КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

До захисту допущено
Завідувач кафедри ІСТ

_____ Олександр КУЧАНСЬКИЙ
(підпис) (ім'я, ПРІЗВИЩЕ)

"___"_____ 2022р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Спеціальності 126 «Інформаційні системи та технології»
Освітньої програми «Програмні технології інтернет речей»
На тему: **«Реалізація проекту інтернет речей на базі інфраструктури Amazon»**
Виконав: студент 4 курсу, групи ІР-41
(шифр групи)

Євгеній МАТВІЄНКО_____ _____
(Ім'я, ПРІЗВИЩЕ) (підпис)

Керівник к.т.н., асистент Мирослава ГЛАДКА
(посада, науковий ступінь, вчене звання, Ім'я, ПРІЗВИЩЕ) (підпис)

Консультант нормо контроль к.т.н., доц, Ростислав ЛІСНЕВСЬКИЙ
(назва розділу) (посада, вчене звання, науковий ступінь, Ім'я, ПРІЗВИЩЕ) (підпис)

Рецензент директор ТОВ "Тест" Андрій ВОВК_____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, Ім'я, ПРІЗВИЩЕ) (підпис)

Засвідчую, що пояснювальна записка не має запозичень з праць інших авторів без відповідних посилань.
Здобувач освіти _____
(підпис)

Київ – 2022 року

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра Інформаційні системи та технології
Освітній рівень Бакалавр
Спеціальність 126 Інформаційні системи та технології
Освітня програма Програмні технології інтернет речей

ЗАТВЕРДЖУЮ
Завідувач кафедри,
д.т.н., доцент
Олександр КУЧАНСЬКИЙ

_____
«__» _____ 2022 року

ЗАВДАННЯ
НА ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

Здобувач освіти: Матвієнко ЄВГЕНІЙ

Група: IP-41

1. **Тема кваліфікаційна робота бакалавра**: «Реалізація проекту інтернет речей на базі інфраструктури Amazon».

Затверджена протоколом засідання кафедри ІСТ №05/21_22 від 03.12.2021 року

2. **Строк подання студентом готової роботи** - «24» червня 2022 р.

3. **Вихідні дані до роботи**: дослідження в області рішення інтернет проекту на базі сервісів Amazon. Програмно-апаратні рішення для керування пристроями за допомогою протоколу MQTT з серверів Amazon. Дані температури з датчика, керування реле з використанням систем IoT.

4. Зміст роботи: робота складається з трьох розділів: дослідження теоретичної бази для розумних будинків і попередній огляд проекту, реалізація проекту інтернет речей на базі інфраструктури Amazon для автоматизації термоконтролю, експеримент і аналіз рішення.

Також робота включає висновок і два додатки.

6. **Календарний план виконання роботи:**

Етапи виконання кваліфікаційної роботи бакалавра	**Термін виконання**	**Результат виконання**
1. Вибір тематики кваліфікаційної роботи бакалавра	01.12.2021	виконано
2. Наказ про затвердження тем кваліфікаційної роботи бакалавра та призначення керівників	03.12.2021	виконано
3. Розробка плану кваліфікаційної роботи бакалавра і його погодження з керівником	25.12.2021	виконано
4. Написання І розділу кваліфікаційної роботи	20.01.2022	виконано
5. Написання ІІ розділу кваліфікаційної роботи	19.02.2022	виконано
6. Написання ІІІ розділу кваліфікаційної роботи	05.03.2022	виконано
7. Підготовка висновків і пропозицій		виконано
8. Попередній захист кваліфікаційної роботи	05.04.2022	виконано
9. Перевірка на плагіат	20.04.2022	виконано
10. Нормоконтроль	15.06.2022	виконано
11. Рецензування кваліфікаційної роботи бакалавра і представлення роботи на кафедру в друкованому вигляді	17.06.22	виконано
12. Захист кваліфікаційної роботи бакалавра	24.06.2022	

Дата видачі завдання «03» грудня 2021р.

Керівник роботи: к.т.н. Мирослава ГЛАДКА _____ (підпис)

Завдання прийняв до виконання:

Здобувач освіти на освітньому рівні «бакалавр» 4-го курсу групи ІР-41

Євгеній МАТВІЄНКО _____

(Власне Ім'я, ПРІЗВИЩЕ) (підпис)

АНОТАЦІЯ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра Інформаційних систем та технологій

Освітня програма «Програмні технології інтернет речей»

Кваліфікаційна робота бакалавра МАТВІЄНКА Євгенія

Тема роботи: «Реалізація проекту інтернет речей на базі інфраструктури Amazon».

Мета кваліфікаційної роботи бакалавра – розробка проекту квартири з автоматизацією системи опалення і реалізація програмного забезпечення для мікроконтролера, що буде підключатися до серверів Amazon, відправляти на нього дані датчиків та керуватись реле з серверів.

Об’єкт дослідження – мікроконтролер ESP-32.

Предмет дослідження – хмарні рішення Amazon, робота з протоколом MQTT, робота EPS32.

Кваліфікаційна робота бакалавра складається зі змісту, вступу, основної частини, яка включає три розділи, висновків та списку використаних джерел, два додатки. Всього 67 сторінок.

КЛЮЧОВІ СЛОВА: IoT, Wi-Fi, повторювач, мікроконтролер, ESP, Amazon, ESP-32.

ABSTRACT

TARAS SHEVCHENKO KYIV NATIONAL UNIVERSITY

Faculty of Information Technology

Department of Information Systems and Technologies

Educational program "Software technologies of the Internet of Things"

Qualifying work of bachelor MATVIENKO Eugene

Thesis topic: "Implementation of the Internet of Things project based on Amazon infrastructure".

The purpose of the bachelor's degree is to develop software for a microcontroller that will connect to Amazon's servers, send sensor data to it, and control relays from the servers.

The object of research is the ESP-32 microcontroller.

The subject of the research is Amazon's cloud solutions, work with the MQTT protocol, work of EPS32.

The bachelor's thesis consists of the content, introduction, main part, which includes three sections, conclusions and a list of sources used. Total 67 pages.

KEY WORDS: IoT, Wi-Fi, repeater, microcontroller, ESP, Amazon, ESP-32.

ЗМІСТ

Вступ

В даній роботі розглянуто реалізацію проекту інтернет речей для автоматизації контролю за температурою у квартирі на базі сервісів Amazon. Під час роботи буде наведено детально рішення використані для роботи та технології для реалізації. У виконанні було задіяно весь спектр необхідних компонентів для інтернет речей, зокрема, протокол для взаємодії розумних пристроїв MQTT, мікроконтролер ESP32, сервіси Amazon IoT Core, Amazon MQTT tester, Amazon Lambda Function. Для програмування було спеціально використано дві мови програмування, щоб продемонструвати доступність реалізації проектів інтернет речей не на конкретних стеках технологій і не конкретних діях для реалізації. А різноманітне використання тих чи інших технологій в рамках одного проекту і для різних задач.

Ключовою концепцією, що обрано для проєктування є технології Amazon Web Services, для серверної взаємодії з своїми проектами інтернет речей і не тільки для них. Amazon Web Services це великий спектр хмарних сервісів для різноманітних індустрій. Компанія Amazon довела часом ефективність своїх рішень, тому що клієнти не дискутують, яке рішення на ринку є кращим, а вони голосують за продукти своїми грошима. А це багато, що значить у сучасному світі капіталізму. У роботі аргументовано вибір саме Amazon Web Services перед своїм найближчим конкурентом Microsoft Azure. Досліджено правильне використання політик безпеки та робота по підключенню мікроконтролеру з Amazon Web Services.

Ціль роботи показати спосіб вирішення звичайної задачі автоматизації за допомогою інтернет речей та хмарних технологій конкурентного на ринку Amazon Web Services. Подібне рішення можна адаптувати до інших задач автоматизації дому та навіть невеликих задач підприємств, теплиць та розумних офісів.

Використання технології дає реалізувати у проекті керування реле на основі отриманих даних температури. Як перевага цього рішення є швидке і

просте керування значеннями заданих температур, що будуть тригерами для реле. Це відбувається без перепрограмування мікроконтролеру.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ТЕОРЕТИЧНОЇ БАЗИ ДЛЯ РОЗУМНИХ БУДИНКІВ, АНАЛІЗ РІШЕНЬ ДЛЯ РЕАЛІЗАЦІЇ ТА ОГЛЯД ПРОЕКТУ

1.1 Існуючі завдання мікроконтролерів у розумному будинку.

Мікроконтролери використовуються будь якому пристрої, де є складна логіка роботи. Люди з часом все більше вигадують рішень по підвищенню комфорту життя, автоматизації рутини, зниження шкідливості для здоров`я, щоб підвищити довжину життя. На допомогу приходять маленькі комп`ютери, які називаються мікроконтролерами. За їх допомогою ми змогли автоматизувати хатні турботи (пральні машини, посудомийки, розумні плити, мультиварки та інше), подовжувати життя (фітнес-трекери, круіз-контроль та інше.) і підвищувати продуктивність (автоматизовані лінії на заводах).

Задачі мікроконтролерів для дому можна поділити на такі групи:

- безпека;
- комфорт;
- розваги;
- підвищення стану здоров`я;
- контроль температури;
- освітлення;
- хатній клопіт.

У цій роботі буде розглянуто рішення для автоматизації контролю за температурою у квартирі. Будуть наведені конкретні рішення та реалізація для здійснення такої задачі.

1.2 Попередній огляд системи.

Такі системи розроблюються для автоматизації буденних задач людей і підвищення їх комфорту. Комерційні системи поставляються зазвичай у повному комплекті: хаб (мікроконтролер), датчики, виконавчі пристрої. У додаток до цього добрий досвід користувача у налаштуванні і все базується на

хмарний рішеннях компанії постачальника пристроїв. Такі рішення розроблюються для конкретних завдань: опалення, освітлення, безпека будинку та інші.

У цій роботі буде наглядно реалізовано проект автоматизації опалення квартири. Самостійно буде запрограмовано мікроконтролер, налаштовано хмарне рішення, спроектовано квартиру у трьохвимірній моделі, розроблено базу даних і веб-додаток для моніторингу системи.

Особливу увагу надано вибору хмарного рішення серед двох найпопулярніших на ринку Amazon IoT і Microsoft Azure. Серед критеріїв були вартість, простота використання окремих сервісів.

1.2.1 AWS IoT.

AWS IoT надає хмарні послуги, які з'єднують пристрої IoT з іншими хмарними пристроями та службами компанії. Платформа надає програмне забезпечення для пристроїв, яке може допомогти вам інтегрувати пристрої IoT у рішення на основі Веб-сервісу Амазон. Якщо ваші пристрої можуть підключатися до нього, AWS IoT може підключати їх до хмарних служб, що надаються компанією відповідно.

Завдяки хмарній платформі, оптимізованій для Інтернету речей можна:

- встановити з'єднання між програмним забезпеченням, апаратним забезпеченням і хмарою Інтернету речей;
- налаштувати сховище, аналітичні дані, обробку потоків варіантів розвитку подій і рішення для даних Інтернету речей;
- підсумувати статистику та подавати її далі, щоб ініціювати цільові дії та варіантів покращення роботи системи;
- увімкнути обробку, що допоможе вам збирати дані про користування і роботу системи, що в свою чергу покращить якість експлуатації;
- відстежувати та керувати своїми датчиками, приладами та іншими пристроями IoT.

Додатковою перевагою буде те, що така багатофункціональна платформа може допомогти вам подолати будь-які проблеми, з якими ви можете зіткнутися, коли ви тільки знайомитесь з Інтернетом речей. Платформа дає вам прямий контроль над обладнанням та додатками, посилює безпеку та допомагає вам розібратися у системі за допомогою готових функцій.

Дану систему я обрав, тому що вона є найлегшою у використанні, та надає гарантії якісної роботи. Технологія Інтернету речей AWS є дуже масштабною. AWS IoT вважається найбільш повною платформою для малих і великих рішень IoT — навіть для виробництва і домівок. Вона використовується для керування пристроями, розгортання, зберігання даних, аналітики, різних операцій та інших завдань, отримуючи вигоду від інтеграції з основними хмарними сервісами Amazon.

У пристрої є датчик температури та реле, що у відповідь на встановлені параметри отриманих даних вмикає і вимикає прилади опалення для регуляції температури. Ми можемо використати це, управляючи автоматичним опаленням, за допомогою налаштування параметрів так, що при температурі, коли вам холодно, система подає сигнал і за допомогою цього обігрівач вмикається та нагріває повітря до визначеної також у параметрах температури. Доки температура не потрапить у правильний за нормами діапазон температур, то система буде тримати відповідний пристрій в увімкненому стані.

Було розроблено архітектуру рішення (рис. 1.1) з взаємодією хмарних сервісів Amazon. Все починається з передачі даних про температуру з сенсору, що під'єднаний до мікроконтролеру. Ця інформація передається протоколом MQTT у топік “esp32/pub”. Це повідомлення проходить низку вузлів у мережі інтернет. Сервіс Amazon IoT Core, в якому попередньо зареєстрований пристрій, отримані секретні ключі для з`єднання пристрою і налаштовані політики взаємодії, отримує це повідомлення і направляє його до дуже корисного сервісу Amazon Lambda.

Саме це рішення буде оброблювати вхідні дані та передавати потрібну конфігурацію увімкнення реле, тобто керувати температурою квартири. Цей сервіс дає можливість оброблювати дані, тільки коли вони потрапляють до вузла через тригер спрацювання, що дає велику економію для компанії Amazon. Інакше кажучи, ми отримуємо серверну ноду безкоштовно і дуже великою продуктивністю. Обробка вхідних даних відбувається за лічені мілісекунди.

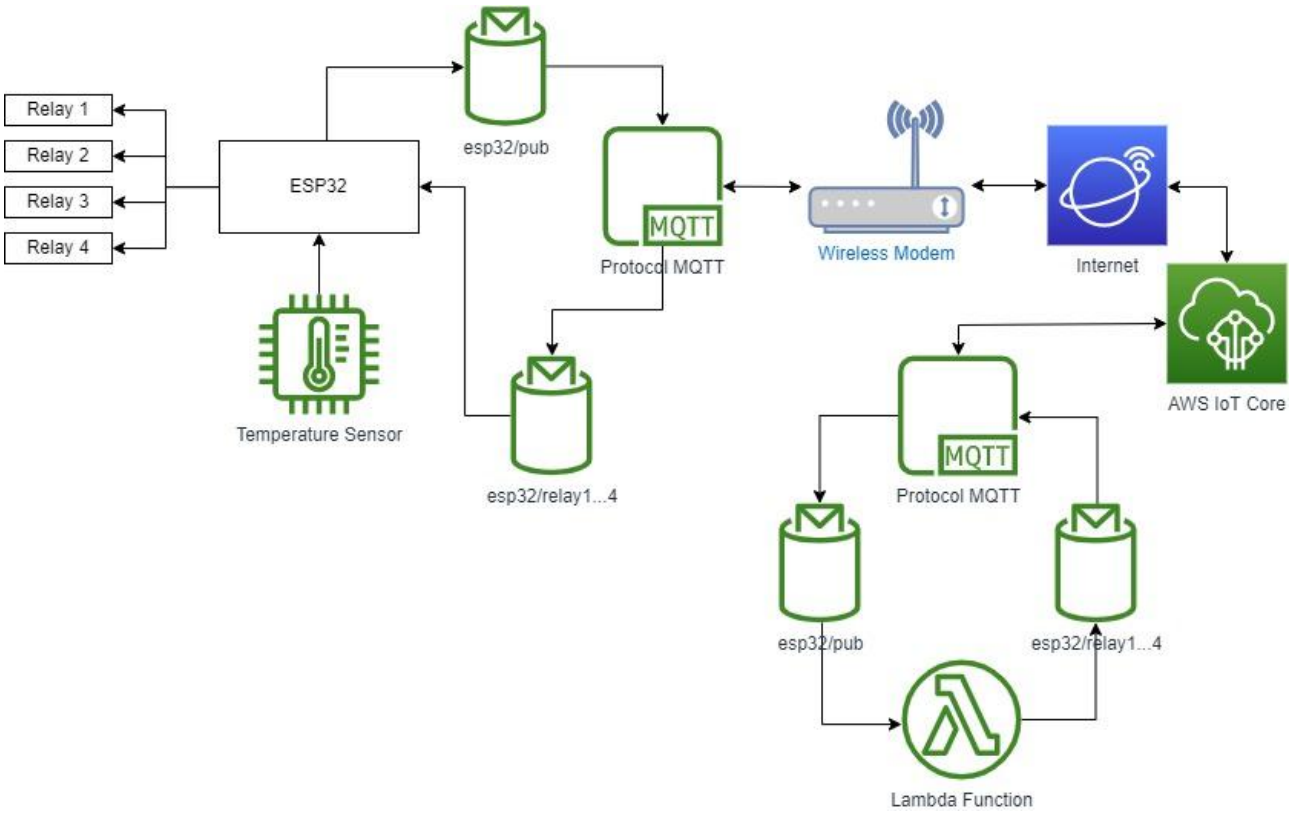


Рисунок 1.1 - Архітектура рішення автоматизації опалення

Сильною стороною цієї архітектури є масштабованість. Можна під`єднати будь-яку кількість датчиків та мікроконтролерів до даного хмарного сервісу і однаково швидко оброблювати отримані температури. У тому числі ніхто не обмежує у кількості отримуванихємих даних і часу роботи, що є неоціненною перевагою. Дана послуга є безкоштовною, що ідеально підходить для даного невеликого проекту.

