

Scenariusz główny:

1. Administrator dokonuje dodania urządzenia do systemu.
2. Urządzenie zostaje automatycznie zintegrowane z innymi urządzeniami w sieci dzięki implementacji z użyciem odpowiednich bibliotek.
3. Urządzenie przesyła swoje informacje identyfikacyjne do serwera.
4. Serwer inicjuje przesłanie informacji o nowym urządzeniu w całej sieci.
5. Urządzenia w sieci odbierają informację o nowym urządzeniu.
6. Urządzenia w bezpośrednim otoczeniu nowego urządzenia nawiązują z nim bezpośrednie połączenie.

Scenariusz alternatywny:

- 2.a.1. Urządzenie nie może nawiązać połączenia z innymi urządzeniami w sieci.
- 2.a.2. System informuje o niepowodzeniu.
- 2.a.3. Konfiguracja zostaje przerwana.

Scenariusz alternatywny:

- 6.a. Żadne urządzenie nie znajduje się w wystarczająco bliskim położeniu do nowego urządzenia.
- 6.a.1 System informuje o niepowodzeniu.

3.1.4. Przesył danych pomiędzy urządzeniami (topologia mesh) (identyfikator: UC1.3)

Aktorzy:	Urządzenia, czas
Zakres:	System
Poziom:	Systemowy
Udziałowcy i ich cele:	Urządzenia chcą przesyłać informację w topologii siatki
Zdarzenie wyzwalające (trigger):	Zmiana stanu czujnika.
Warunki wstępne:	Urządzenia są skonfigurowane w systemie
Warunki końcowe dla sukcesu:	Informacje zostają przesłane pomiędzy urządzeniami.
Warunki końcowe dla niepowodzenia:	Wszystkie urządzenie skonfigurowane w systemie otrzymują informację.

Scenariusz główny:

1. Stan czujnika zmienia się w czasie.
2. Urządzenie przesyła informację poprzez topologię mesh do innych urządzeń.
3. Zgodnie z zasadami działania tej sieci informacja jest dostarczana do pozostałych urządzeń i serwera.
4. W zależności od stanu jaki został zmieniony wykonywane są kolejne kroki.

Scenariusz alternatywny:

- 2.a.1. Urządzenie nie może połączyć się z innymi urządzeniami.
- 3.a.2. Przesył danych kończy się niepowodzeniem.

Scenariusz alternatywny:

- 4.a.1. Zmiana stanu nie wpływa na działanie systemu.
- 4.a.2. Kolejne kroki nie są podejmowane.

3.1.5. Sterowanie oświetleniem z wykorzystaniem fizycznych przełączników (identyfikator: UC1.4)

Aktorzy:	Urządzenie wykonawcze, przełącznik, użytkownik
Zakres:	System
Poziom:	Systemowy
Udziałowcy i ich cele:	Użytkownik chce włączyć/wyłączyć oświetlenie.
Zdarzenie wyzwajające (trigger):	Nacisnięcie przełącznika.
Warunki wstępne:	Urządzenia są skonfigurowane w systemie
Warunki końcowe dla sukcesu:	Stan oświetlenia zostanie zmieniony.
Warunki końcowe dla niepowodzenia:	Stan oświetlenia nie ulegnie zmianie.

Scenariusz główny:

1. Użytkownik naciska przełącznik.
2. Przełącznik przesyła dane do wszystkich urządzeń i serwera z wykorzystaniem topologii mesh.
3. Urządzenie wykonawcze po odebraniu informacji zmienia swój stan w zależności od konfiguracji.
4. Urządzenie wykonawcze rozsyła informację na temat swojego stanu do wszystkich urządzeń i serwera z wykorzystaniem topologii mesh.

Scenariusz alternatywny:

- 2.a.1. Urządzenie nie zmienia swojego stanu zgodnie z założeniami konfiguracji.
- 2.a.2. System wyświetla informację o wadliwej konfiguracji.

Scenariusz alternatywny:

- 4.a.1. Informacja nie dociera do wszystkich urządzeń skonfigurowanych w systemie.
- 4.a.2. System wyświetla informację o błędach w komunikacji.

3.1.6. Manualne sterowanie temperaturą (identyfikator: UC1.5)

Aktorzy:	Użytkownik, urządzenie grzewcze/chłodzące
Zakres:	Aplikacja
Poziom:	Systemowy
Udziałowcy i ich cele:	Użytkownik chce zmienić działanie urządzenia.
Zdarzenie wyzwalające (trigger):	Użytkownik wybiera urządzenie.
Warunki wstępne:	Użytkownik musi być zalogowany, posiadać uprawnienia do pomieszczenia, urządzenie musi być podłączone i w pełni działać
Warunki końcowe dla sukcesu:	Funkcja urządzenia zostaje zmieniona
Warunki końcowe dla niepowodzenia:	Funkcja urządzenia nie zostaje zmieniona, komunikacja została przerwana

Scenariusz główny:

1. Użytkownik z listy urządzeń danego pomieszczenia wybiera urządzenie grzewcze/chłodzące.
2. System pobiera informację o danym urządzeniu.
3. System wyświetla informację i możliwe interakcje z urządzeniem(properties).
4. Użytkownik wybiera opcję włączenia lub wyłączenia urządzenia.
5. System przesyła do urządzenia zadane polecenie(typ).
6. Urządzenie przyjmuje polecenie i wykonuje zgodnie z implementacją.
7. Informacja o wykonanej czynności jest zwracana do systemu
8. System wyświetla informację o wykonywanej czynności.