Далі ми розглянемо алгоритм роботи цього пристрою інтернет речей (рис. 1.2), з інтеграцією сервісів Amazon. В даній схемі є логічне розбиття на дві

області. Перша - це мікроконтролер ESP32 і його логіка роботи. Друга - це сервіс Amazon і його логіка обробки вхідних даних.

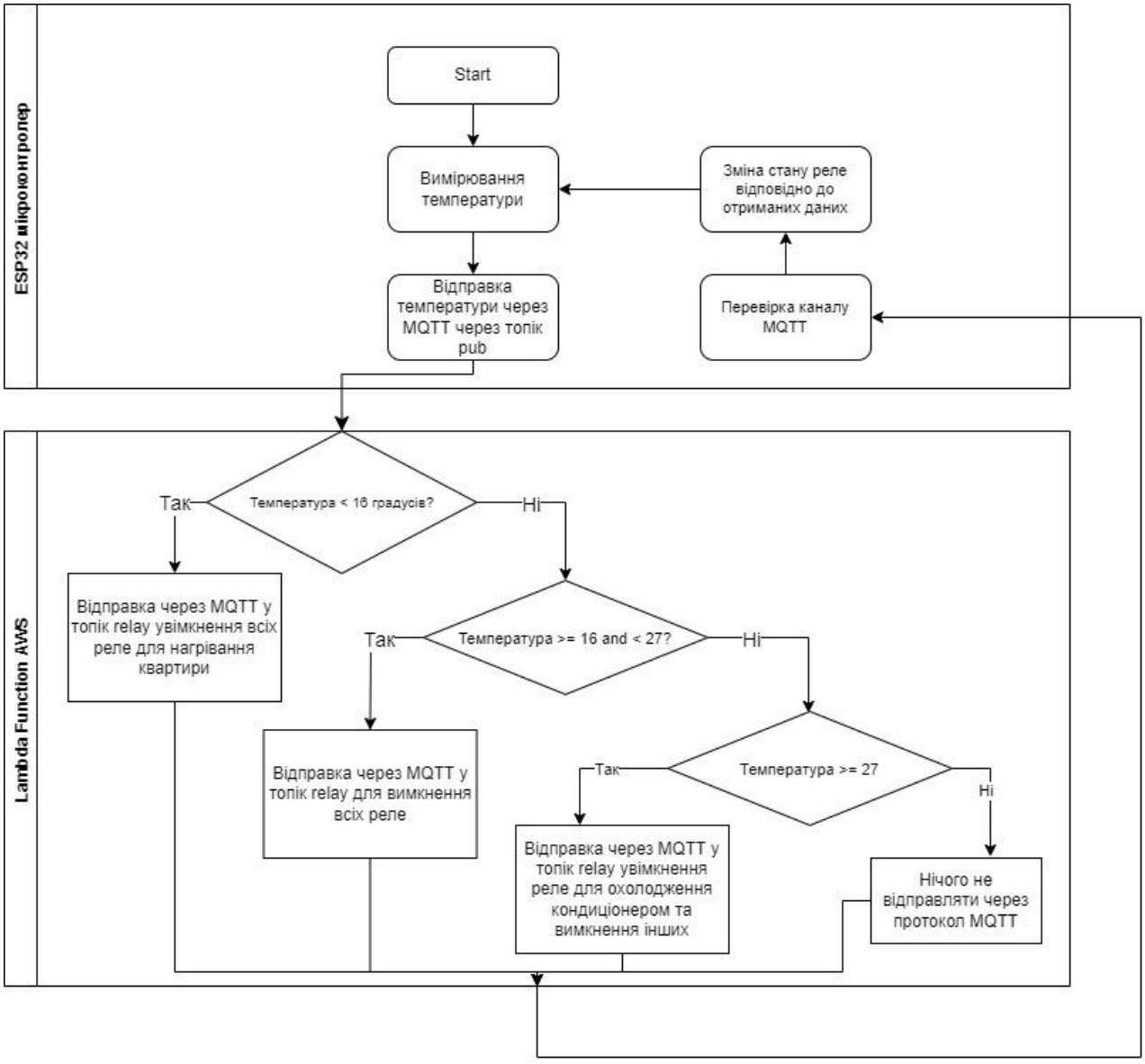


Рисунок 1.2 - Алгоритм роботи системи автоматизації опалення

Задачею ESP32 є вимірювання температури, відправка цих даних протоколом MQTT через топік “/pub”. Дані проходять ланцюг інтернет вузлів і отримуються вже на серверах Amazon.

В Amazon отримуються дані і передаються у внутрішній сервіс Amazon Lambda. Сервер розпаковує повідомлення протоколу MQTT. Та відправляє у топік відповідного реле потрібну конфігурацію увімкнення за граничними температурами (таб. 1.1).

Таблиця 1.1 Логіка роботи реле

№ реле	**Температурний діапазон**		
1	2	3	4
	temp < 16	temp >= 16 and temp < 27	27 ≥ temp
1	1	0	0
2	1	0	0
3	1	0	0
4	0	0	1

Після відправки сервером Amazon повідомлення з конфігурацією реле - відбувається тригер на ESP32 і мікроконтролер реалізує отриману конфігурацію увімкнення приладів.

Також це може спрацювати з жалюзі, наприклад, якщо світить сонце і нагріває поверхні у будинку, тим самим через це, нагріваючи повітря, система отримує дані про завищену температуру і жалюзі закриваються (також це може працювати просто зі світлом, просто потрібно налаштувати систему так, що за допомогою датчика, який буде контролювати рівень освітлення кімнати і передавати дані в систему, жалюзі буде закриватися).

Розглянемо кожне реле відповідно до пристрою, що під`єднаний:

- реле №1 - кондиціонер, для підігрівання приміщень;
- реле №2 - котел;
- реле №3 - тепла підлога;
- реле №4 - кондиціонер, для охолодження приміщень.

Далі ми розглянемо логіку спрацювання реле (таблиця 1.1) у відповідь на отримані дані температури з датчику. Дана конфігурація температури була обрана відповідно до санітарних норм температурного режиму приміщень.

За даною таблицею ми розуміємо, що:

- перший стан при граничній температурі меншій, ніж шістнадцять градусів цельсію будуть увімкнені усі пристрої підігрівання приміщення. А кондиціонер у режимі охолодження буде відімкнутий;
- наступний стан у діапазоні між шістнадцятьма включно і менше двадцяти сіми не включно будуть відімкнені взагалі усі пристрої, як нагрівальні, так і охолоджувальні. Тому що температура вважається допустимою і комфортною у даній квартирі;
- останній стан являє собою стан перенагрітої квартири. Він вмикається, якщо температура більша ніж двадцять сім градусів цельсію включно. При ньому вимикаються усі нагрівальні пристрої і вмикається кондиціонер на режимі охолодження.

1.2.2 Рішення Danfoss.

На ринку мало подібних рішень даного проекту. Один з нечисленних виявився Danfoss (рис. 1.3). Але це більш функціональне рішення. В ньому реалізовано бездротову взаємодію між віддаленими пристроями, а не дротовим способом, як у цій роботі.

Радіаторні терморегулятори Danfoss Ally™ сертифіковані за стандартом Zigbee 3.0. Це означає, що вони спілкуються однією мовою за допомогою бездротового зв'язку, як і безліч інших побутових смарт-пристроїв по всьому світу, дозволяючи вам підключати ці терморегулятори до існуючої системи розумного будинку на протоколі Zigbee 3.0.

Для хмарного сервісу компанія використовує не Amazon Web Services, а власну розробку. Додатково вони реалізували свій мобільний додаток та інтеграцію з розумними асистентами.

Його додаткові функції і більш технологічна реалізація обумовлена його великою ціною. Мінімальний набір обійдеться покупцю у, приблизно, чотирнадцять тисяч гривень.

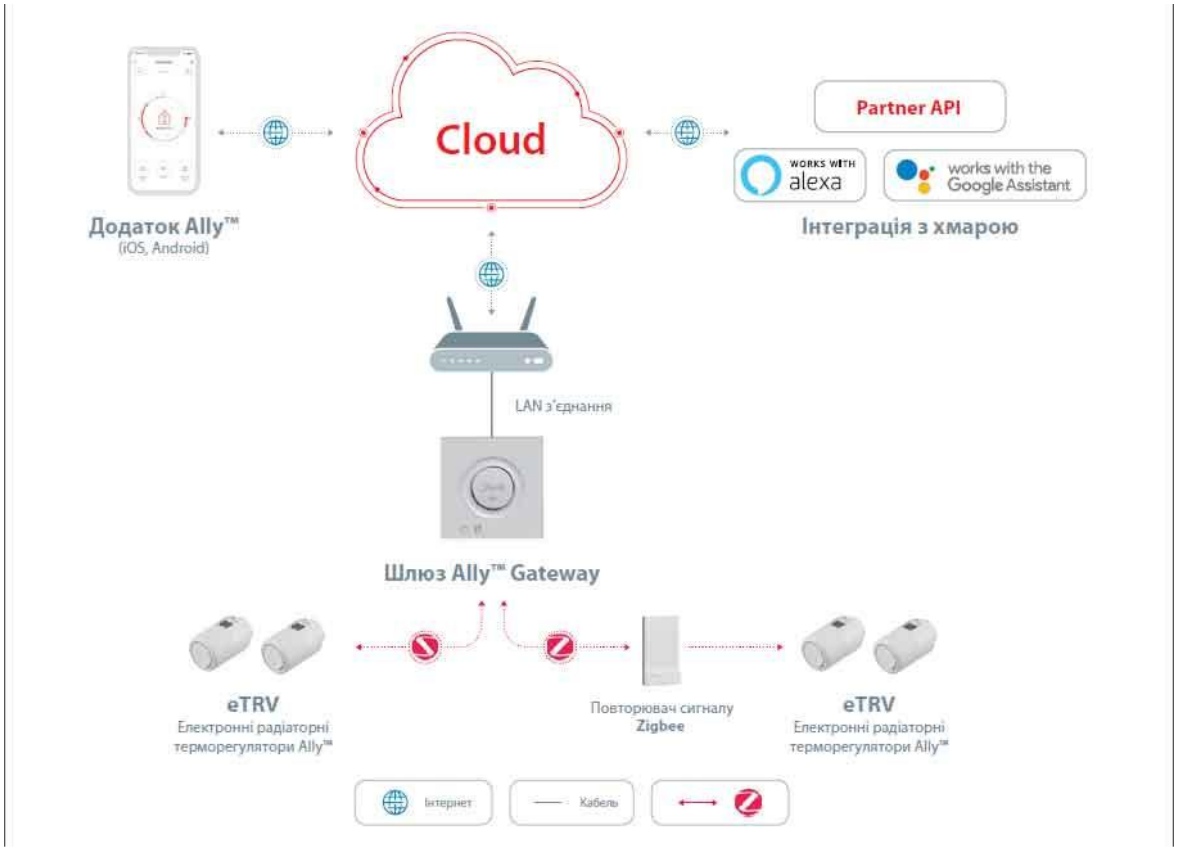


Рисунок 1.3 - Архітектура рішення від Danfoss

1.3 Вибір мікроконтролеру.

Було дослідження різні рішення для реалізації проекту інтернет речей (таб. 1.2). Як головною характеристикою для вибору, було присутність WIFI модуля.

В інтернеті було знайдено безліч матеріалів по роботі з даним мікроконтролером. Це дало змогу швидко перейти до розробки, а не поглиблюватись у деталі.

Для порівняння було обрано мікроконтролери: Intel 8051, ICD2 USB, Arduino Uno R3, STM32, ESP32. Далі буде коротке описання кожного з них.

Intel 8051 є мікроконтролером далеких 1980-х років. Він має дуже низьку ціну у 70 гривень, але зовсім низьку потужність і відсутність готових плат, для використання.

ICD2 USB є мікроконтролером на базі архітектури PIC. Вже можна знайти доступну документацію. Продуктивність трохи більша за Intel 8051, але ціна дуже висока у розмірі 4386 гривень.

Arduino Uno R3 є дуже популярною серією плат розробників для мікроконтролерів. Безліч навчальних матеріалів, доступна ціна, але невидатна продуктивність у сучасних реаліях.

STM32 є цікавим рішенням на ринку мікроконтролерів. Його ядра на відомій архітектурі ARM. За ціною, приблизно, такий самий, як і Arduino Uno, за продуктивністю має перевагу, але недоліком є деякі нюанси при розробці програмного забезпечення.

ESP32 є найпотужнішим серед наведених мікроконтролерів. Ціна в рамках допустимого, враховуючи таку продуктивність. А найголовніше, що у мікроконтролері з заводу реалізований бездротовий інтерфейс. Що відмінно лягає у концепцію цього проекту інтернет речей.

Таблиця 1.2 Таблиця порівняння мікроконтролерів

Параметр оцінювання	**Мікроконтролери для оцінювання**				
1	2	3	4	5	6
	Intel 8051	**ICD2 USB**	**Arduino Uno R3**	**STM32**	**ESP32**
Простота і інф.	1	2	10	8	10
Готові плати	нема	Є 1	БАГАТО	Є вибір	Є вибір
Продуктивність	1	2	5	7	10
WIFI	Нема	Нема	Є модулі	Є модулі	Вмонтовано
Ціна (грн)	70	4386	286	280	360

За дослідженням ринку було обрано мікроконтролер саме ESP32, він має таку саму простоту програмування як і рішення готови плат від Arduino, лідує

по продуктивності, а найголовніше на відразу має підримку WIFI та інших бездротових технологій, що є великою перевагою. І все цей мікроконтролер на готовій платі коштує невеликі кошти.

1.4 Вибір хмарного рішення.

Серед найкращих платформ IoT пропозиції Amazon і Microsoft є найпопулярнішими. Частки ринку Amazon Web Services та Microsoft Azure складаю 31% і 22% відповідно, як постачальників інфраструктури.

Також, з точки зору інтернету речей, AWS займає лідируючі позиції. Користувачі знаходять безліч переваг платформи, а саме:

- доступ до сервісів на AWS, агрегація та обчислення даних;
- використання різних апаратних рішень партнерів Amazon;
- великий вибір передналаштованих середовищ розробки SDK;
- автоматизація процесу аналітики, підключення та розробки програмних рішень IoT і його сумісне;
- у Amazon Web Services є більше безкоштовних можливостей, ніж у Microsoft Azure;
- необмежене масштабування всіх хмарних рішень, що пропонує Amazon;
- додаткові можливості AI сервісів.

Azure в свою чергу надає:

- обмежений обсяг послуг IoT;
- апаратна сумістність менша за Amazon;
- менш розвинена інтеграція AI;
- мало безкоштовних рішень.

За даними параметрами AWS є привабливішим рішенням, ніж Microsoft Azure. Запропонований вибір сервісів від постачальників слід урахувати для більш об`єктивної оцінки. Можемо оцінити, що AWS надає цілих більше сервісів інтернету речей, в той час як Azure надає вдвічі менше послуг. Розглянемо сервіси та їх позитивні сторони.

Сервіси AWS IoT:

• freeRTOS - операційна система розумних пристроїв, яка постачається з системою для об'єднання мікроконтролерів і граничних пристроїв;

• AWS IoT Greengrass - ще один програмний продукт для обробки, яке дозволяє швидко реалізовувати та розгортати IoT рішення;

• AWS IoT Core — хаб керування пристроями IoT, в ньому можна налаштувати керування вхідними і вихідними даними;

• управління пристроями AWS IoT — сервіс, що керує пристроями незалежно від апаратної реалізацію. Є можливість оновлень на керувань станом;

• AWS IoT Events — сервіс керування пристроями за допомогою тригер-подій;

• AWS IoT SiteWise — служба моніторингу апаратного устаткування на серверах Amazon;

• AWS IoT Things Graph — сервіс налаштування процесів між розумними пристроями;

• AWS IoT Device Defender — хмарний сервіс безпеки для пристроїв інтернет речей;

• AWS IoT 1-Click — особливий сервіс реалізації дій над пристроєм в 1 клік із підключенням лямбда функцій;

• AWS IoT Analytics — хаб для збору інформації, візуалізації і аналітиці.

Розглянуто сервіси, що пропонує Amazon. Далі подивимось, що є у Microsoft Azure.

Головні сервіси Azure:

• Azure IoT Hub — хаб для розумних пристроїв, є функції керування;

• Azure IoT Central — сервіс для інтеграції рішень у пристроях;

• Azure IoT Edge — сервіс реалізації процесів керування і аналітики розумних пристроїв;

• Insights — аналітика та моніторинг даних з пристрою;

• Azure Digital Twins — сервіс реалізації цифрових дублів пристроїв, що взаємодіють фізично.

Досліджено Azure, а тепер розглянемо головні складові хмарних рішень, а саме: безпеку, масштабованість, аналітику, керування пристроєм, розвиток та інтеграцію та крайні можливості.

Почнемо аналіз з безпеки. Ця складова допоможе нам зберегти майно і наш гаманець від противоправних дій інших людей. Azure і AWS на 100% надійні, якби це було не так, їх сукупна частка ринку була б нижче нуля. Обидві системи дозволяють налаштувати двосторонню аутентифікацію, дозволи на різних рівнях, включаючи найменші деталі, унікальні посвідчення та безризикові процеси обміну даними. З'єднання TLS і шифрування об'єднані в усі шари хмари, тому обидві системи за цим пунктом є рівними.

Масштабованість:

Amazon має перевагу у кількості пропозицій для хмарної реалізації інтернет речей. Коли вам потрібні ресурси для нових пристроїв за допомогою вбудованого механізму правил інтеграції з лямбда та іншими службами є дуже корисним компонентом. Гнучка цінова політика дає змогу уникнути обмеження по вирахувальним потужностям і зберіганням даних.