Scenariusz alternatywny:

- 6.a. Urządzenie nie przyjmuje informacji w sposób poprawny
- 6.a.1 System informuje o niepoprawnym działaniu.

Scenariusz alternatywny:

- 3.a. Funkcja nie może być zmieniona ze względu na ustawienie stałej funkcji.
- 3.b.1 System informuje użytkownika o takiej sytuacji.

3.1.7. Manualne sterowanie oświetleniem (identyfikator: UC1.6)

Aktorzy:	Użytkownik, urządzenie grzewcze/chłodzące
Zakres:	Aplikacja
Poziom:	Systemowy
Udziałowcy i ich cele:	Użytkownik chce zmienić działanie urządzenia.
Zdarzenie wyzwalające (trigger):	Użytkownik wybiera urządzenie.
Warunki wstępne:	Użytkownik musi być zalogowany, posiadać uprawnienia do pomieszczenia, urządzenie musi być podłączone i w pełni działać
Warunki końcowe dla sukcesu:	Funkcja urządzenia zostaje zmieniona
Warunki końcowe dla niepowodzenia:	Funkcja urządzenia nie zostaje zmieniona, komunikacja została przerwana

Scenariusz główny:

Scenariusz alternatywny:

6.a. Urządzenie nie przyjmuje informacji w sposób poprawny

6.a.1 System informuje o niepoprawnym działaniu.

Scenariusz alternatywny:

3.a. Funkcja nie może być zmieniona ze względu na ustawienie stałej funkcji.

3.b.1 System informuje użytkownika o takiej sytuacji.

3.1.8. Zmiana harmonogramu pracy urządzeń sterujących (identyfikator: UC1.7)

Aktorzy:	Urządzenie wykonawcze, czujnik, czas
Zakres:	Aplikacja
Poziom:	Systemowy
Udziałowcy i ich cele:	Administrator chce zaplanować działanie urządzeń.
Zdarzenie wyzwajające (trigger):	Administrator ustawia zadanie cykliczne.
Warunki wstępne:	Administrator musi być zalogowany i posiadać uprawnienia.
Warunki końcowe dla sukcesu:	Zostaje zapisany nowy tryb pracy.
Warunki końcowe dla niepowodzenia:	Zapis trybu pracy nie powiódł się.

Scenariusz główny:

1. Użytkownik wybiera urządzenie opisane jako sterownik temperatury dla danego pomieszczenia.
2. System wyświetla obecną konfigurację dla urządzenia.
3. Użytkownik ustawia tryb pracy, temperaturę i inne konieczne właściwości np. ustawia godziny w jakich pracuje klimatyzacja/ogrzewanie.
4. System zapisuje konfigurację i przesyła ją do urządzeń i czujników
5. Użytkownik jest informowany o przebiegu konfiguracji.

Scenariusz alternatywny:

- 5.a. Komunikacja z urządzeniami nie powiodła się
- 5.a.1 System informuje użytkownika o niepowodzeniu

3.1.9. Odbiór danych z czujnika (identyfikator: UC1.8)

Aktorzy:	Czujnik, aplikacja
Zakres:	Urządzenie fizyczne, aplikacja webowa
Poziom:	Systemowy
Udziałowcy i ich cele:	Czujnik chce wysłać pomiar.
Zdarzenie wyzwalające (trigger):	Czujnik wysyła pomiar do serwera.
Warunki wstępne:	Urządzenie fizyczne musi być włączone do systemu.
Warunki końcowe dla sukcesu:	Wartość pomiaru jest przesyłana do serwera.
Warunki końcowe dla niepowodzenia:	Wartość pomiaru nie dociera do serwera.

Scenariusz główny:

1. Czujnik odbiera dane i przesyła je do urządzenia sterującego.
2. Urządzenie sterujące przesyła dane do serwera lub urządzenia pośredniczącego z wykorzystaniem zaprogramowanego protokołu.
3. Serwer odbiera dane od urządzenia fizycznego bądź urządzenia, które pośredniczy w przesyłaniu danych.
4. Serwer wystawia API dla aplikacji webowej.
5. Aplikacja webowa tworzy zapytanie do serwera i odbiera dane na temat temperatury
6. Aplikacja webowa wyświetla odebrane parametry.

Scenariusz alternatywny:

- 2.a.1. Urządzenie nie może skomunikować się z serwerem bądź innym urządzeniem.
- 2.a.2. Przesył danych nie powiódł się.

Scenariusz alternatywny:

- 5.a.1. Nastąpił problem w komunikacji z serwerem
- 5.a.2. Odczyt API nie powiódł się.

3.1.10. Automatyczna kontrola temperatury. (identyfikator: UC1.9)

Aktorzy:	Urządzenie sterujące, czujnik, czas
Zakres:	Urządzenie fizyczne, serwer
Poziom:	Systemowy
Udziałowcy i ich cele:	System chce wysłać utrzymać zaplanowaną temperaturę.
Zdarzenie wyzwalające (trigger):	Czujnik wysyła pomiar do serwera.
Warunki wstępne:	Urządzenia fizyczne muszą być włączone do systemu. Oczekiwana temperatura musi być ustawiona przez użytkownika.
Warunki końcowe dla sukcesu:	Zaplanowana wcześniej temperatura zostaje uzyskana/utrzymana.
Warunki końcowe dla niepowodzenia:	Czujnik nie przesyła poprawnych danych. Urządzenie sterujące nie działa poprawnie.

Scenariusz główny:

1. Czujnik, w czasie, kontroluje wartość temperatury w konkretnym pomieszczeniu.
2. Temperatura jest inna niż oczekiwana (0.4 stopnia różnicy) więc system podejmuje działania.
3. Urządzenie grzewcze/chłodzące zostaje uruchomione.
4. Temperatura z czujnika spełnia wymagania zadanej temperatury.
5. Urządzenie grzewcze/chłodzące zostaje wyłączone.
6. System w czasie kontynuuje kontrole temperatury.

Scenariusz alternatywny:

- 2.a.1. Temperatura oczekiwana jest zgodna z temperaturą rzeczywistą.
- 2.a.2. System w czasie kontynuuje kontrole temperatury.

Scenariusz alternatywny:

- 5.a.1. Nastąpił problem w komunikacji z urządzeniami.
- 5.a.2. Urządzenie jest niedostępne.

3.2. Architektura systemu

W rozdziale 2 omówione zostały technologie, które stosowane są podczas tworzenia omawianego rozwiązania. Architektura tworzonego przeze mnie systemu zostanie oparta właśnie o te technologiczne rozwiązania.

Projektowanie systemu należy rozpocząć od omówienia wymagań technologicznych co zostało rozpisane wraz z przypadkami użycia w poprzednim podrozdziale. W dalszej kolejności konieczne jest zaprojektowanie systemu w taki sposób by był w stanie spełniać te wymagania.

Tworzony w ramach pracy dyplomowej system składał będzie się z:

- Urządzeń elektronicznych/elektrycznych – posłużą one do obsługi dowolnych urządzeń wykonawczych, umożliwiając implementację w systemie.
- Serwera – program działający w ramach maszyny (komputer/serwer zdalny), odpowiadający za komunikację pomiędzy częścią UI (interfejs użytkownika), a urządzeniami fizycznymi,
- Aplikacji webowej – służącej w głównej mierze do konfiguracji systemu,
- Aplikacji mobilnej – służącej do kontrolowania połączonych z systemem urządzeń,

Rys. 3.3. Schemat komunikacji w systemie.

3.2.1. Urządzenia sterujące

Jedną z najbardziej kluczowych części systemu są urządzenia sterujące. Jak zostało to opisane w rozdziale 2, oparte zostaną o płytkę ESP8266, która posiada wbudowany moduł Wifi. Właśnie w ten sposób odbywać się będzie komunikacja płytki z serwerem i innymi urządzeniami. Kombinacja płytki wraz z przekaźnikiem elektrycznym odbywać się będzie z wykorzystaniem wbudowanych do ESP pinów pozwalających na generowanie sygnałów wartości 5V. Właśnie tyle, do sterowania, potrzebuje przekaźnik SRD-5VDC-SL-C pozwalający do sterowania urządzeniami do 250V prądu stałego lub przemiennego. Taka wariancja napięcia pozwala na podłączenie praktycznie każdego urządzenia zewnętrznego jakie stosowane są w rozwiązaniach typu Smart Home.

Kluczem do poprawnego działania tego urządzenia wraz z całym systemem jest odpowiednia implementacja oprogramowania na procesorze. Płytkę została wyposażona w system pozwalający na implementację oprogramowania z wykorzystaniem Arduino IDE, które jest jednym z najbardziej popularnych programów do programowania mikrokontrolerów. Podczas tworzenia oprogramowania należy zadbać o poprawne adresowanie pinów służących do sterowania przekaźnikiem lub odczytu danych z czujnika w zależności od typu urządzenia oraz odpowiednia implementacja komunikacji z innymi urządzeniami i serwerem. Dokładna komunikacja, zastosowane metody i protokoły zostaną opisane w kolejnym podrozdziale, w którym omówię pracę serwera.

Rys. 3.4. Schemat elektryczny urządzenia sterującego.(Dodany zostanie schemat)

3.2.2. Serwer i komunikacja

Jak zaprezentowane zostało na Rys.3.3. serwer jest centrum całego systemu. Można porównać go do mózgu w organizmach żyjących. Odpowiada za odbiór, przetwarzanie i przekazywanie informacji. To dzięki niemu jesteśmy w stanie skomunikować dwa tak różne środowiska jakimi są aplikacje dla użytkowników oraz urządzenia fizyczne. W pierwszej kolejności należy rozważyć komunikację takiego serwera z mikrokontrolerami oraz aplikacjami.