Аналітика:

Незалежно від поставленої мети: прогнозування, виведення обладнання зі звичного режиму або визначення закономірностей, AWS стане надійним інструментом, що за допомогою пакеті SDK реалізує потоки даних від розумних пристроїв до сервісу аналітики. Доповненням SiteWise є особливо цінним сервісом, що задіяний у промисловості. Azure не має розумних рішень для промислових проектів.

Керування пристроєм:

В арсеналі AWS наявні всі необхідні можливості для будь-яких пристроїв. Рішення з інтеграції AWS IoT Core, Device Management та Device Defender дає змогу реалізовувати реальні промислові проекти без задіяння особливого

капіталу та навичок. На відміну від сервісу Azure Hub вищезгадані служби краще піклуються про пристрої клієнтів, використовуючи більший спектр можливостей.

Розвиток та інтеграція:

Дані платформи тісно конкурують по визначеним інструментам для розробки, налаштування, моніторингу і розгортання рішень інтернет речей. Але для спеціальних програм пристрої інтернет речей більш бажані для роботи у цій індустрії є інструменти SDK. Це дає змогу реалізовувати проекти інтернет речей з багатьма функціями, якщо є відповідна команда розробників.

Крайні можливості:

Нам допоможуть розгорнути рішення інтернет речей підготовлені бібліотеки FreeRTOS і Greengrass, що окремо виділяється перевагою AWS. У Azure реалізований механізм контейнеризації Microsoft, що обмежує гнучкість нашого рішення інтернету речей. Для кращого заощадження на сховищах даних і непотрібних хмарних налаштувань, а також для розширення можливостей у різних випадках використання периферійних обчислень AWS має ширшу інфраструктуру.

Отже, AWS все таки кращий вибір, який надасть вам багато можливостей та зручність користування за даною аналітикою.

1.5 Вибір компонентів для проектування системи регулювання опалення у квартирі.

На базі рішення буде спроектовано систему, що буде автоматично керувати опаленням у квартирі. За допомогою цього у користувачів буде підвищення комфорту та економія енергії.

У системі використано:

- термоелектричний привід;
- мікроконтролер;
- кондиціонер.

1.5.1 Вибір термоелектричного приводу.

Термоелектричні міні-приводи призначені для двопозиційного керування різними регулюючими клапанами в системах опалення та охолодження місцевих вентиляційних установок.

На ринку було розглянуто привід Danfoss, RA, NC, 24В (рис. 1.4). с За відгуками виріб є надійним і мінімальна кількість браку після покупки. У комплекті є доступна для розуміння інструкція. Перевагою є велика кількість відеоматеріалів по впровадженню в експлуатацію цих приводів.

Привід оснащений візуальним індикатором ходу, який показує, в якому положенні знаходиться клапан – закритий або відкритий. Для керування треба просто подати напругу на дроти цього виробу. Цією задачею буде займатися реле під контролем ESP32. А мікроконтролер вже буде отримувати конфігурацію для вмикання приладів вже з серверів хмарного сервісу Amazon Lambda Function.

Рисунок 1.4 - Термоелектричний привід Danfoss, RA, NC, 24В

Наступним для порівняння став TECH STT-230/2 M28x1,5 (рис. 1.5). З отриманої інформації з відкритих джерел стало відомим, що у даного пристрою мало документації, мало відгуків і ціна більша ніж у Danfoss. Рішення вважаю ненадійним і не привабливим для використання.

Рисунок 1.5 - Термоелектричний привід TECH STT-230/2 M28x1,5

Серед порівняних приводів було обрано для даного проекту саме Danfoss, RA, NC, 24В. Не дивно, що саме ця компанія є лідером на цьому ринку.

1.5.2 Вибір кондиціонера.

Haier AS25TADHRA-CL (рис. 1.6) – кондиціонер серії Tibio Inverter, працює на обігрів до -20 °С. Положення жалюзі встановлюється автоматично залежно від вибраного режиму. При роботі на охолодження струмінь спрямований вгору, щоб унеможливити переохолодження та застудні захворювання. При обігріві повітря спрямоване вниз, тому швидше досягається комфортна температура у приміщенні. Усього існує 11 варіантів розташування жалюзі.

Рисунок 1.6 - Кондиціонер Haier

Спліт-система розрахована обслуговування приміщень до 36 квадратних метрів. Конструкція піддону зовнішнього блоку не дозволяє йому промерзати. Передбачені додаткові дренажні отвори швидкого відведення зайвої вологи. Це полегшує обслуговування, продовжує термін експлуатації спліт-системи.

Наступний для порівняння кондиціонер від Electrolux EACM-10AG (рис. 1.7). Цей кондиціонер поставляється з ексклюзивним дизайном і з функцією очищення повітря. Забезпечує рівномірну віддачу повітря у квартирі. Кондиціонер має такі режими: очищення повітря, охолодження, підігрів.

Рисунок 1.7 - Кондиціонер Electrolux EACM-10AG

Пульт вмонтований у блок кондиціонера. У реалізації є спеціальний потужний термостат, таймер і вибір режиму вентиляції.

Основним недоліком кондиціонера є його мара площа покриття приміщення, усього 15 м^2. Тому це рішення не підходить для реалізації проекту. Кондиціонер Haier був обраний з додатковою функцією підігріву, що дасть змогу підсилити опалення при дуже низьких температурах. Даний варіант має перевагу через свою велику площу покриття регулювання температури. Вона приближена до площі цілої квартири за планом проекту.

Висновок до розділу 1

У даному розділі було розглянуто теоретичну базу мікроконтролерів, їх використання у житті суспільства. Проведений огляд проекту автоматизації термоконтролю. Розглянута його архітектура, алгоритм роботи системи.

Знайдено аналогічне рішення на ринку для даного проекту. Проаналізовані і підібрані допустимі компоненти автоматизації опалення, що будуть виконувати всі поставленні завдання проекту.

Обрано Amazon Web Services кращим хмарним рішенням е порівнянні з Microsoft Azure. Основною перевагою стали безкоштовні сервіси і продумані рішення саме для інтернет речей.

РОЗДІЛ 2. РЕАЛІЗАЦІЯ ПРОЕКТУ ІНТЕРНЕТ РЕЧЕЙ НА БАЗІ ІНФРАСТРУКТУРИ AMAZON ДЛЯ АВТОМАТИЗАЦІЇ ТЕРМОКОНТРОЛЮ

2.1. Розробка квартири з автоматичним регулюванням температури.

2.1.1. Розробка плану квартири.

Для плану було розроблено стандартну по складу квартиру невеликого розміру (рис. 2.1). В ній всього одна спальна кімната, роздільний санвузол, вітальня і кухня. Вже на етапі моделювання буде розміщено обігрівальні елементи системи.

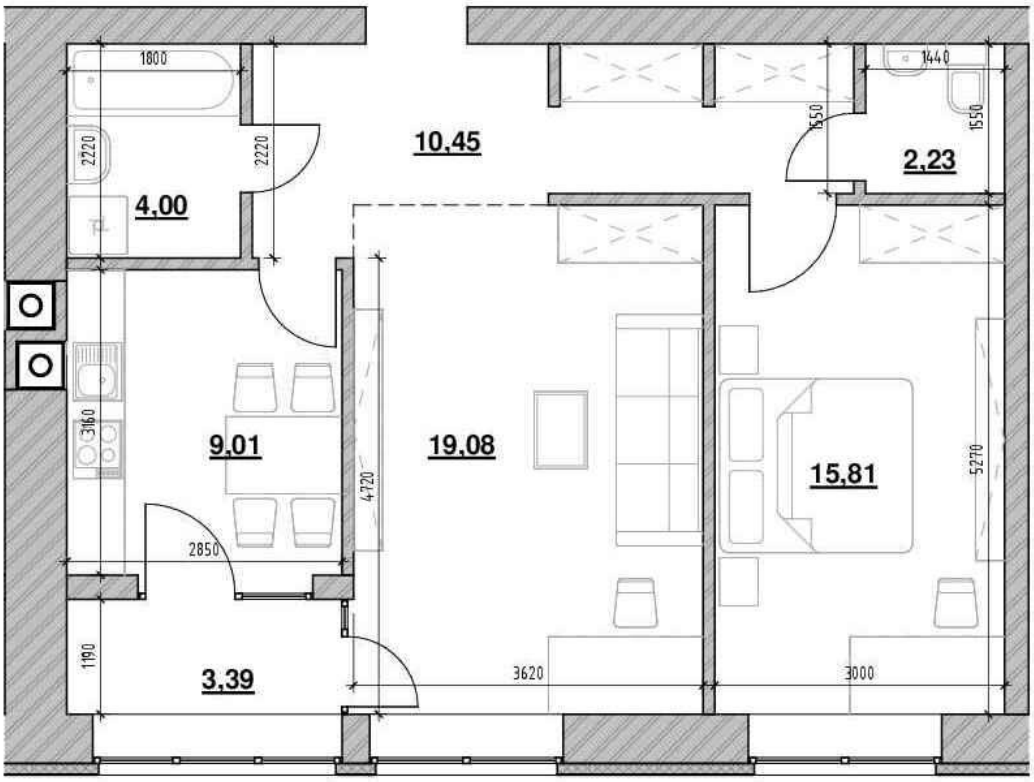


Рисунок 2.1 - План квартири

2.1.2 Побудова 3-х мірної моделі квартири з пристроями.

За допомогою програми Blender було спроектована 3-х мірна модель квартири (рис. 2.2).

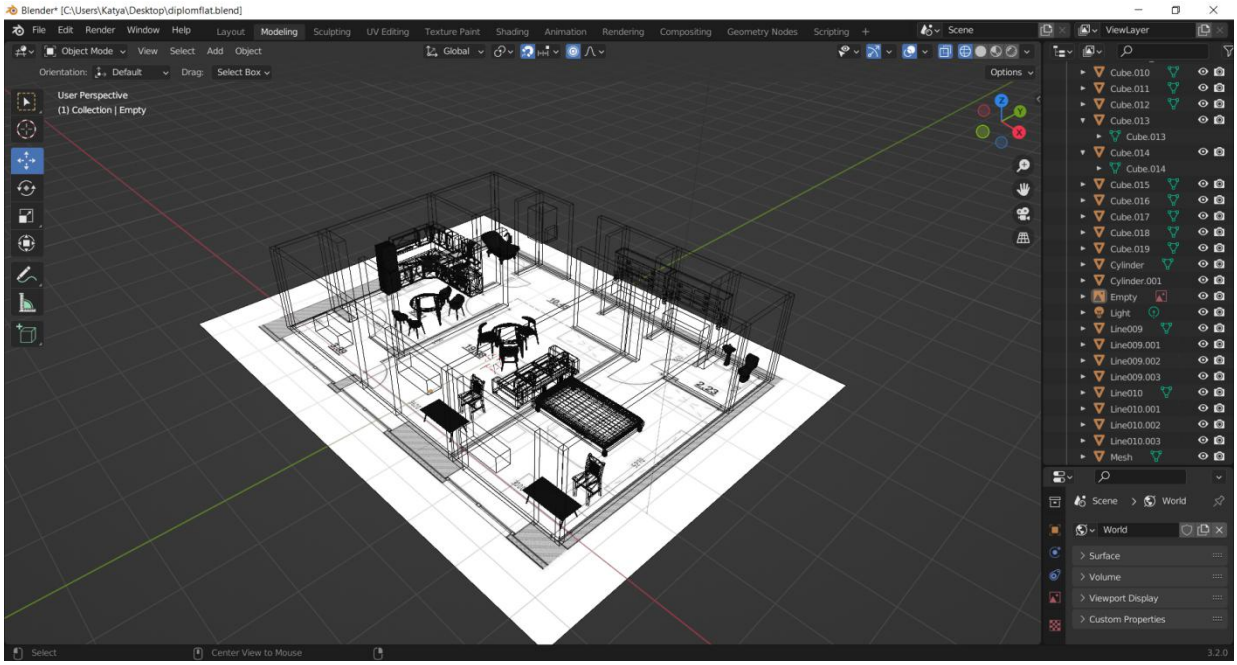


Рисунок 2.2 - 3-х мірна модель квартири побудована на основі плану

Унаслідок вийшла модель квартири і в ній базовими фігурами показано розташування елементів системи автоматизації термоконтролю квартири (рис. 2.3).

Означення елементів за кольором:

1. червоний - котел, центральний вузол опалення;
2. зелений - плата керування ESP32 та блок реле;
3. жовтий - кондиціонер;
4. синій - батареї опалення з встановленими термоелектричними приводами Danfoss.

Згідно з побудованою моделлю було розташовано п`ять батарей і по одному екземпляру кожного іншого приладу.

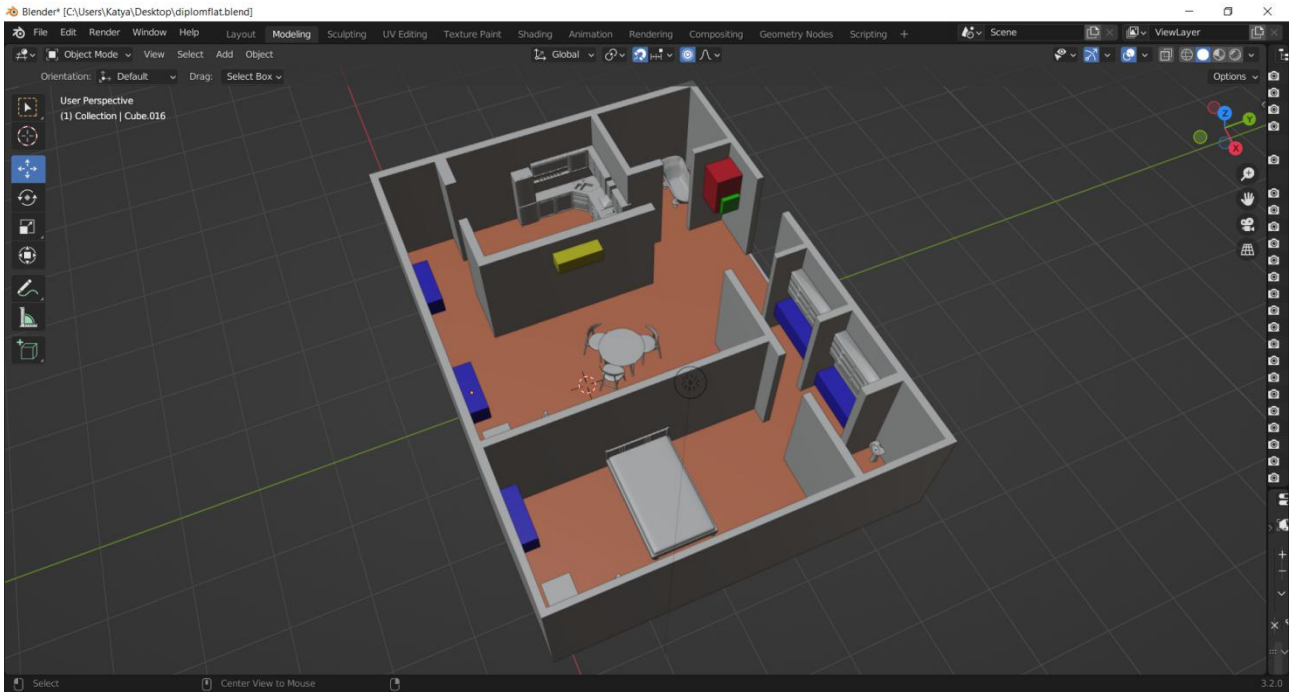


Рисунок 2.3 - Модель квартири з ідентифікацією компонентів системи

2.2 Створення проекту інтернет речей на базі Amazon.

2.2.1 Апаратна реалізація.

Як вже зазначалось, у проекті використаний мікроконтролер ESP32 (рис. 2.4). Він має з заводу інтегрований модуль WIFI, велику продуктивність і невелику ціну.

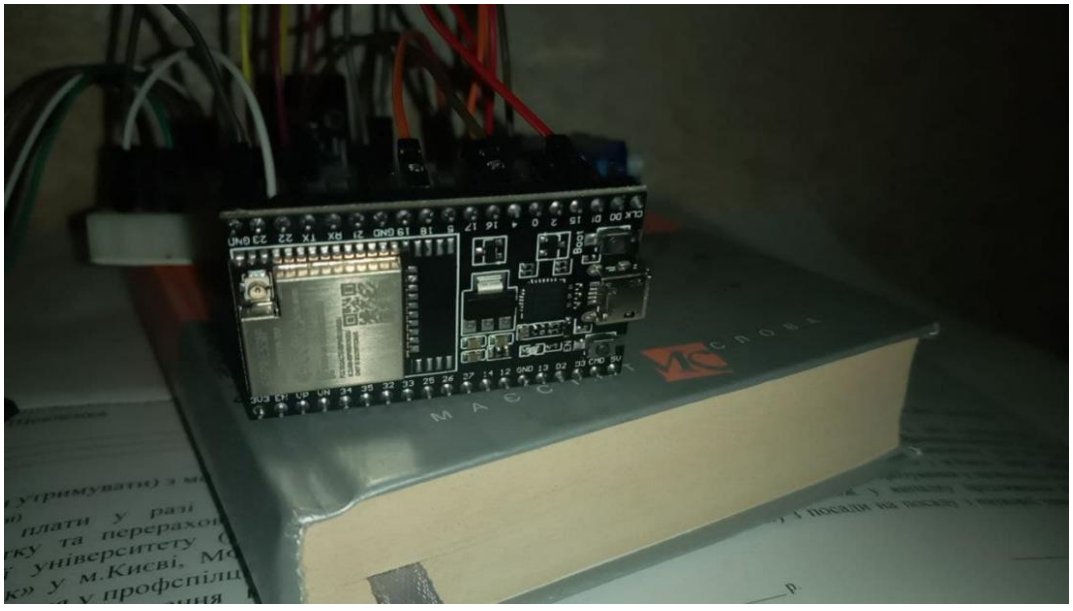


Рисунок 2.4 - Плата на базі ESP32

Модулі було під`єднано до таких пінів ESP32:

1. датчик температури - 16;
2. реле №1 - 15;
3. реле №2 - 2;
4. реле №3 - 4;
5. реле №4 - 22.