Te pierwsze, mikrokontrolery, wcześniej opisywane szerzej jako urządzenia fizyczne potrzebują komunikacji z serwerem w celu otrzymywania rozkazów od użytkowników, którzy posługiwać się będą aplikacjami z interfejsem użytkownika. Sam serwer dzielić się będzie na dwie części. Warstwa logiki działania urządzeń wykonawczych oraz warstwa dla działania aplikacji tj. część odpowiedzialna za konfigurację urządzeń po stronie użytkownika. Podobnie zachowywać się będzie warstwa komunikacji. Ze względu na konieczność nieprzerwanego działania urządzeń komunikacja z serwerem będzie musiała odbywać się poprzez technologię WebSocket. Zastosowanie takiego rozwiązania pozwala na nieustanną „rozmowę” urządzeń z serwerem i odwrotnie. Jest to rozwiązanie pozwalające na działanie in-real-time. Wyżej wspomnianą komunikację możemy uzyskać poprzez wykorzystanie dedykowanej biblioteki po stronie mikrokontrolera oraz dedykowana, wcześniej wspomniana, bibliotekę socket.io, po stronie serwera. Dodatkowo konieczne będzie podłączenie urządzeń do bezprzewodowej sieci. W tym celu skorzystać należy z dedykowanej biblioteki do łączenia z siecią WiFi. Konieczne będzie również nadanie adresu IP oraz mac dla urządzeń w celu prostej identyfikacji ich we wcześniej wspomnianej sieci. Serwer jak i urządzenia między sobą będą się rozróżniać dzięki wykorzystaniu tych adresów. Sama komunikacja, jak wcześniej wspomniałem, oparta będzie o websocketsy czyli technologię zbudowaną na protokole TCP/IP. Ten model został oparty o model OSI lecz liczba warstw, kosztem ich rozmiarów, została zmniejszona. Komunikacja polega na wykorzystaniu adresów IP urządzeń. Zapewnia dostarczenie danych pomiędzy przesyłającymi bez utraty pakietów co sprawia, iż system staje się bardziej stabilny i odporny na awarie. [adnotacja do tcp/ip]

Patrząc na drugą stronę systemu, aplikacje, tutaj sprawa jest znacznie prostsza. Nie istnieje tutaj potrzeba nieustannej komunikacji serwera z aplikacją, ponieważ dane na temat temperatury itp. mogą być przesyłane w zadanych odstępach czasowych. Jednakże konieczna jest tutaj natychmiastowa komunikacja na linii aplikacja – serwer ze względu na konieczność natychmiastowego przesyłania rozkazów użytkownika. Aby osiągnąć dwa powyższe założenia możemy skorzystać z technologii API, która omówiona dokładniej została w rozdziale 2. Pozwala nam na przesyłanie get, post i put requestów z poziomu aplikacji webowej lub mobilnej. Get requesty pozwolą na odbiór informacji z serwera w konkretnych interwałach, post i put requesty pozwolą na przesyłanie informacji do serwera, a w konsekwencji do urządzeń fizycznych pozwalając na ich uruchomienie. Jeżeli chodzi o stronę backend'u, to get, post, put i inne metody API są wbudowane w silnik i możliwe do stosowania praktycznie od

razu.[adnotacja do dokumentacji] Komunikacja API ma znaczące zalety jakim jest stabilność, brak utraty danych i możliwość wielokrotnych zapytań w celu uzyskania danych w konkretnej chwili czasu.

3.2.3. Aplikacja webowa i mobilna

Kolejnym już, ostatnim krokiem w tworzeniu systemu jest aplikacja webowa i mobilna. Przeznaczenie tych dwóch jest dość oczywiste oraz bardzo ważne. Służą one użytkownikowi do komunikowania się z systemem. Projektując aplikacje dla użytkowników proces dzielimy na dwa etapy. Pierwszy z nich to zaprojektowanie logiki aplikacji w tym komunikacji, co pozwoli na komunikację z resztą systemu, a drugi to projektowanie interfejsu użytkownika. Jeżeli chodzi o komunikację całego systemu należy tutaj powrócić do rozdziału drugiego gdzie omówiona została komunikacja z wykorzystaniem REST API w technologii React. Z pomocą w tym przypadku przychodzi biblioteka axios z pakietu NPM. Pozwala ona na bardzo proste i bezawaryjne wywoływanie post, put i get requestów, czyli wszystkiego co potrzebne do komunikacji z serwerem. Get requesty posłużą nam do odbierania danych z serwera na podstawie temperatury, post i put requesty będą stanowić dwa typy wywołań. Jeden służył będzie konfiguracji urządzeń w systemie, a drugi odpowadał będzie za requesty sterujące urządzeniami już skonfigurowanymi w systemie. Jak zostało wspomniane wcześniej, przesył danych z wykorzystaniem REST API odbywa się z użyciem danych w formacie JSON. Pozwoli to na prosty odczyt plików przez serwer.

Drugą stroną odmienną od logiki aplikacji jest jej interfejs. Głównymi założeniami dla interfejsu jest dostarczenie użytkownikowi narzędzia do sterowania systemem. W tworzonym w ramach pracy systemie zaplanowane zostały konkretne interfejsy:

- Okno logowania – tutaj administrator/użytkownik będzie mógł się zalogować do systemu. W zależności od zalogowanego konta, użytkownicy będą posiadać inne prawa.
- Okno widoku dostępnych urządzeń – w tym przypadku użytkownik będzie posiadał możliwość przeglądania wszystkich dostępnych urządzeń i wyboru każdego z nich w celu sterowania nim.
- Okno konfiguracji urządzeń – panel dla administratora, w którym dodawane i konfigurowane będą mogły być nowe urządzenia.

Kolejnym krokiem w projektowaniu aplikacji jest przedstawienie wstępnego wizualnego zarysu interfejsów, zostanie to dokonane w późniejszych podrozdziałach.