Для комфортнішого з`єднання було використано безпайочну монтажну плату. Це дало змогу зробити більш наглядні фото пристрою.

В якості реле використано SONGLE SRD-05VDC-SL-C (рис. 2.5). Тому що вона була під рукою.

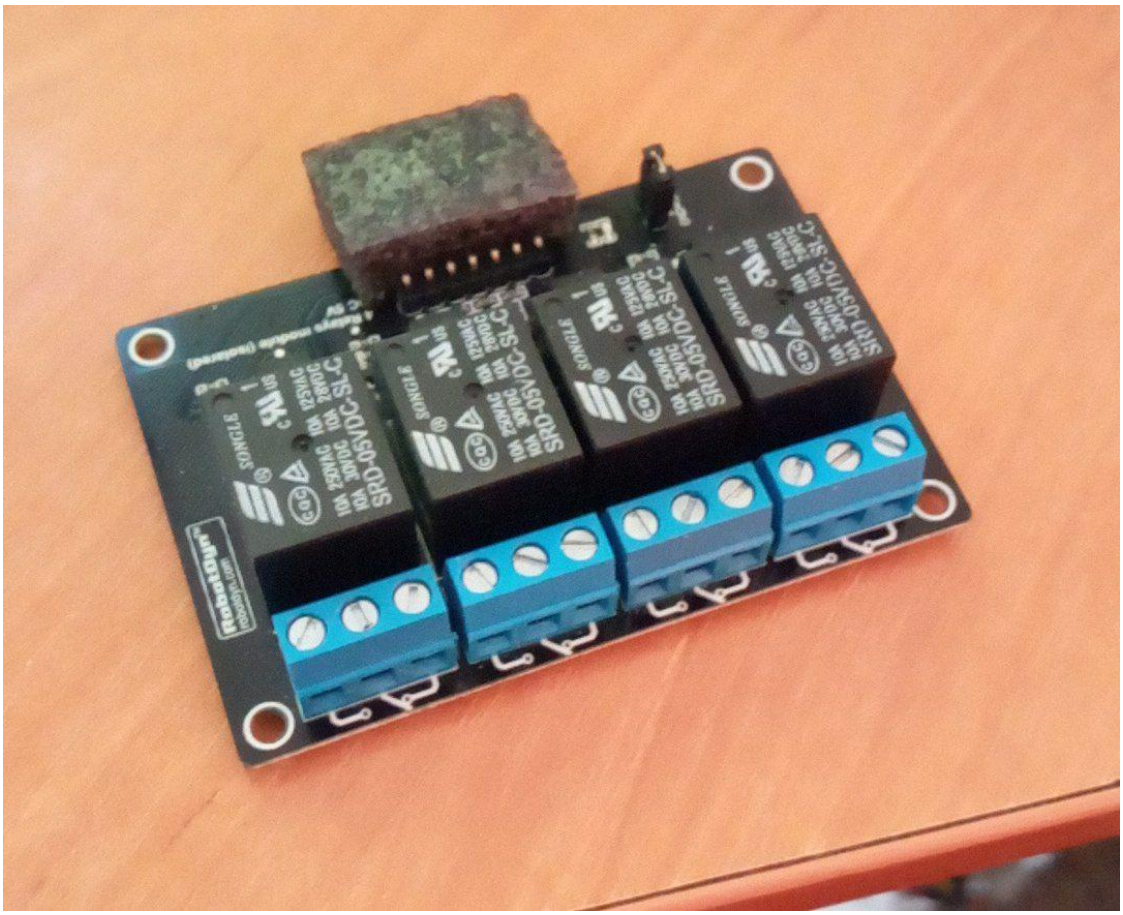


Рисунок 2.5 - Реле для проекту

2.2.2 Ініціалізація проекту у системі Amazon.

Спочатку було створено розумний пристрій у системі Amazon IoT Core (рис. 2.6).

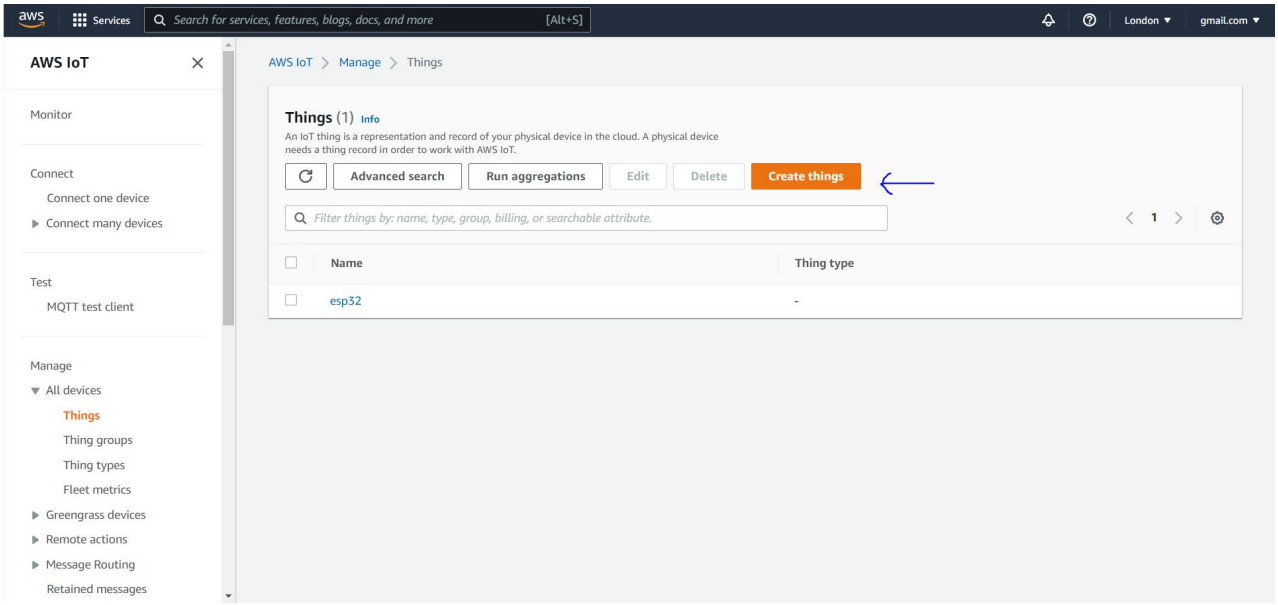


Рисунок 2.6 - Додавання пристрою в Amazon

Далі згенеровано сертифікат для безпечного доступу до хмари Amazon (рис. 2.7). Він дає змогу ідентифікувати з`єднання пристрою з хмарним сервісом Amazon серед інших пристроїв. Також це дає змогу забезпечити безпечне спілкування мікроконтролеру з сервером, великий ключ безпеки який буде у внутрішній пам`яті пристрою і ніякий зловмисник про нього не дізнається.

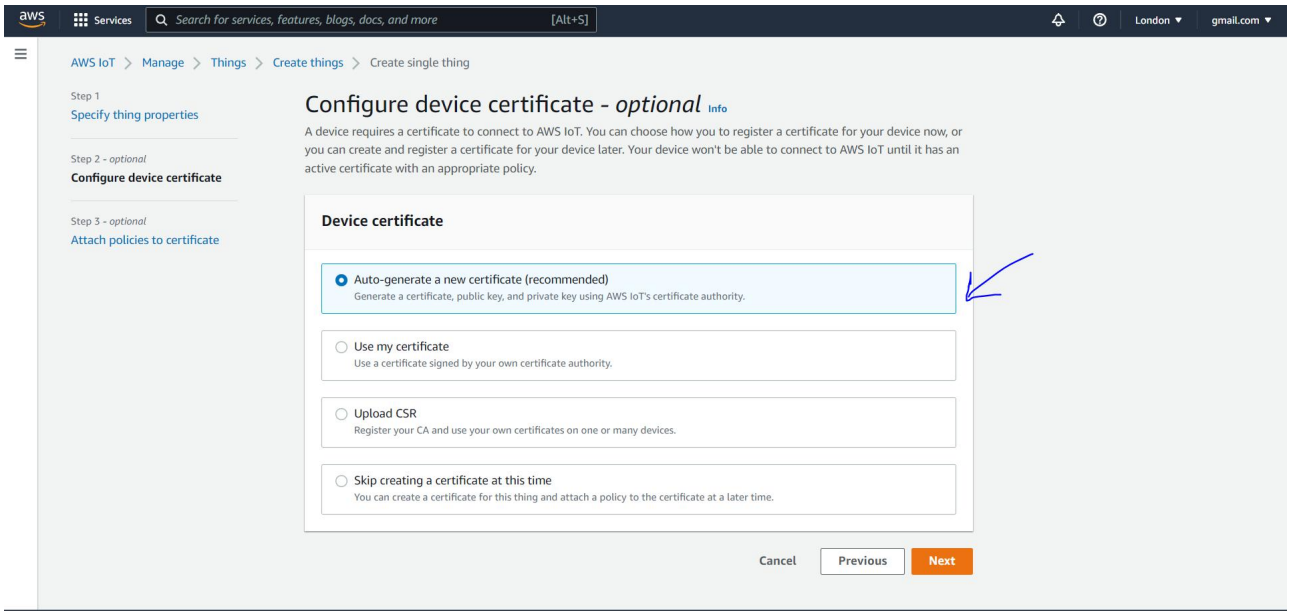


Рисунок 2.7 - Створення сертифікату

Останній етап під’єднання пристрою - додавання політик доступу до конкретного сертифікату безпеки (рис. 2.8). Це дасть змогу обмежити доступ пристрою до інших сервісів, права на зміну, додавання, видалення даних. Щоб помилка в коді, чи зловмисне використання плати не дало змоги нашкодити хмарному рішенню.

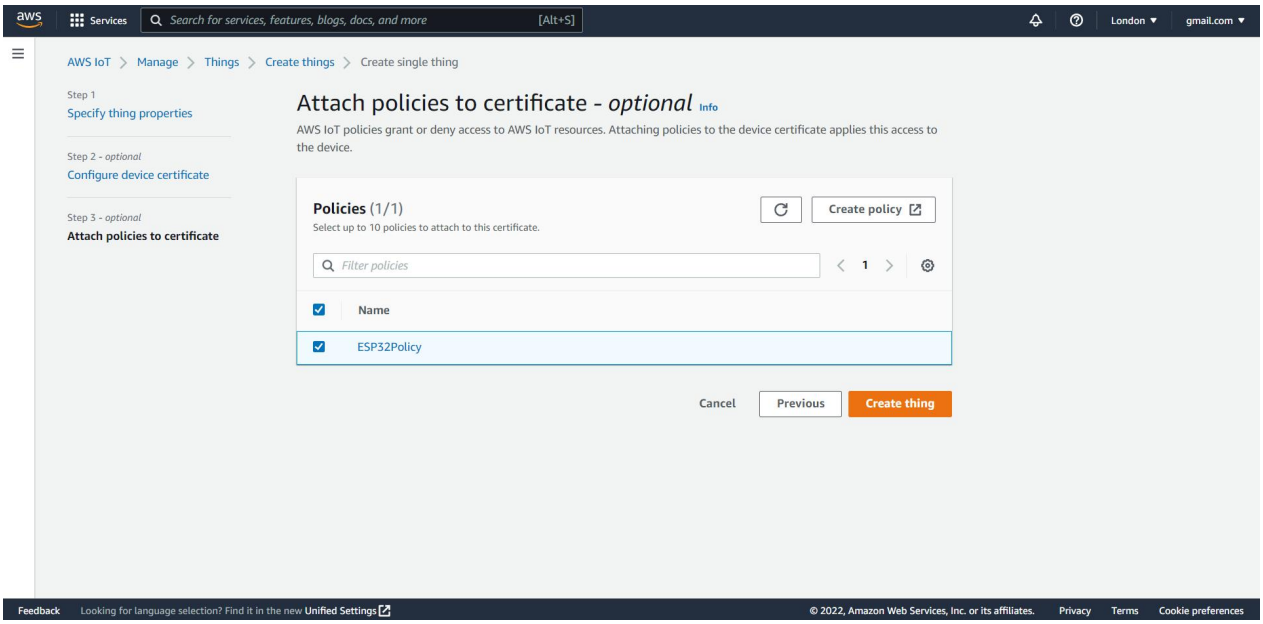


Рисунок 2.8 - Підключення політики до сертифікату безпеки

Після цього ми отримуємо довгожданий сертифікат для скачування (рис. 2.9). В ньому окремо є відкритий і закритий ключ шифрування. Ключі по підключенню до сервісного ендпоінту Amazon Web Services. Обов`язково потрібно скачати саме обидва ключі шифрування, тому що інакше система не випустить з даного вікна, а якщо його закрити, то ніяк ці ключі більше не переглянути заново і прийдеться реєструвати пристрій заново, а старий видалити.

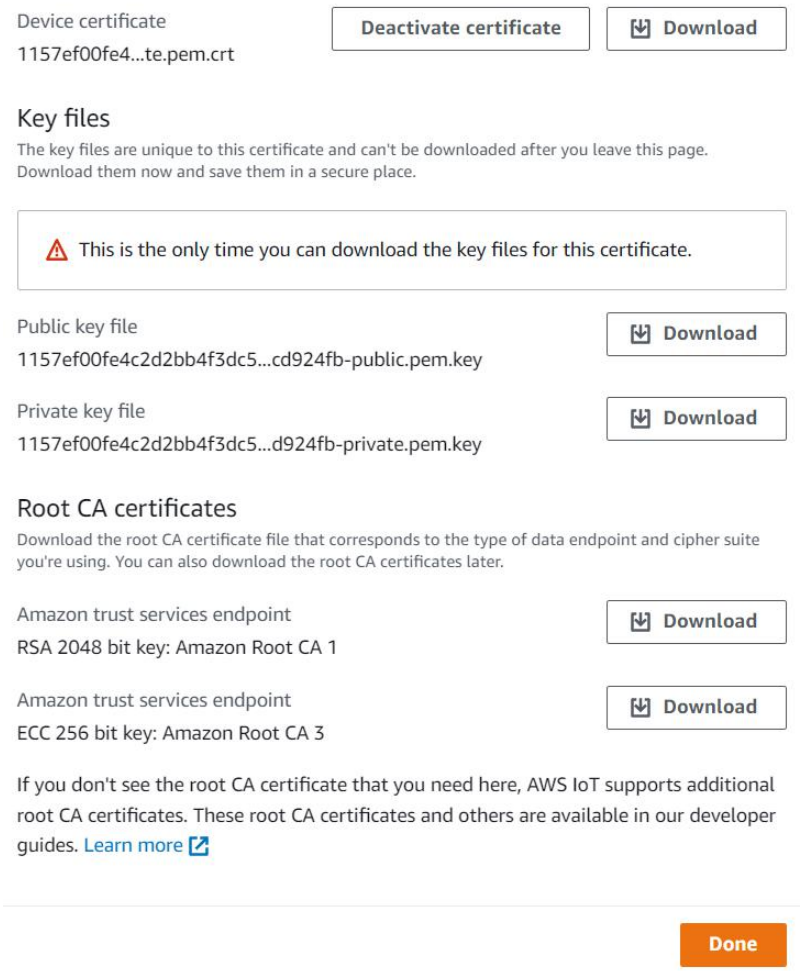


Рисунок 2.9 - Сертифікат безпеки для скачування

2.2.3 Програмування мікроконтролеру.

Для початку було створено окремий файл “secret.h” з внесеними ключами безпеки і унікальним ендпоінтом для підключення пристрою до серверів Amazon Web Services (рис. 2.10).

Додатково було внесені дані для підключення до мережі WIFI, а саме: назва мережі, пароль. Всю конфігурацію було винесено в окремий файл, щоб довгі ключі не заважали програмувати основний функціонал пристрою. Ключі в головній частині прошивки існують у вигляді змінних тепер.

І як додаткова перевага, що основний код можна відправляти іншим людям і не перейматися, що твоє хмарне рішення буде зруйновано. Інша людина вже під себе змінить конфінденційні ключі.

Даний файл розташований поряд з основною програмою прошивки і названий “secret.h”. У разі створення публічного репозиторію Git крайнє рекомендовано додати цей файл в ігнорування, щоб інші користувачі не отримали доступ до хмарного сервісу Вашого акаунту.

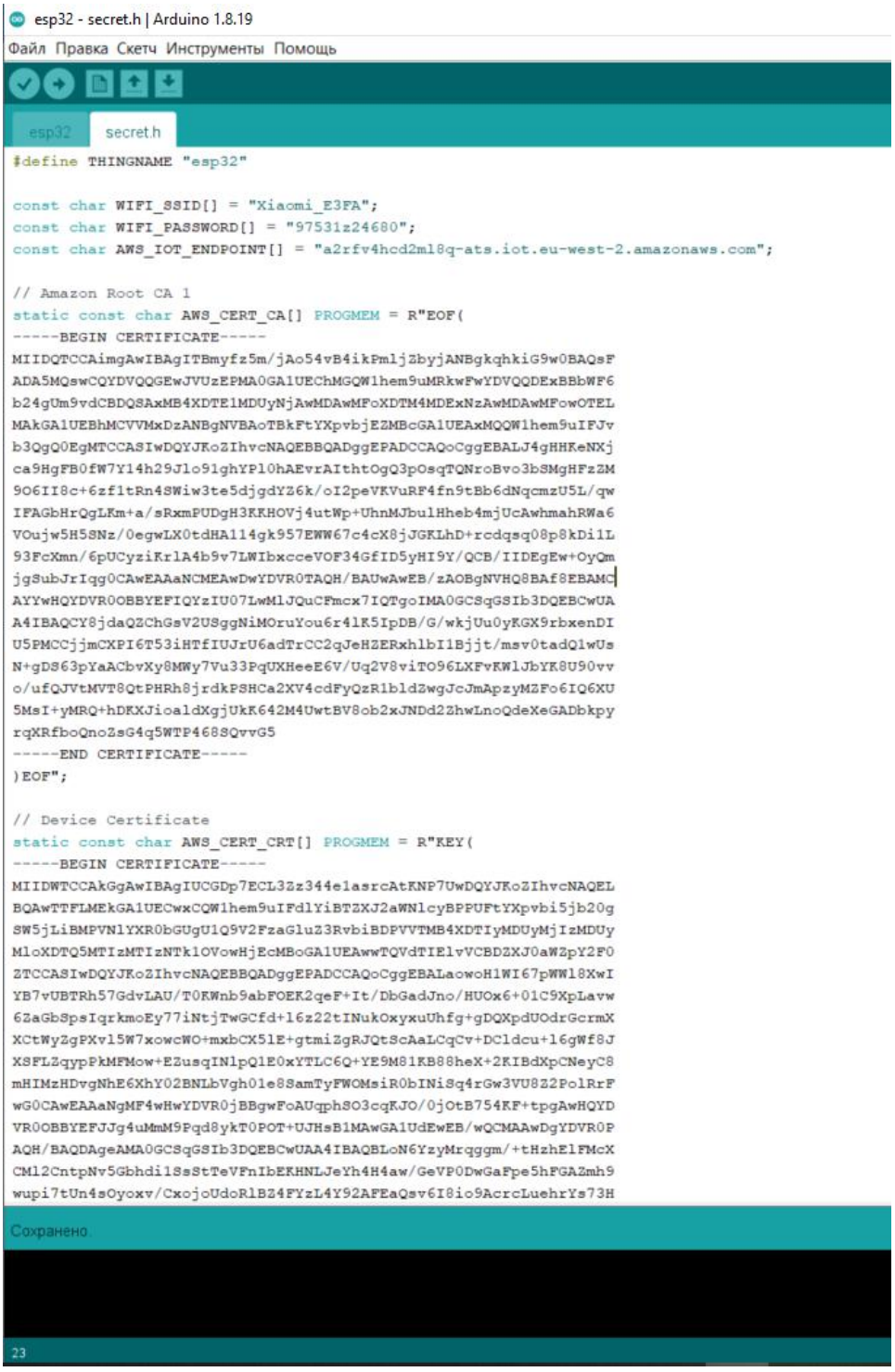


Рисунок 2.10 - Ключі у файлі secret.h

Далі треба реалізувати основну частину програмного забезпечення для мікроконтролера. Все починається з імпортування потрібних бібліотек (рис. 2.11) і внесення у змінні необхідних завідома відомих констант.

Рисунок 2.11 - Імпорт потрібних бібліотек і ініціалізація потрібних змінних

Наступною частиною буде реалізація функції (рис. 2.12), що буде спочатку вимірювати температуру і приводити її у цельсії. Якщо попередня температура була такою ж, тоді нам плата у консолі про це повідомить і не буде нічого відправляти. Це зроблено, щоб економити продуктивність мікроконтролеру і трафік мережі.

Далі задача цієї функції є упакування даних температури у JSON формат та відправити за допомогою протоколу MQTT у до цього встановлений топік за адресою ендпоінта з шифруванням відкритим ключем. Інформація про це відображається у консолі.

Ця функція буде визиватись кожну ітерацію роботи мікроконтролера. Це буде відбуватися у функції Loop.

```
void myTimerEvent()
{
  temperatureSave = analogRead(temperaturePin);
  temp = (double)temperatureSave * 5.0 / 1024;
  temp = temp * 100;

   if (temp != prev_temp){
    StaticJsonDocument<200> doc;
    prev_temp = temp;
    doc["temperature"] = temp;
    char jsonBuffer[512];
    serializeJson(doc, jsonBuffer);
    client.publish(AWS_IOT_PUBLISH_TOPIC, jsonBuffer);
    Serial.print("Published Current Temperature C*: ");
    Serial.println(temp);
   } else {
      Serial.print("C*: ");
      Serial.print(temp);
      Serial.print(" == ");
      Serial.println(prev_temp);
     Serial.println("Same temperature. Dont send to AWS");
  }
}
```

Рисунок 2.12 - Реалізація функції по відправці температури з датчика

Наступною частиною буде функція, яка буде підписуватись на вхідні дані з топіків конфігурації реле (рис. 2.13). В ній реалізований механізм розшифровки формату JSON та діставання звідти потрібних даних.

Для кожного реле роздільний топік. Реле окремо прослуховує вхідну конфігурацію для себе. Ця функція буде вже передана callback функцією до об`єкту визову мережі і її конфігурації.

Заключна функція буде під`єднувати мікроконтролер до хмарного сервісу Amazon (рис. 2.14). В ньому реалізовано з`єднання з мережою WIFI, ініціалізація протоколу MQTT з заданими параметрами ключів шифрування з файлу Secret.h.

Також в ньому реалізована підписка на нові повідомлення з попередньо введених топіків протоколу MQTT. Кожен раз як один з них буде отримувати нові дані, наша попередня функція “MessageHandler” буде оброблювати отриману конфігурацію реле.

```
void messageHandler(char* topic, byte* payload, unsigned int length)
{
  Serial.print("incoming: ");
  Serial.println(topic);

  if ( strstr(topic, "esp32/relay1") )
  {
    StaticJsonDocument<200> doc;
    deserializeJson(doc, payload);
    String Relay_data = doc["status"];
    int r = Relay_data.toInt();
    digitalWrite(Relay1, !r);
    Serial.print("Relay1 - "); Serial.println(Relay_data);
  }

  if ( strstr(topic, "esp32/relay2") )
  {
    StaticJsonDocument<200> doc;
    deserializeJson(doc, payload);
    String Relay_data = doc["status"];
    int r = Relay_data.toInt();
    digitalWrite(Relay2, !r);
    Serial.print("Relay2 - "); Serial.println(Relay_data);
  }

  if ( strstr(topic, "esp32/relay3") )
  {
    StaticJsonDocument<200> doc;
    deserializeJson(doc, payload);
    String Relay_data = doc["status"];
    int r = Relay_data.toInt();
    digitalWrite(Relay3, !r);
    Serial.print("Relay3 - "); Serial.println(Relay_data);
  }
```

Рисунок 2.13 - Функція перевірки нової конфігурації для реле

```
void connectAWS()
{
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.println("Connecting to Wi-Fi");

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  net.setCACert(AWS_CERT_CA);
  net.setCertificate(AWS_CERT_CRT);
  net.setPrivateKey(AWS_CERT_PRIVATE);
  client.setServer(AWS_IOT_ENDPOINT, 8883);
  client.setCallback(messageHandler);
  Serial.print("Connecting to AWS IOT");

  while (!client.connect(THINGNAME)) {
    Serial.print(".");
    delay(100);
  }

  if (!client.connected()) {
    Serial.println("AWS IoT Timeout!");
    return;
  }

  client.subscribe(AWS_IOT_SUBSCRIBE_TOPIC1);
  client.subscribe(AWS_IOT_SUBSCRIBE_TOPIC2);
  client.subscribe(AWS_IOT_SUBSCRIBE_TOPIC3);
  client.subscribe(AWS_IOT_SUBSCRIBE_TOPIC4);
  Serial.println("AWS IoT Connected!");
}
```

Рисунок 2.14. - Функція підключення мікроконтролеру до мережевих служб

У стандартній функції “setup” відбувається конфігурація потрібних пінів, підключення до серверів Amazon і налаштування серіал порта (рис. 2.15). Ця функція виконується один раз перед запуском основного циклу роботи.

```
void setup(){
  Serial.begin(9600);
  pinMode(Relay1, OUTPUT);
  pinMode(Relay2, OUTPUT);
  pinMode(Relay3, OUTPUT);
  pinMode(Relay4, OUTPUT);

  connectAWS();
}
```

Рисунок 2.15 - функція “setup”

Функції перевірки нової конфігурації реле та вимірювання температури для відправки безперервно визиваються у безкінечному циклі “loop” (рис. 2.16).

```
void loop()
{
  client.loop();
  myTimerEvent();
}
```

Рисунок 2.16. - функція “loop”

2.2.4 Програмування серверної частини Amazon.

Тепер час розглянути серверне рішення (рис. 2.17), яке буде отримувати дані температури і приймати рішення по конфігурації реле. Дане серверне рішення спрацьовує лише у відповідь на тригер-подію, що дає змогу економити комп`ютерні потужності, а значить гроші. У випадку Amazon Web Services ця послуга, взагалі, безкоштовна.

У програмі є низка логічних перевірок вхідних даних, у разі спрацювання одного з них, формується масив конфігурації для реле. Цей масив перебирається потім у циклі і кожен з реле отримує свій стан роботи.

Рисунок 2.17. - Реалізація лямбда функції

2.2.5 Розробка бази даних.

Всі ці дані обов`язково треба зберігати і показувати користувачу. Інакше реалізація не має сенсу. Для цього було розроблено модель бази даних, що на рис. 2.18. У ній є можливість створення не одного мікроконтролеру, той у свою чергу прив`язаний до конкретно визначеної локації.

Цей головний пристрій може мати безліч сенсорів і реле. Сенсор може мати певний тип, що має бути обраним зі списку. Також є акумулятори даних для реле і сенсорів, що дасть змогу переглядати історію змін даних для цих приладів з прив`язкою до часу. Можна придумати безліч вибірок таких даних для комфортного моніторингу користувачем.

У базі даних реалізовано 7 сутностей з такими полями:

- мікроконтролер:
  - ім’я;

    - публічний ключ шифрування Amazon;
    - ендпоінт Amazon;
    - на вибір одна з запропонованих локацій прив’язки мікроконтролеру до певної локації;
- локація:
    - ім’я;
- датчик:
    - ім’я;
    - тип, що обирається на вибір з уже передвстановлених;
    - мікроконтролер, до якого було підключено датчик;
- тип датчику:
    - ім’я;
- збереження даних з датчиків:
    - датчик, зв’язок з уже існуючим датчиком;
    - значення вимірів;
    - час виміру;
- реле:
    - ім’я пристрою для увімкнення;
    - мікроконтролер, до якого було підключено реле;
- збереження конфігурацій реле:
    - реле, зв’язок з уже існуючим реле;
    - значення стану реле;
    - час запису.

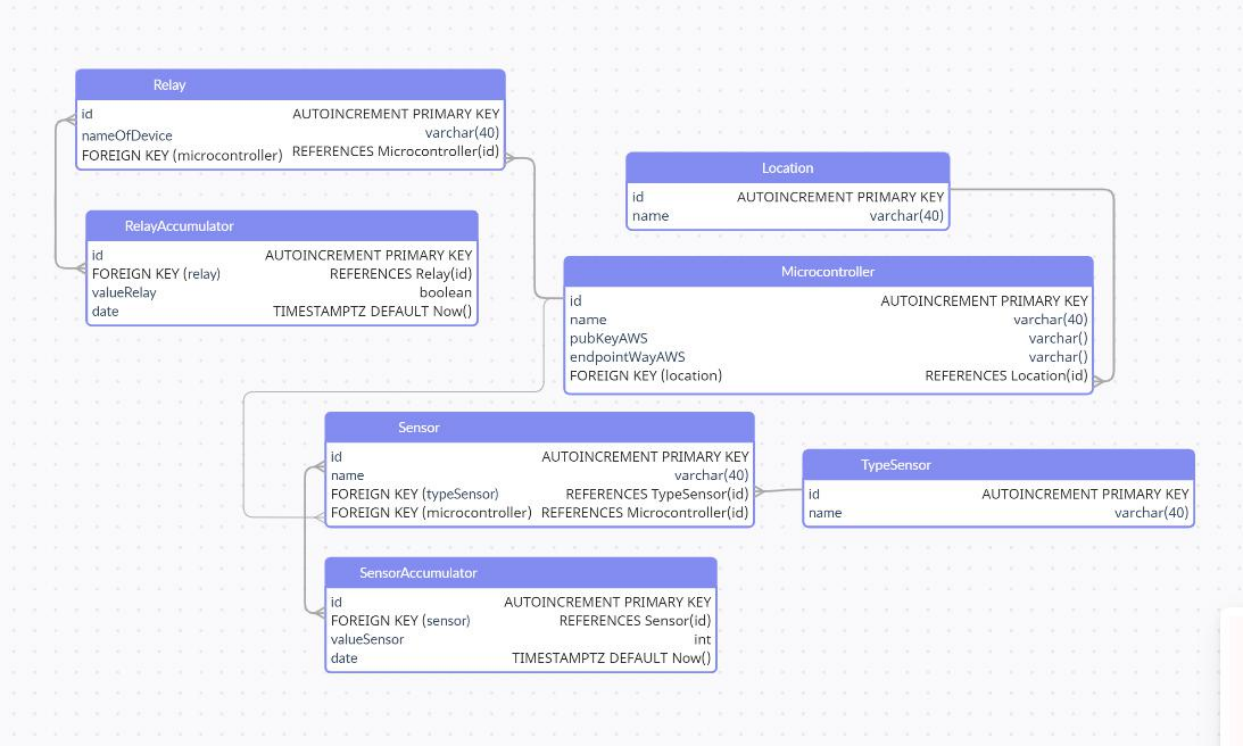


Рисунок 2.18. - Модель бази проекту керування температурою

2.2.6 Розробка програми для бази даних.

Для написання програми, яка буде працювати з розробленою вище базою даних - було обрано фреймворк Django (рис. 2.19). У ньому вже встановлено більшість потрібних функцій для програми. А саме: адмінська панель з авторизацією, відображення необхідних таблиць, додавання, видалення, оновлення.

На головній панелі є сутність мікроконтролеру, список створених екземплярів. Налаштування реле, датчиків і вся потрібна інформація.

Було використано СУБД SQLite, враховуючи, що проект розроблений для особистого користування і не потребує особливих функцій, по типу паралельний вводів, виводів інформації. Це рішення поставляється “з коробки” у системі Django. Перевагою цього рішення є маленький розмір СУБД, швидка робота, малий набір функцій, для швидкої реалізації проекту.

База даних знаходиться локально у папці з веб додатком. Після завантаження проекту на віддалений сервер, все буде працювати як і локально.

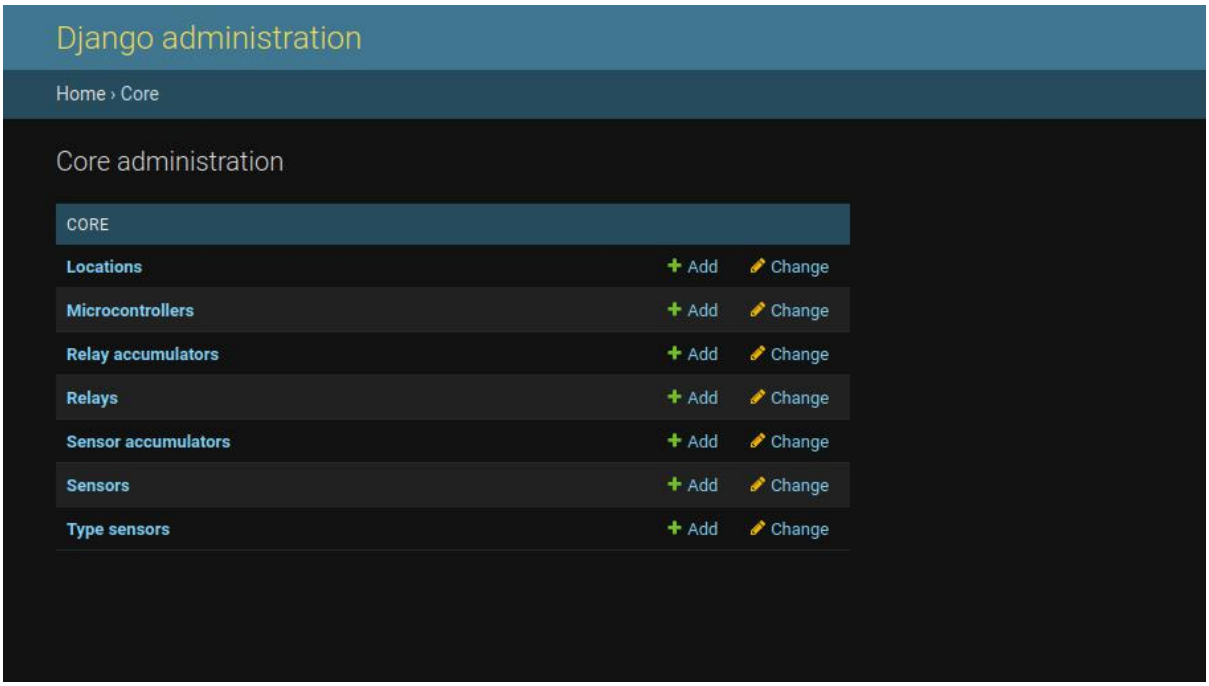


Рисунок 2.19. - Розроблена панель бази даних

Висновок до розділу 2

У даному розділі було розроблено план і модель квартири для автоматизації контролю температури. Тепер наглядно помітно розташування всіх приладів контролю температури і від цього можна планувати прокладання електричних кабелів.