3.3. Zachowania systemu

W poprzednim podrozdziale opisane zostały podsystemy całego projektu z uwzględnieniem ich komunikacji i architektury. Zwieńczeniem całego projektu jest stworzenie odpowiedniego kodu dla urządzeń, serwera i aplikacji. Programowanie będzie oparte o wcześniej założone cele,

stworzone scenariusze jak i opisy komunikacji. Dlatego jako ostatnie stworzone zostaną programy obsługujące cały system.

3.3.1. Logowanie użytkownika

Aby użytkownik w bezpieczny sposób mógł korzystać z systemu udostępniona zostanie opcja logowania co pozwoli zabezpieczyć system przed zewnętrznymi niebezpieczeństwami. Logowanie zostanie zrealizowane w sposób uproszczony, aby pominąć stosowanie bazy danych realacyjnych. W momencie instalacji systemu administrator udostępni wygenerowane przez system kody dostępu, którymi następnie użytkownik będzie miał możliwość się zalogować. Na całość procesu logowania składać się będzie serwer, czyli strona weryfikująca poprawność danych oraz interfejs użytkownika czyli aplikacja, która pozwoli na uzupełnienie danych i przesłanie ich do serwera. Serwer po weryfikacji udostępni token, który pozwoli na wykonywanie kolejnych zapytań do API i sterowanie systemem. Po stronie aplikacji token będzie przechowywany w session storage co pozwoli na łatwy dostęp do niego i możliwość wykorzystania w każdym momencie.

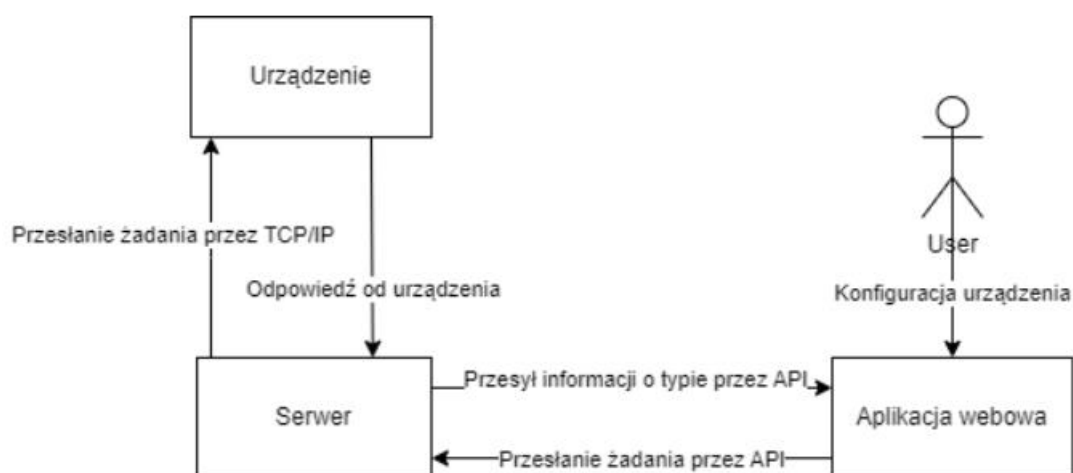
Proces logowania i pozyskania tokenu odgrywa kluczową rolę. Zabezpiecza system przed niechcianymi zapytaniami do API, które będą niemożliwe przy braku tokenu lub w momencie gdy token będzie nieaktualny. Architektura każdego zapytania do API oparta będzie o wykorzystanie tego właśnie tokenu, który przesyłany w tzw. headers zapytań będzie zapewniał dostęp do zasobów serwera.

3.3.2. Konfiguracja urządzeń

Podstawą całego systemu jest konfiguracja nowych urządzeń. Każde z nich, będzie posiadać swój unikalny identyfikator nadany podczas produkcji, dzięki temu będzie można łatwo rozróżniać je między sobą. Urządzenia wyposażone zostaną w program pozwalający na wybranie typu urządzenia z kilku istniejących, dzięki temu otrzymamy możliwość konfiguracji urządzenia na różne sposoby. Konfiguracja dla konkretnych typów zostanie stworzona w taki sposób, aby jedno urządzenie było przystosowane do obsługi kilku typów takich jak:

- Czujnik np. Temperatury, wilgotności itp.
- Urządzenie wykonawcze np. Światło, żaluzje
- Hybryda tzn. Czujnik temperatury i termostat

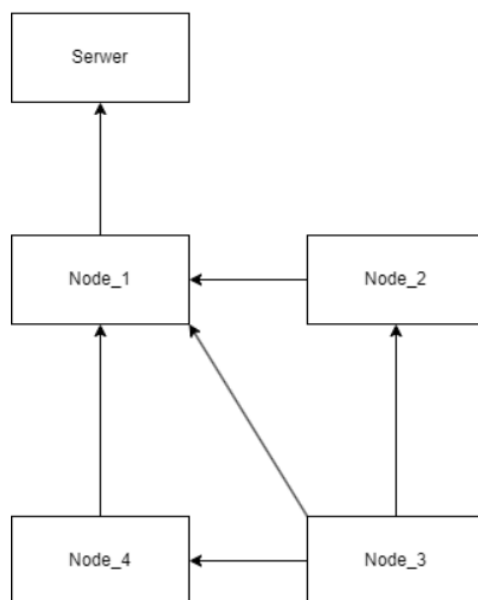
Zdefiniowanie powyższych typów wystąpi po stronie procesora płytki, jednakże wybierane one będą z poziomu aplikacji webowej co przedstawione jest na schemacie poniżej.



Rys.3.5. Schemat działania konfiguracji

3.3.3. Przesył danych pomiędzy urządzeniami

Jak wspomniane zostało w poprzednich rozdziałach cały system oparty zostanie o topologię siatki (mesh). Zalety powyższej topologii są już szeroko znane, w celu jej realizacji zastosowana zostanie biblioteka `painlessMesh Library`, która w prosty sposób pozwoli na zainicjowanie sieci i umożliwi implementację przesyłania danych pomiędzy urządzeniami znajdującymi się w sieci. Przesył danych pomiędzy urządzeniami będzie odbywał się w momencie gdy wartości czujników zmienią się lub w momencie gdy użytkownik zainicjuje ich zmianę. Przykładową sytuacją może być tutaj zmiana temperatury. Zaimplementowany czujnik temperatury w systemie pobiera wartość temperatury w interwałach czasowych np. 60s. Jeżeli nowa temperatura będzie różna od poprzedniej zostanie ona nadpisana i informacja o jej zmianie wysłana do innych urządzeń, w tym do serwera.



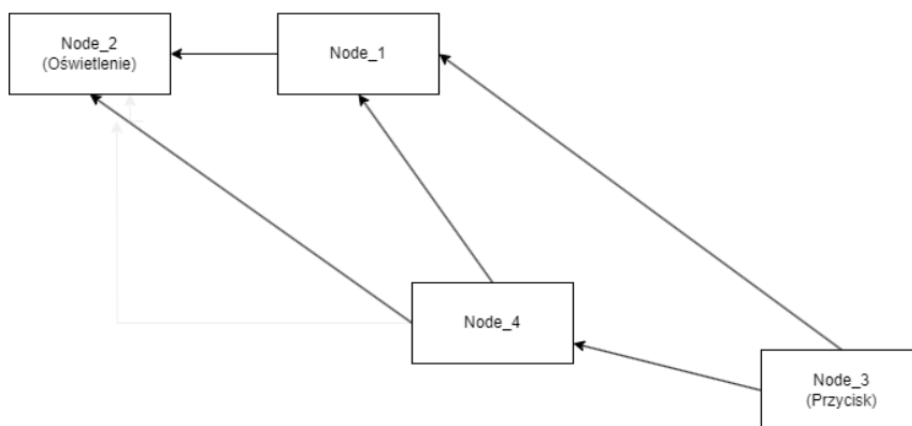
Rys. 3.6. Wizualizacja przesyłania danych.

Patrząc na powyższy schemat w urządzeniu Node_3 następuję odczyt temperatury, który powoduję zmianę temperatury w systemie więc Node_3 inicjuje emitowanie danych do innych węzłów siatki. Zakładając, że węzeł Node_2 będzie urządzeniem grzewczym to po otrzymaniu informacji o zmianie temperatury, jeżeli dana temperatura będzie inna niż założona temperatura minimalna to urządzenie rozpocznie proces grzania, aż do czasu, w którym odczytana temperatura na urządzeniu Node_2 będzie zgodna z tą ustawioną dla danego pokoju.

Każde z urządzeń, które znajdować się będzie w systemie zostanie przystosowane w odpowiedni sposób do swojego działania. W zależności od zastosowania software na każdym z urządzeń będzie się nieznacznie różnił w zależności czy będzie to urządzenie sterujące np. Światłem bądź ogrzewaniem lub urządzenie pomiarowe np. czujnik temperatury bądź zwykły przycisk.

3.3.4. Sterowanie urządzeniami z wykorzystaniem fizycznych komponentów.

Kolejną ważnym zachowaniem systemu jest sterowanie urządzeniami w sieci z wykorzystaniem innych fizycznych urządzeń. Dobrym przykładem jest sterowanie oświetleniem. Dzięki zaimplementowanej topologii mesh, urządzenia, które znajdują się w systemie nie będą musiały komunikować się z serwerem w celu podjęcia pracy. Po uprzedniej konfiguracji, tzn. połączeniu urządzeń ze sobą np. przycisku oraz oświetlenia, urządzenie po naciśnięciu prześle dane do wszystkich innych urządzeń w sieci. Dzięki temu, urządzenie, które odpowiedzialne jest za sterowanie światłem odbierze informację na temat zmiany stanu przycisku i w dalszej kolejności będzie mogło uruchomić oświetlenie. Taka operacja jest możliwa dzięki połączeniu dwóch urządzeń, które jest niczym innym jak wyczuleniem urządzenia sterującego na zmianę statusu czujnika. Sytuacja wyglądać będzie identycznie przy sterowaniu manualnym temperaturą gdzie po naciśnięciu przycisku dane zostaną rozesłane i urządzenie, które połączone jest z przyciskiem i wyczułone na jego stan podejmie lub zaprzestanie pracy. Całość procesu jest dobrze widoczna na poniższym schemacie.