Виконано аналіз та обгрунтування доцільності використання різних рішень для виконання проєктної розробки

Було реалізовано апаратне і програмне рішення для цього проекту і детально розглянуті рішення, які використовувались при роботі. Розроблено проект бази даних для моніторингу системи і реалізовано для неї веб-додаток.

РОЗДІЛ 3. ЕКСПЕРИМЕНТ ТА АНАЛІЗ ЕФЕКТИВНОСТІ РІШЕННЯ

3.1 Опис реалізованої системи.

У реалізованій системі за проектом квартири (рис. 3.1) є мікроконтролер, датчик на ньому і реле керування приладами опалення.

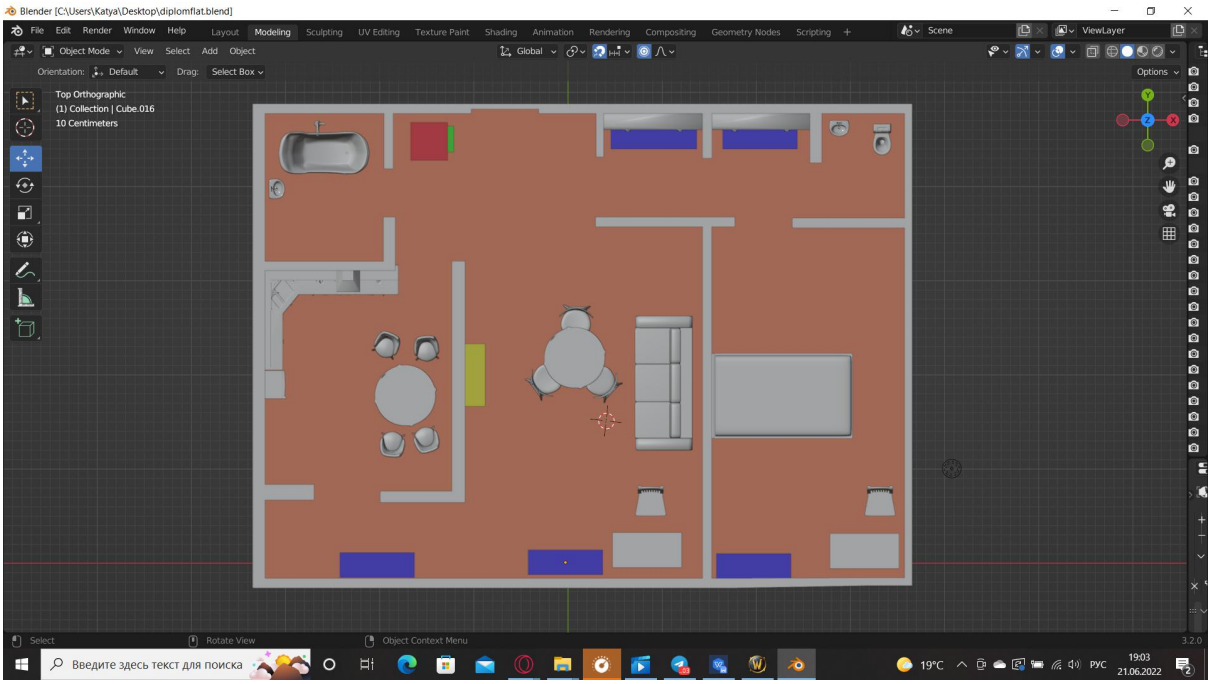


Рисунок 3.1 - Проект квартири

Користувач встановлює плату мікроконтролеру з датчиком. Під'єднує реле до кінцевих пристроїв опалення. Вмикає все до мережі і чекає. Далі отримує дані датчику у кабінеті Amazon, а ще у кабінеті користувача у веб-додатку, що реалізовано на основі побудованої бази даних.

У веб-додатку користвач має всі компоненти системи, а далі зможе керувати кожним елементом окремо. Для цього треба зайти авторизуватись у панелі користувача (рис. 3.2) за адресою http://127.0.0.1:8000/admin (логін: root, пароль: root).

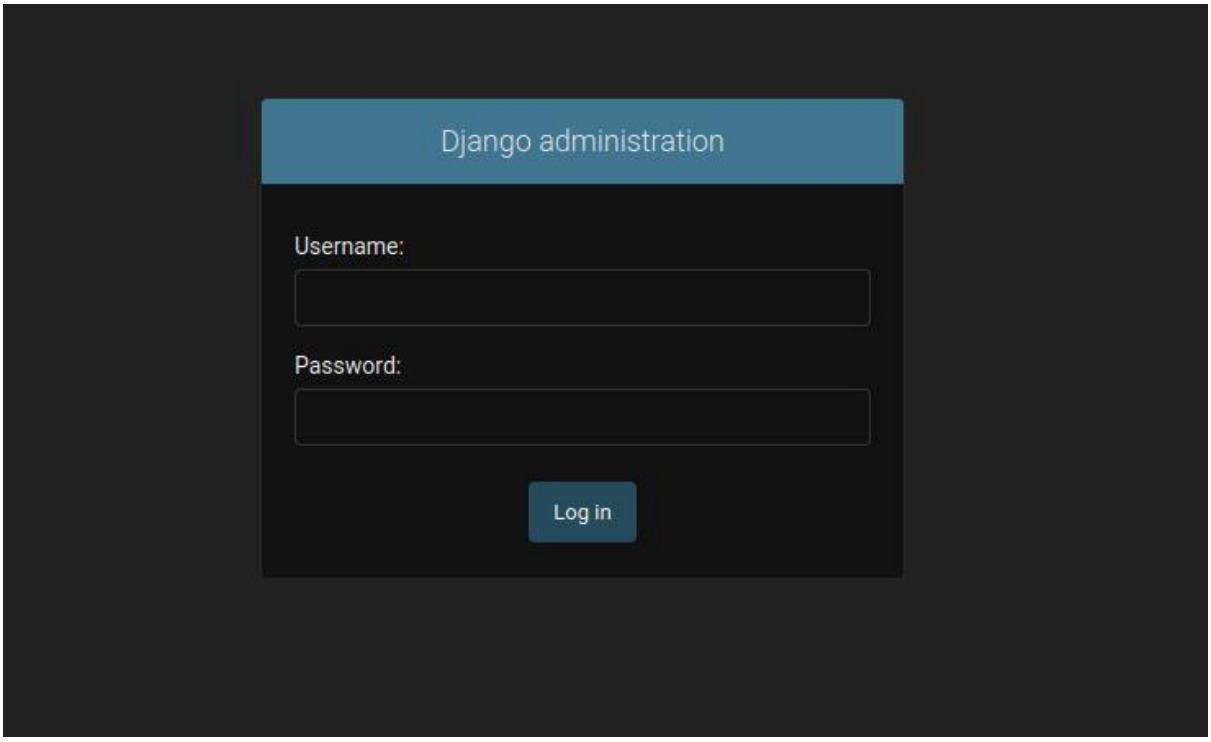


Рисунок 3.2 – Панель авторизації

Після авторизації ми отримуємо повну панель керування користувача (рис. 3.3). У ній буде налаштування всієї системи.

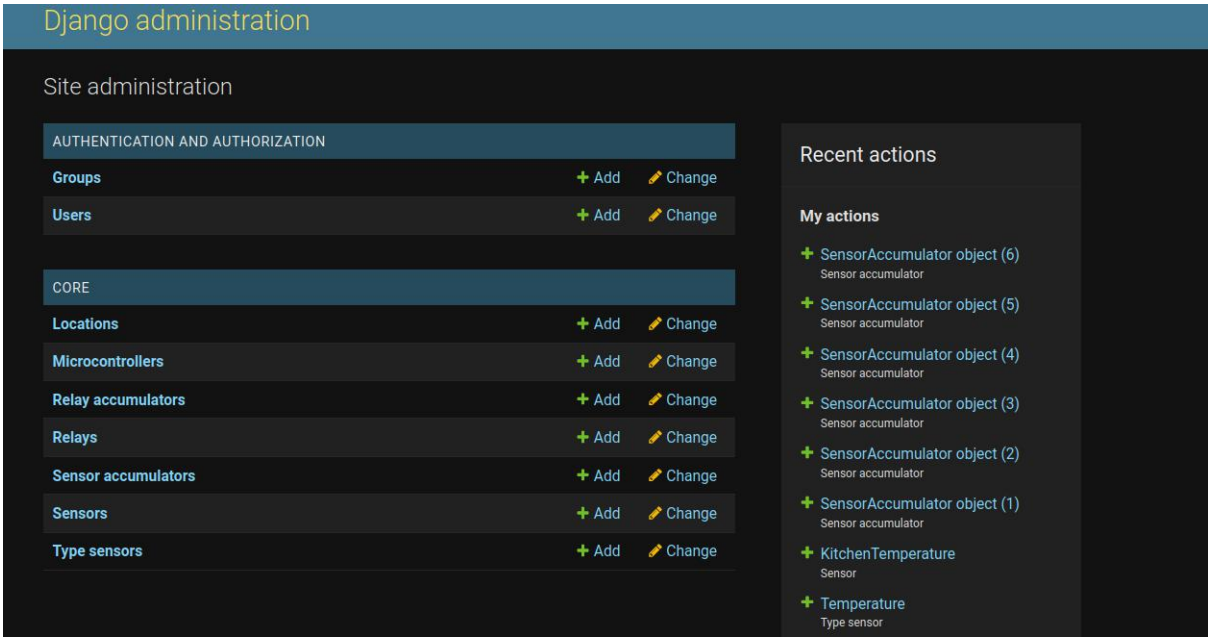


Рисунок 3.3 – Панель керування

Перше, шо треба налаштувати, це локацію, до якої буде прив’язка нашої системи. Користувач має зайти у розділ “Locations” (рис. 3.4). Тут буде

показуватись всі створені локації до цього. Так само буде у розділах інших сутностей.

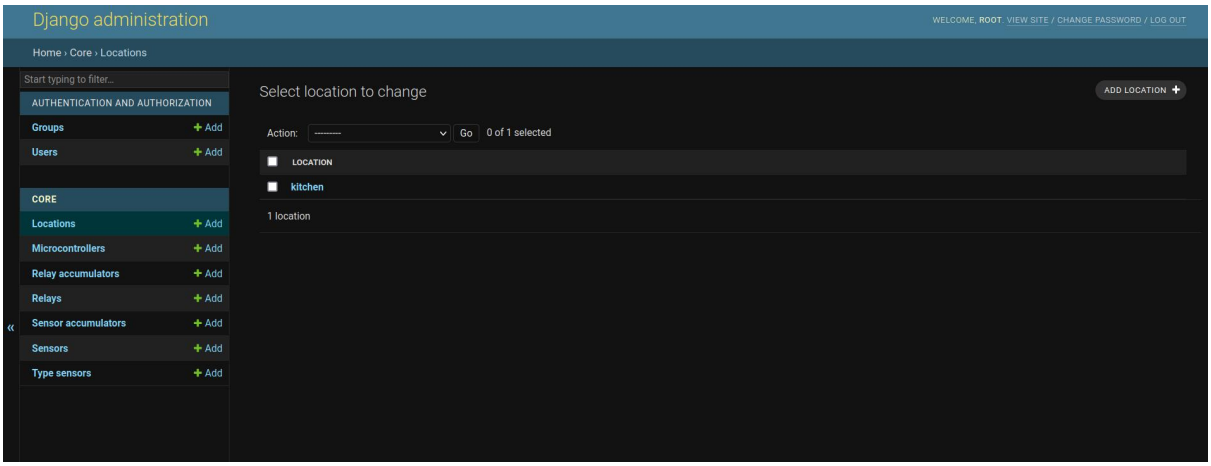


Рисунок 3.4 – Панель списку локацій

Далі треба зайти у “Add location +”, (рис. 3.5), внести дані про назву локації і натиснути “Save”. Локація створена і ми маємо перейти назад у панель керування.

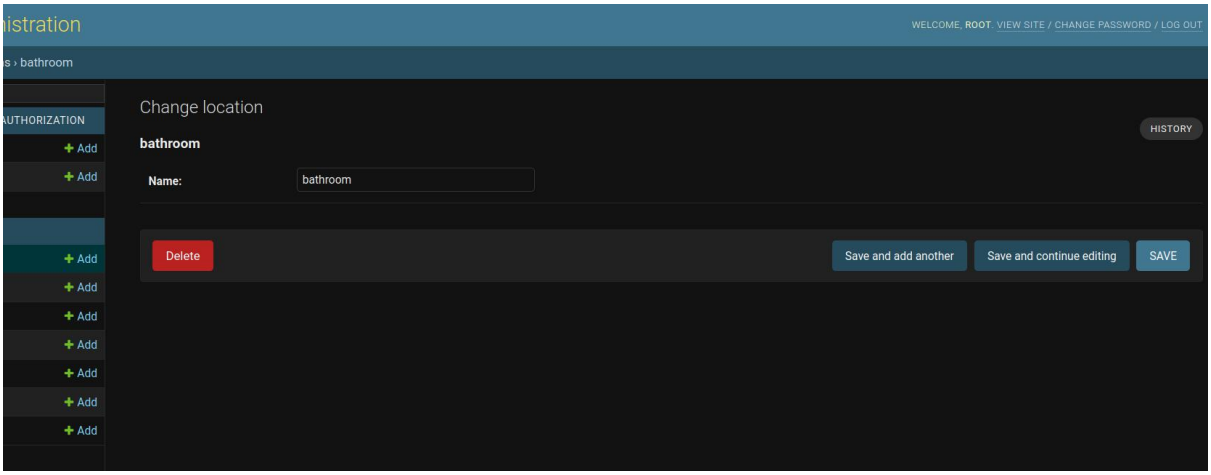


Рисунок 3.5 – Панель створення локацій

Наступне буде додавання мікроконтролеру у систему. Для цього на панелі треба перейти у розділ “Microcontrollers”. Там буде відображено весь список зареєстрованих попередньо пристроїв. Додаємо новий, (рис. 3.6). У поля маємо вказувати ключ шифрування і ендпоінт для відправки даних на сервер. Цю інформацію отримали у кабінеті Amazon, що було описано у 2.2.2 розділі. Також треба вказати попередньо створену локацію.

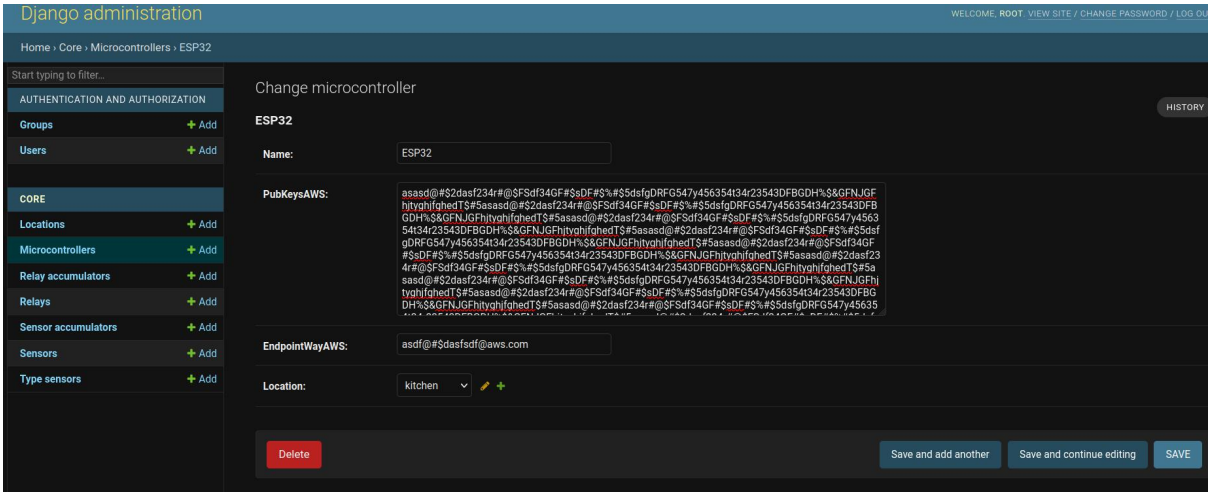


Рисунок 3.6 – Панель створення мікроконтролеру

Таким самим способом створюється тип датчику, датчик, реле. У цих сутностях теж треба буде обирати зі списку до якого саме мікроконтролеру вони відносяться. Виконувати реєстрацію треба саме у цьому порядку, щоб не порушувати зв’язки таблиць у базі даних.

Після цих налаштувань, користувач зі свого кабінету користувач бачить у реальному часі дані з датчику, що у розділі панелі “Sensor accumulators” (рис. 3.7), який саме датчик передає інформацію, до якого мікроконтролеру він відноситься і до якої локації відноситься ця система.