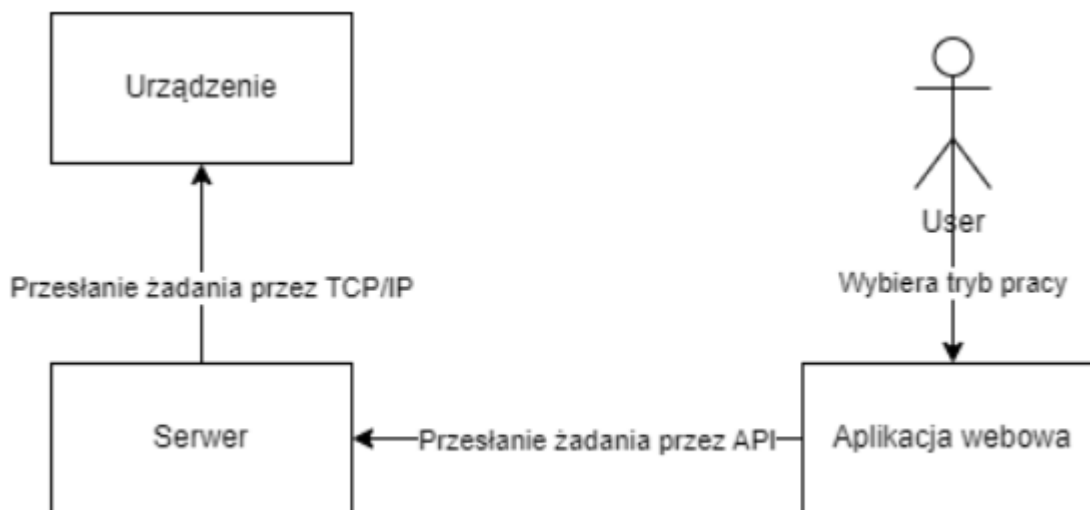
Rys. 3.7. Schemat sterowania urządzeniem

Jak możemy zaobserwować dane przesyłane na schemacie są w topologii mesh, pomimo iż zasięg sieci Node_3 nie pozwala na połączenie z Node_2 to dane docierają do niego. Pozwala to na uruchomienie oświetlenia po naciśnięciu przycisku. Warunkiem jest tutaj oczywiście zlinkowanie tych dwóch urządzeń z wykorzystaniem aplikacji webowej.

3.3.5. Sterowanie urządzeniami z wykorzystaniem aplikacji.

Pomysłna konfiguracja da nam możliwość sterowania konkretnymi urządzeniami. Ze względu na różnorodność typów urządzeń konieczne jest stworzenie osobnych typów interfejsów dla każdego z nich w taki sposób aby po otwarciu okna sterowania każdego z urządzeń wyświetlił nam się panel sterowania przystosowany do jego typu. Z serwera będziemy otrzymywać informacje o typie urządzenia co pozwoli nam wyświetlić odpowiedni panel do sterowania. Wygląd panelu zaprogramowany będzie po stronie aplikacji.

Użytkownik po wyświetleniu odpowięgo panelu sterowania będzie posiadał możliwość sterowania urządzeniem bądź odczytu danych z niego. Odczyt danych nie jest niczym skomplikowanym i został opisany w poprzednich podrozdziałach. Jeżeli chodzi o sterowanie urządzeniem, w tym przypadku użytkownik będzie wybierał z dostępnego panelu opcje udostępnione przez system np. włączenie światła.



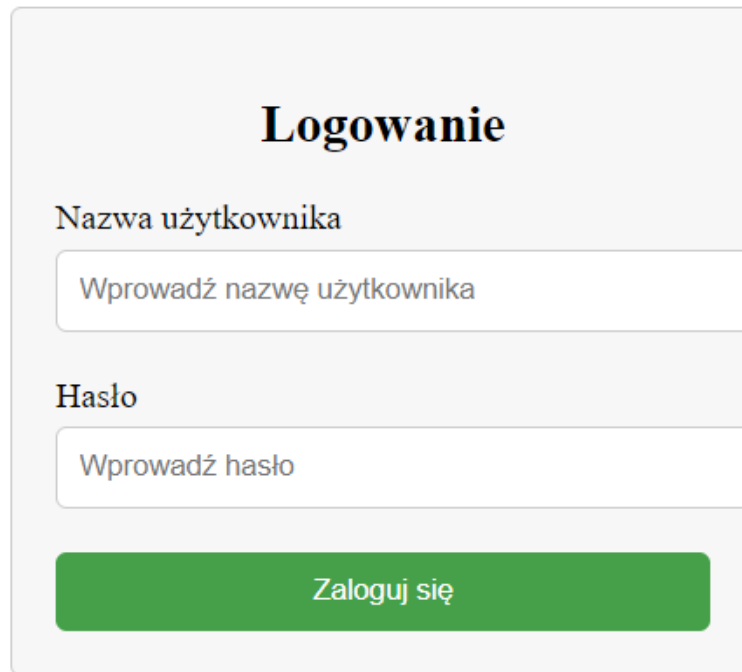
Rys. 3.8. Schemat sterowania urządzeniem

Następnie po wyborze odpowiedniej funkcji serwer będzie informowany o przebiegu zdarzenia z wykorzystaniem POST/PUT requesta. W następnej kolejności zadaniem serwera będzie odpowiednie przetworzenie zapytania i komunikacja z odpowiednim urządzeniem poprzez zaimplementowaną logikę dla topologii siatki.