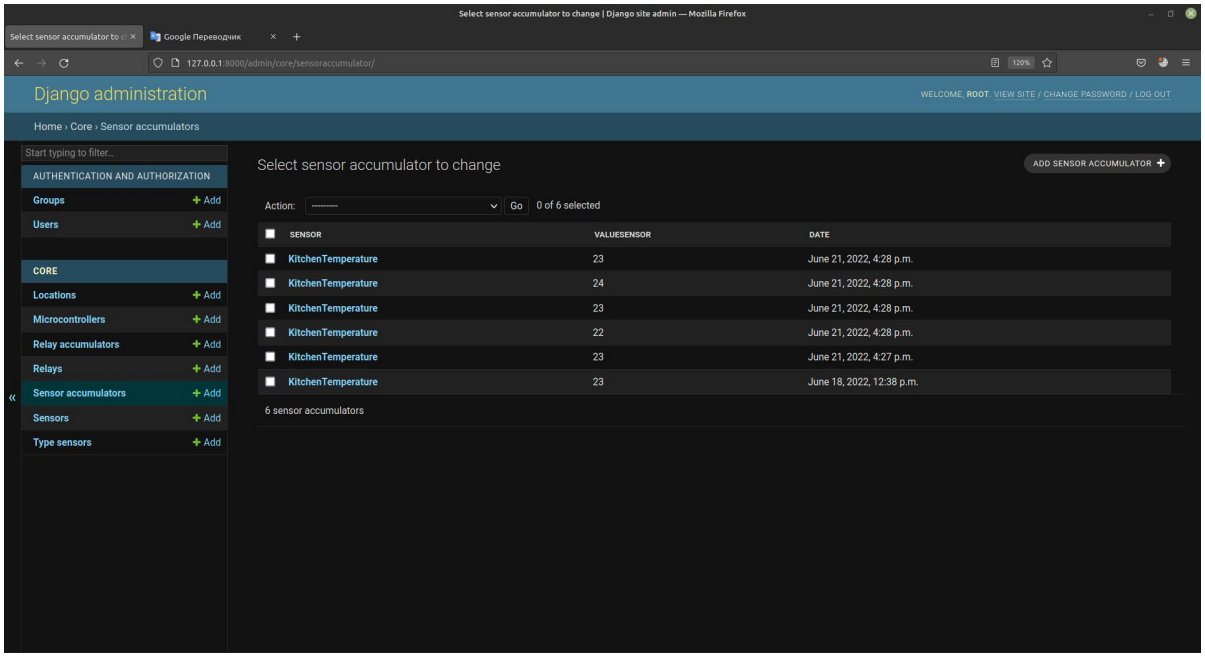


Рисунок 3.7 – Розділ моніторингу датчиків

Також користувач має повний контроль над системою і можна змінити статус реле вручну у веб-додатку. Для цього він має перейти у розділ “Relay accumulators”. У цьому розділі показана історія змін статусу кожного реле. Додаємо новий актуальний статус за допомогою кнопки “Add relay accumulator” (рис. 3.8).

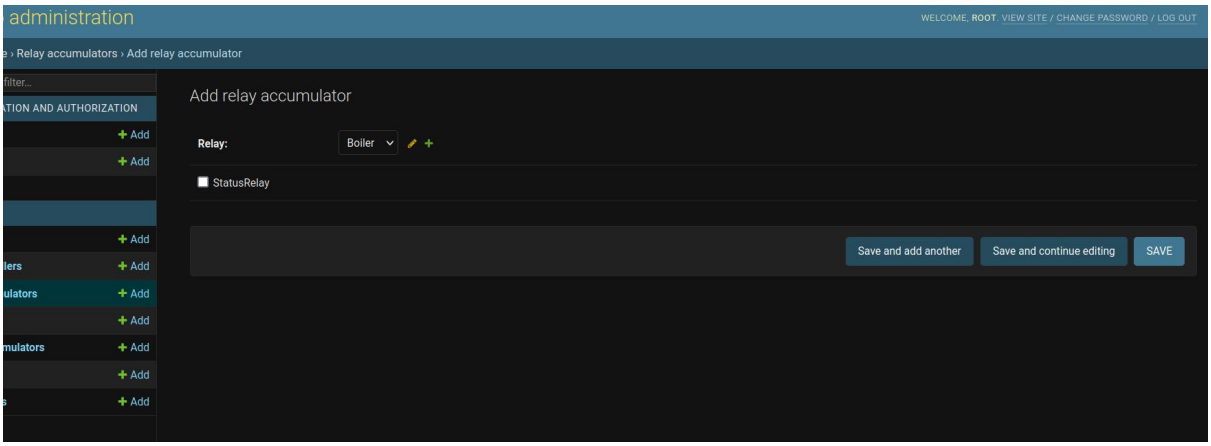


Рисунок 3.8 – Панель зміни статусу реле

Користувачу доступне редагування температурного режиму на панелі Amazon. Треба зайти у сервіс Lambda і відредагувати діапазони як тільки заманеться (рис. 3.9).

Рисунок 3.9 – Конфігурація температури у сервісі Amazon Lambda

Також на сторінці “Sensor accumulators” є можливість вмикання аналітики даних з сенсорів за добу. Ця панель знаходиться нижче списку даних (рис. 3.10).

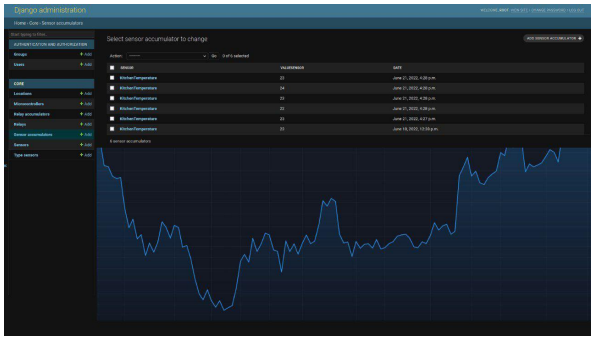


Рисунок 3.10 – Побудова аналітики даних датчику

За допомогою даної системи користувач отримує:

- автоматизовану систему керування опаленням квартири;

- особистий кабінет користувача у веб-додатку, що дає бачити інформацію про увімкнені прилади опалення і дані датчику температури;
- легку у маштабованні систему, можна додати на кожну кімнату подібну систему і окремо регулювати температуру у них і цим досягти економію енергії.

3.2 Тестування приладу.

Після прошивки мікроконтролеру потрібно перевірити правильність з'єднання контактів за допомогою мультиметра в режимі "прозвінка" (даний режим дозволить нам не тільки перевірити правильність з'єднання, а й виявити проблеми, такі як коротке замикання). Потім під`єднав у USB живлення 5В, плата має підключитись до інтернету і в Амазоні мають почати приходити дані температури.

Після вмикання плати до живлення можна побачити постійне повідомлення з серіал порта про температуру у даний момент. Цю ж інформацію можна побачити у консолі сервісу інтернет речей Amazon Web Services. Реле одразу увімкнулось у конфігурацію відповідно до потрібної конфігурації для даної температури.

Якщо закрити датчик рукою, то температура буде змінюватись у більшу сторону. Конфігурація реле відповідно зміниться теж.

Керувати платою можна через Amazon та при зміні температури. При команді увімкнення якогось каналу відбувається характерний звук і споживач на вивідному порті вмикається (рис. 3.11).

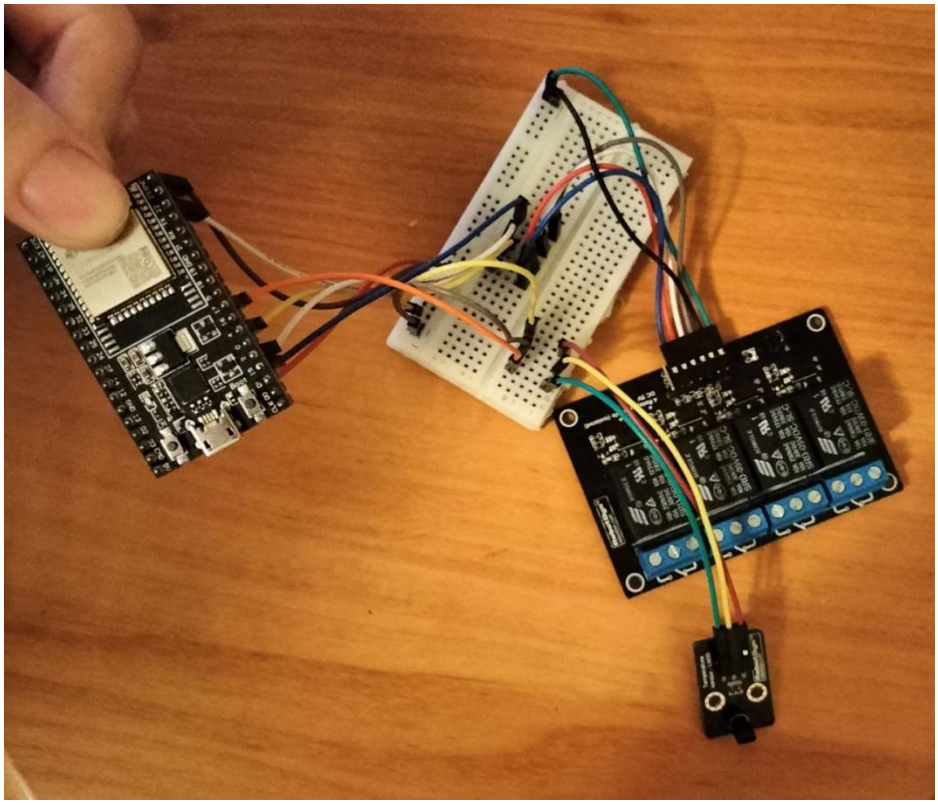


Рисунок 3.11 - Готовий виріб

Насправді ми розглянули найбазовіший варіант тестування плати, а взагалі є багато пунктів для перевірки правильної роботи обладнання.

Після тестування - все працює. Рішення являється роботоспроможним.

3.3 Введення мікроконтролеру в експлуатацію.

Для того, щоб все працювало, було проведено ряд дій, які забезпечили правильне функціонування мікроконтролеру. А саме:

- непомітне розташування датчику температури у квартирі;
- з’єднання реле з пристроями регулювання температури;
- під’єднання мікроконтролеру до живлення;
- очікування індикації з`єднання з мережею WIFI;
- очікування індикації з`єднання з сервером Amazon.

Налаштувати його було зовсім нескладно, підключення до системи відбулося майже відразу.

3.4 Недоліки системи.

Недоліком є постійне перемикання конфігурації реле, якщо температура коливається між двома граничними поділками температури. Це можна виправити доданням плаваючих границь, щоб збільшити час зміни конфігурації і не дратувати постійно користувачів звуками реле.

Також недолік є відсутність на платі зовнішньої антени, що сильно зменшує область використання. Інколи мікроконтролер зовсім від`єднується від мережі. Для цього можна купити дану запчастину і встановити у відповідний роз`єм.

Висновок до розділу 3

У цьому розділі було розглянуто тестування приладу, а саме, перевірка на коротке замикання, перевірка на можливість з’єднання приладу до мережі інтернет, перевірка на можливість авторизації до серверів Amazon, що допомогло для введення в експлуатацію.

Додатково було розглянуто недоліки системи та можливість їх усунення, а саме:

- часте перемикання конфігурації реле;
- мала відстань для з’єднання мікроконтролеру з мережою WIFI.

Запропоновано рішення недоліків системи, що дасть змогу покращити продуктивність проекту і стабільність роботи:

- додання плаваючих границь, щоб збільшити час зміни конфігурації реле;
- встановити антену, для більшої відстані з’єднання.

Це дало змогу всебічно подивитися на проект.

ВИСНОВОК

У даній кваліфікаційній роботі бакалавра було запропоновано реалізацію проекту автоматичної регуляції температури у квартирі з інтеграцією хмарних рішень Amazon. Додатково було реалізовано модель квартири з розташованими елементами регуляції температури, база даних і веб-додаток, для моніторингу даного проекту у реальному часі.

Виконано:

1. Розроблено архітектуру рішення системи автоматизації опалення.
2. Розроблено алгоритм роботи системи автоматизації.
3. Розроблено квартиру з автоматичним регулюванням температури.
4. Створенно проект інтернет речей на базі Amazon, для автоматизації термоконтролю.
5. Розроблено бази даних для веб-додатку моніторингу системи автоматичного опалення.
6. Розроблено програму для бази даних з функцією моніторингу.
7. Тестування системи і введення в експлуатацію.

Під час реалізації даного проекту отримано наступні результати:

1. На практиці досліджено взаємодію хмарних рішень з мікроконтролерами за допомогою мережі інтернет.
2. Реалізований план квартири з проектуванням трьохвимірної моделі.
3. Базу даних, для моніторингу компонентів системи.
4. Реалізований веб-додаток, моніторингу компонентів системи.
5. Запрограмована плата розробника на базі ESP32, що працює з хмарним рішенням Amazon.

Мета кваліфікаційної роботи бакалавра досягнута.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Документація AWS IoT Core [Електронний ресурс] - Режим доступу до ресурсу: https://docs.aws.amazon.com/iot/index.html (дата звернення 10.05.2022)
2. Документація до esp32 [Електронний ресурс] - Режим доступу до ресурсу: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf (дата звернення 14.05.2022)
3. Стаття "Автоматичні системи: опалення та водопостачання для будинку" [Електронний ресурс] - Режим доступу до ресурсу: https://aw-therm.com.ua/nasosi-dlya-opalennya-ta-vodopostachannya-budinku/ (дата звернення 17.05.2022)
4. Офіційний сайт Philips [Електронний ресурс] - Режим доступу до ресурсу: https://www.philips-hue.com/ru-ru (дата звернення 21.05.2022)
5. Офіційний сайт Amazon AWS [Електронний ресурс] - Режим доступу до ресурсу: https://aws.amazon.com/ru/ (дата звернення 22.05.2022)
6. Документація до Amazon AWS [Електронний ресурс] - Режим доступу до ресурсу: https://docs.aws.amazon.com/index.html?nc2=h_ql_doc_do (дата звернення 22.05.2022)
7. Дані про сайт для створення діаграм Draw.io [Електронний ресурс] / Режим доступу: https://app.diagrams.net/ (Дата звернення 22.05.2022)
8. Документація до Django [Електронний ресурс] - Режим доступу до ресурсу: https://django.fun/docs/django/ru/4.0/ (дата звернення 23.05.2022)
9. Дані про сайт для створення діаграм баз даних Creately [Електронний ресурс] - Режим доступу до ресурсу: https://app.creately.com (дата звернення 27.05.2022)
10. Документація до мови програмування Python [Електронний ресурс] - Режим доступу до ресурсу: https://www.python.org/doc/ (дата звернення 30.05.2022)

11. Документація до мови програмування C++ [Електронний ресурс] - Режим доступу до ресурсу: https://docs.microsoft.com/en-us/cpp/?view=msvc-170 (дата звернення 30.05.2022)
12. Огляд до СУБД SQLite [Електронний ресурс] - Режим доступу до ресурсу: https://habr.com/ru/post/149356/ (дата звернення 01.06.2022)
13. Документація SQL на прикладі MySQL [Електронний ресурс] - Режим доступу до ресурсу: https://dev.mysql.com/doc/ (дата звернення 02.06.2022)
14. Документація до Arduino IDE [Електронний ресурс] - Режим доступу до ресурсу: https://docs.arduino.cc/ (дата звернення 02.06.2022)
15. Огляд кондиціонерів [Електронний ресурс] - Режим доступу до ресурсу: https://vencon.ua/articles/reyting-luchshikh-konditsionerov (дата звернення 05.06.2022)