3.4. Interfejsy użytkownika

Aby system spełniał oczekiwania klienta ważną kwestią jest dobre zaplanowanie interfejsów użytkownika. Poniżej zaprezentowane zostaną bardzo uproszczone interfejsy użytkownika, które zostaną rozbudowane i upiękkszzone podczas tworzenia aplikacji.

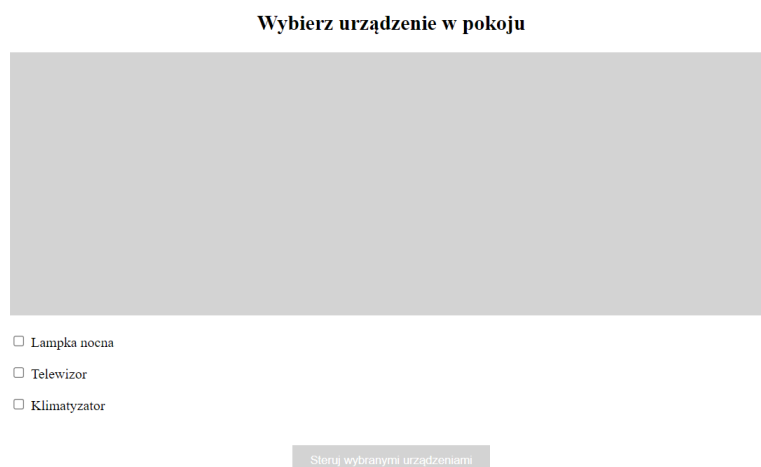
3.4.1. Okno logowania



The mockup shows a light gray rectangular window with rounded corners. At the top center, the title "Logowanie" is displayed in a bold, black, serif font. Below the title, the label "Nazwa użytkownika" is positioned above a white text input field with a light gray border and placeholder text "Wprowadź nazwę użytkownika". Further down, the label "Hasło" is positioned above another white text input field with a light gray border and placeholder text "Wprowadź hasło". At the bottom center of the window is a solid green rectangular button with the white text "Zaloguj się".

Rys.3.7. Okno logowania

3.4.2. Okno widoku dostępnych urządzeń



The mockup shows a light gray rectangular window with rounded corners. At the top center, the title "Wybierz urządzenie w pokoju" is displayed in a bold, black, serif font. Below the title is a large, empty gray rectangular area intended for displaying device options. At the bottom left of the window, there is a list of three items, each preceded by an unchecked checkbox: "Lampka nocna", "Telewizor", and "Klimatyzator". At the bottom center of the window is a light gray rectangular button with the text "Steruj wybranymi urządzeniami".

Rys.3.8. Okno wyboru urządzenia

3.4.3. Okno konfiguracji urządzeń

Urządzenia

Nazwa	IP	Typ
Urządzenie 1	192.168.0.1	Lampka
Urządzenie 2	192.168.0.2	Termostat

Dodaj urządzenie

Rys.3.9. Okno konfiguracji urządzeń

Dodaj urządzenie

IP urządzenia:

Typ urządzenia:

Lampka

Lampka

Termostat

Dodaj

Rys.3.10. Okno konfiguracji pojedynczego urządzenia

3.4.4. Okno sterowania urządzeniem

Steruj urządzeniem

device-status: On ▼

Jasność:

Temperatura:

Zapisz

Rys.3.11. Okno sterowania urządzeniem

Bibliografia

1. <https://new.abb.com/low-voltage/products/building-automation/product-range/abb-freeathome>, [Dostęp: 2.11.2022]
2. „Grenton_Technical_WiringDiagrams_PL_v.1.0.1”, Grenton, 2018
3. “What is KNX?”, <https://www.knx.org/knx-en/for-professionals/What-is-KNX/A-brief-introduction/>, [Dostęp: 10.11.2022]
4. "Wireless Mesh Networking: An IoT-Oriented Perspective Survey on Relevant Technologies" - Cilfone, Antonio; Davoli, Luca; Belli, Laura; Ferrari, Gianluigi. 2019
5. “Mesh network topology (mesh network)”, <https://www.techtarget.com/iotagenda/definition/mesh-network-topology-mesh-network>, [Dostęp: 2.11.2022]
6. Maxim, DS18B20 Programmable Resolution 1-Wire Digital Thermometer – Karta Katalogowa
7. Songle Relay SRD – karta katalogowa, https://botland.com.pl/index.php?controller=attachment&id_attachment=244, [Dostęp: 10.02.2023]
8. “Mikrokontrolery ESP32, czyli dobry sposób na IoT”, <https://elektronikab2b.pl/technika/50826-mikrokontrolery-esp32-czyli-dobry-sposob-na-iot>, [Dostęp: 10.11.2022]
9. “Most In-Demand Programming Languages in 2022” <https://bootcamp.berkeley.edu/blog/most-in-demand-programming-languages/>, [Dostęp: 10.11.2022]
10. “What Are the Best Languages for App Development?”, <https://www.coursera.org/articles/best-language-for-app-development>, [Dostęp: 10.11.2022]
11. “React.js”, <https://reactjs.org/>, Meta Platforms, Inc, [Dostęp: 3.11.2022]
12. “About node.js”, <https://nodejs.org/en/about/>, [Dostęp: 7.11.2022]