Додаток А

Код програми для роботи мікроконтролеру

Листів 4

Розробник

Євгеній МАТВІЄНКО

Керівник

Мирослава ГЛАДКА

Київ – 2022

Код від програми для роботи мікроконтролеру

```
#include "secret.h"
#include <WiFiClientSecure.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>
#include <WiFi.h>
#define Relay1 15
#define Relay2 2
#define Relay3 4
#define Relay4 22
#define AWS_IOT_PUBLISH_TOPIC   "esp32/pub"
#define AWS_IOT_SUBSCRIBE_TOPIC1 "esp32/relay1"
#define AWS_IOT_SUBSCRIBE_TOPIC2 "esp32/relay2"
#define AWS_IOT_SUBSCRIBE_TOPIC3 "esp32/relay3"
#define AWS_IOT_SUBSCRIBE_TOPIC4 "esp32/relay4"
int temperaturePin = 16;
int temperatureSave;
double temp;
double prev_temp = 1.0;
WiFiClientSecure net = WiFiClientSecure();
PubSubClient client(net);
void myTimerEvent()
{
  temperatureSave = analogRead(temperaturePin);
  temp = (double)temperatureSave * 5.0 / 1024;
  temp = temp * 100;
   if (temp != prev_temp){
    StaticJsonDocument<200> doc;
    prev_temp = temp;
    doc["temperature"] = temp;
    char jsonBuffer[512];
    serializeJson(doc, jsonBuffer);
    client.publish(AWS_IOT_PUBLISH_TOPIC, jsonBuffer);
    Serial.print("Published Current Temperature C*: ");
    Serial.println(temp);
   } else {
      Serial.print("C*: ");
      Serial.print(temp);
      Serial.print(" == ");
      Serial.println(prev_temp);
     Serial.println("Same temperature. Dont send to AWS");
  }
}
void messageHandler(char* topic, byte* payload, unsigned int length)
{
 Serial.print("incoming: ");
 Serial.println(topic);
 if ( strstr(topic, "esp32/relay1") )
 {
   StaticJsonDocument<200> doc;
   deserializeJson(doc, payload);
   String Relay_data = doc["status"];
   int r = Relay_data.toInt();
   digitalWrite(Relay1, !r);
   Serial.print("Relay1 - "); Serial.println(Relay_data);
 }
 if ( strstr(topic, "esp32/relay2") )
 {
   StaticJsonDocument<200> doc;
```

```
    deserializeJson(doc, payload);
    String Relay_data = doc["status"];
    int r = Relay_data.toInt();
    digitalWrite(Relay2, !r);
    Serial.print("Relay2 - "); Serial.println(Relay_data);
   }
   if ( strstr(topic, "esp32/relay3") )
   {
    StaticJsonDocument<200> doc;
    deserializeJson(doc, payload);
    String Relay_data = doc["status"];
    int r = Relay_data.toInt();
    digitalWrite(Relay3, !r);
    Serial.print("Relay3 - "); Serial.println(Relay_data);
   }
   if (strstr(topic, "esp32/relay4") )
   {
    StaticJsonDocument<200> doc;
    deserializeJson(doc, payload);
    String Relay_data = doc["status"];
    int r = Relay_data.toInt();
    digitalWrite(Relay4, !r);
    Serial.print("Relay4 - "); Serial.println(Relay_data);
   }
}
void connectAWS()
{
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.println("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  net.setCACert(AWS_CERT_CA);
  net.setCertificate(AWS_CERT_CRT);
  net.setPrivateKey(AWS_CERT_PRIVATE);
  client.setServer(AWS_IOT_ENDPOINT, 8883);
  client.setCallback(messageHandler);
  Serial.print("Connecting to AWS IOT");
  while (!client.connect(THINGNAME)) {
    Serial.print(".");
    delay(100);
  }
  if (!client.connected()) {
    Serial.println("AWS IoT Timeout!");
    return;
  }
  client.subscribe(AWS_IOT_SUBSCRIBE_TOPIC1);
  client.subscribe(AWS_IOT_SUBSCRIBE_TOPIC2);
  client.subscribe(AWS_IOT_SUBSCRIBE_TOPIC3);
  client.subscribe(AWS_IOT_SUBSCRIBE_TOPIC4);
  Serial.println("AWS IoT Connected!");
}
void setup(){
  Serial.begin(9600);
  pinMode(Relay1, OUTPUT);
  pinMode(Relay2, OUTPUT);
  pinMode(Relay3, OUTPUT);
  pinMode(Relay4, OUTPUT);
```

```
  connectAWS();
}
void loop()
{
  client.loop();
  myTimerEvent();
}
```

Код від програми для роботи мікроконтролеру

```
import json
import boto3
def lambda_handler(event, context):
    temperature = int(event['temperature'])
    topics = []
    set_relay = []
    if temperature < 10:
        topics.append('1')
        set_relay.append('1')
        topics.append('2')
        set_relay.append('0')
        topics.append('3')
        set_relay.append('0')
        topics.append('4')
        set_relay.append('1')
    elif temperature >= 10 and temperature < 27:
        topics.append('1')
        set_relay.append('0')
        topics.append('2')
        set_relay.append('1')
        topics.append('3')
        set_relay.append('1')
        topics.append('4')
        set_relay.append('0')
    elif temperature >= 27:
        topics.append('1')
        set_relay.append('0')
        topics.append('2')
        set_relay.append('0')
        topics.append('3')
        set_relay.append('0')
        topics.append('4')
        set_relay.append('0')

    mqtt = boto3.client('iot-data')
    for index, relay in enumerate(set_relay):
        mqtt.publish(
            topic='esp32/relay'+ str (index+1),
            qos=0,
            payload=json.dumps({'status': relay})
        )
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps({'topics': topics, 'set_relay': set_relay})
        # 'body': json.dumps(response)
    }
```

Додаток Б

Основні слайди з презентації для захисту дипломної роботи

Листів 2

Розробник

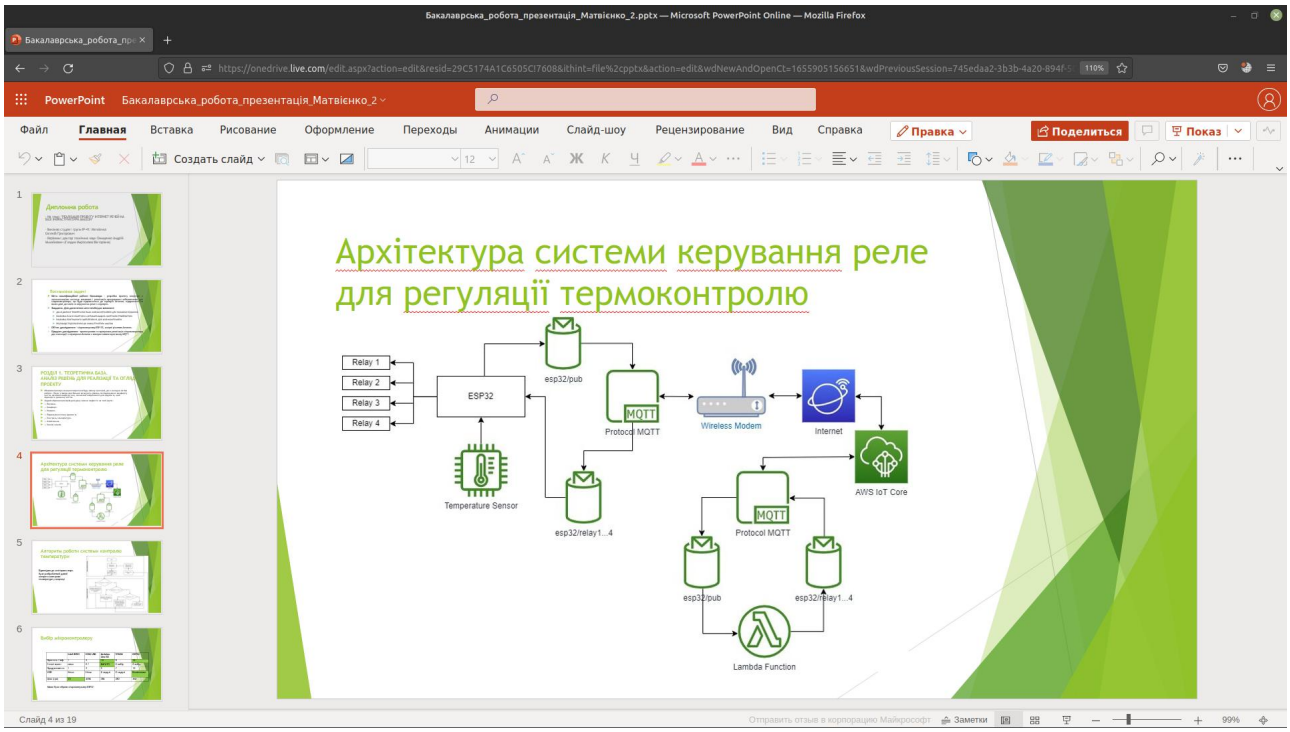
Євгеній МАТВІЄНКО

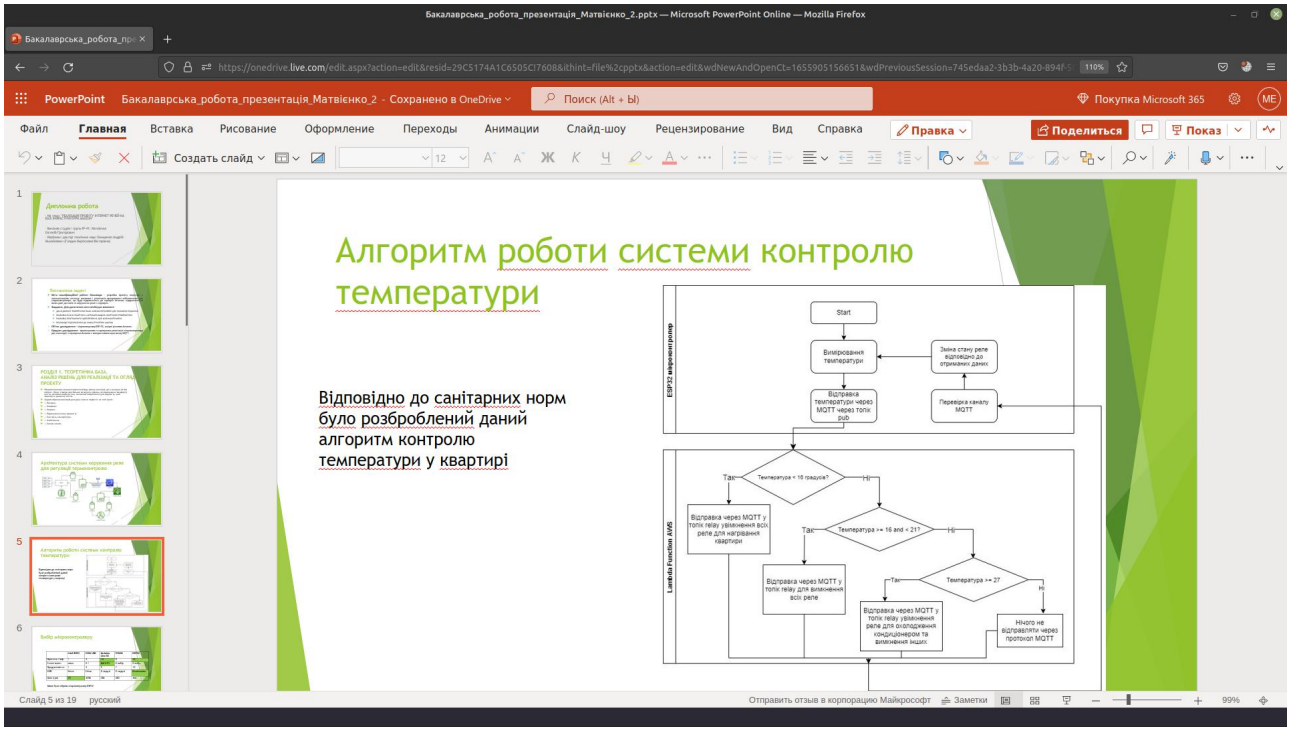
Керівник

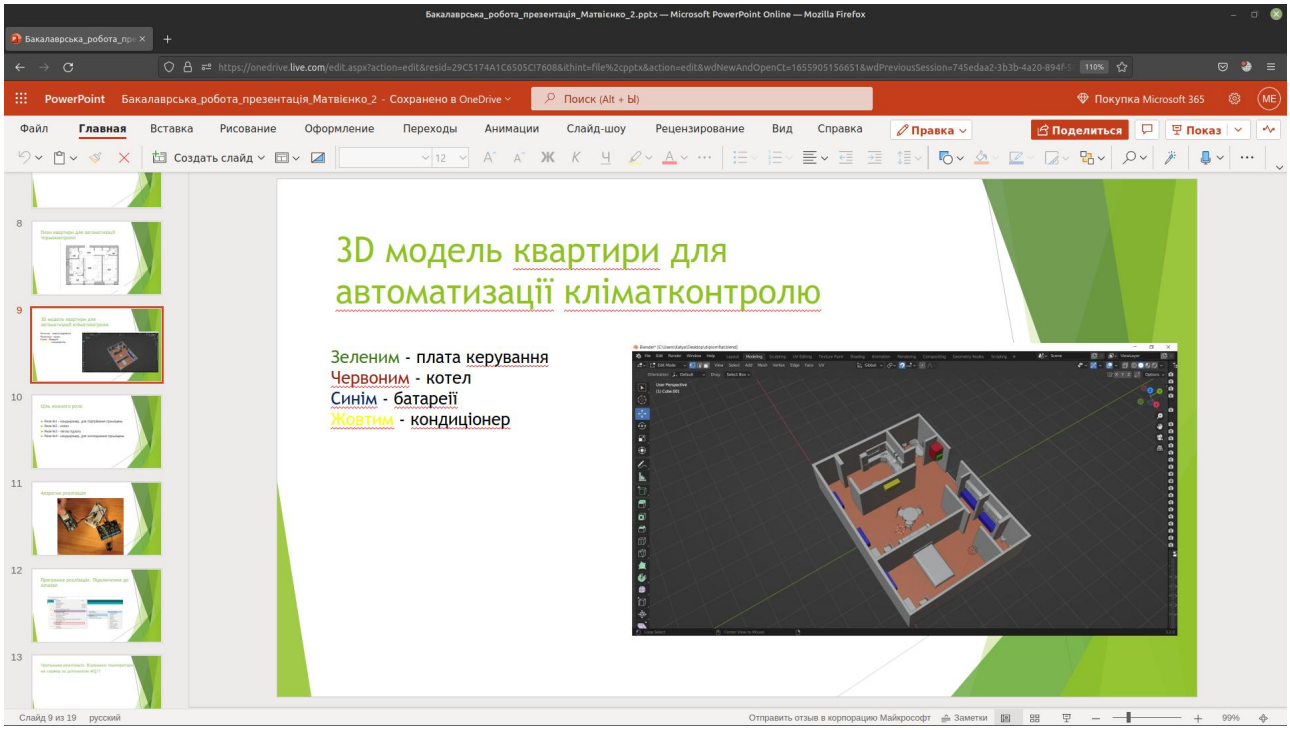
Мирослава ГЛАДКА

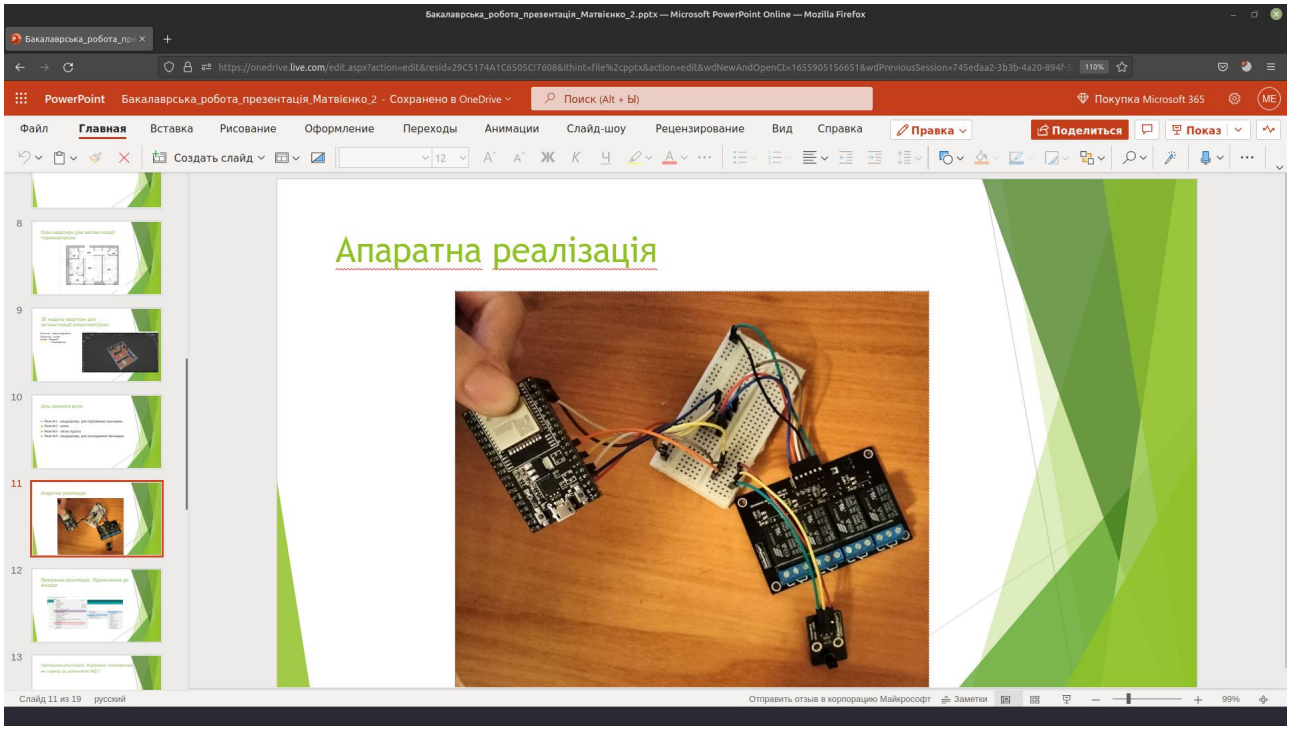
Київ – 2022

Основні слайди з презентації для захисту дипломної роботи






Бакалаврська_робота_презентація_Матвієнко_2.pptx — Microsoft PowerPoint Online — Mozilla Firefox
PowerPoint
Бакалаврська_робота_презентація_Матвієнко_2 - Сохранено в OneDrive
Поиск (Alt + Ы)
Покупка Microsoft 365
Файл
Главная
Вставка
Рисование
Оформление
Переходы
Анимации
Слайд-шоу
Рецензирование
Вид
Справка
Правка
Поделиться
Показ
Создать слайд
3D модель квартири для автоматизації кліматконтролю
Зеленим - плата керування
Червоним - котел
Синім - батареї
Жовтим - кондиціонер
Слайд 9 из 19
русский
Отправить отзыв в корпорацию Майкрософт
Заметки
99%


Бакалаврська_робота_презентація_Матвієнко_2.pptx — Microsoft PowerPoint Online — Mozilla Firefox
PowerPoint
Бакалаврська_робота_презентація_Матвієнко_2 - Сохранено в OneDrive
Поиск (Alt + Ы)
Покупка Microsoft 365
Файл
Главная
Вставка
Рисование
Оформление
Переходы
Анимации
Слайд-шоу
Рецензирование
Вид
Справка
Правка
Поделиться
Показ
Создать слайд
Апаратна реалізація
Слайд 11 из 19
русский
Отправить отзыв в корпорацию Майкрософт
Заметки
99%