МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет комп'ютерних наук, фізики та математики

Кафедра комп'ютерних наук та програмної інженерії

DEVELOPMENT OF THE SOFTWARE MODULE AND TOOLS FOR "SMART HOUSE" SYSTEMS.

Кваліфікаційна робота (проєкт)

на здобуття ступеня вищої освіти «бакалавр»

Виконав: здобувач 4 курсу 441 групи

Спеціальності 121 Інженерія програмного забезпечення

Освітньо-професійної (наукової) програми Інженерія програмного забезпечення

Дубіна Владислав Геннадійович

Керівник: кандидат педагогічних наук, доцент Вінник Максим Олександрович

Співкерівник: доцент кафедри комп'ютерних наук та програмної інженерії ХДУ Єрмолаєв Вадим Анатолійович

Рецензент: професор

кафедри фізики ХДУ Бабічев С.А.

Херсон – 2022

Вступ

Актуальність теми

Одним із перспективних та популярних на сьогоднішній день напрямів розвитку технологій є Інтернет речей (IoT), а саме Розумний дім (“Smart house”), так як інформаційні технології IoT користуються попитом серед споживачів. Завдяки стрімкому розвитку технологій, власники домів та квартир дедалі більше використовують останні розробки у сфері IoT для забезпечення надійності і безпеки приміщення, а також для збільшення комфорту проживання. Автоматизация систем або речей дозволяє витрачати менше часу, та зосередити цей час на більш важливі речі. Наприклад – увімкнення опалення оселі до прихода власника, це дозволяє економити електроенергію і час на підігрів домівки.

Мета і завдання

Об’єкт дослідження - технології, у тому числі IoT, для реалізації системи “Smart house”.

Предмет дослідження - програмні та апаратні засоби для створення і функціонування системи “Smart house”.

Мета дипломної роботи полягає у аналізі популярних IoT платформ та сучасних мікроконтролерів, виявлення кращої платформи та мікроконтролера і послідуючого їх аналізу для вияву недоліків. Спираючись на результати аналізу буде обрана платформа та мікроконтролер, які будуть використані як основа для розробки системи “Розумний будинок” для встановлення в навчальних аудиторіях.

Відповідно до мети можна визначети основні *завдання* роботи:

- Аналіз популярних IoT платформ та мікроконтролерів

- Розробка серверу для обміну та збереження інформації:
    - Розгортання та налаштування серверу.
    - Розгортання та налаштування бази даних.
    - Створення Dashboard для аналізу та відображення інформації отриманої від пристрою.
    - Розгортання та налаштування API для обміну даними.

- Розробка системи моніторингу даних о екологічному стані навчальних аудиторій:
    - Сборка інструменту на базі мікроконтролера ESP8266 для збору екологічної інформації.
    - Сборка LED дісплею на базі мікроконтролера ESP8266 об’єднаного з LED панелею P10 та.

- Модифікація університетських web сайтів

РОЗДІЛ 1.

Новітні технології в системах “Smart house”

1.1 Досвід у використанні систем “Smart house” в сучасному суспільстві

«Smart house» («Розумний будинок») – це житловий будинок сучасного типу, офіс, або інша споруда, організований для роботи або проживання людей за допомогою автоматизації і високотехнологічних пристроїв.
Під цим терміном слід розуміти систему, яка забезпечує безпеку та ресурсозбереження (в тому числі і комфорт) для всіх користувачів.
Зі збільшенням обчислювальної здатності пристроїв концепція «розумний будинок» отримала логічне продовження – це система «Інтернет речей», згідно з якою була проведена первинна стандартизація та визначені основні правила та рекомендації до побудови готового продукту на рівні як системи загалом, так і окремих компонентів. «Розумний будинок» повинен вміти розпізнавати конкретні ситуації, що відбуваються в будівлі, і відповідним чином на них реагувати. Одна з систем може управляти поведінкою інших по заздалегідь виробленим алгоритмам. Основною особливістю інтелектуальної будівлі є об'єднання окремих підсистем в єдиний керований комплекс. До складу такого комплексу входять різноманітні датчики, керуючі елементи та виконавчі пристрої. Головним завданням для системи «Smart house» є – забезпечення комфорту, економне використання енергоресурсів, а також високого рівня безпеки. На розвиток сучасного суспільства сильно впливають комп'ютерні технологій, що проникли в усі сфери людської діяльності, це сприяє утворенню глобального інформаційного простору. В сучасному світі будь яка будівля - адміністративна або житлова складається з набору численних підсистем, що відповідають за виконання деяких функцій, які домогають вирішувати

або вирішують різноманітні завдання процесу роботи будівлі. Догляд та управління такими системами стає все складнішим. Стрімке зростання витрат на ремонт, обслуговування або персонал для цих підсистем, наразі є найбільш актуально проблемою при використанні великих навчальних закладів, виробничих комплексів тощо. Розвиток систем «Smart house», в перспективі може поліпшити роботу, та скоротить витрати на обслуговування підсистем.

1.2 Огляд сучасних IoT платформ

1.2.1 Платформа ThingsBoard

ThingsBoard - це IoT платформа яка надає можливості для збору, обробки, візуалізації та керування пристроями. Платформа підтримує зв'язок з пристроями за стандартними протоколами IoT - MQTT, CoAP і HTTP, також платформа має можливості хмарного або локального розгортання. Відмовостійкість, масштабованість та продуктивність це то що поєднує в собі ThingsBoard. Платформа дозволяє налаштувати інформаційні панелі як забажає користувач та навіть створити додаткові модулі, використовуючи зручний інтерфейс розробки. Кожна інформаційна панель може містити в собі декілька віджетів, які відображають дані з підключенних до неї пристройв. Кожній інформаційній панелі можна призначити свого користувача. Інформаційні панелі мають невелику вагу, це надає можливість мати велику кількість панелей наповнених різноманітною інформацією. [6, 7, 8]

1.2.2 Платформа ThingSpeak

ThingSpeak - платформа з відкритим кодом, яка основана на мові програмування Ruby, що дозволяє користувачам та їх пристроям спілкуватися за підтримки мережі Інтернет. Платформа також надає можливості реєстрації та пошук даних, надаючи власний API Інтегрованя MATLAB від MathWorks до ThingSpeak дозволяє користувачам аналізувати та візуалізувати дані за допомогою MATLAB, без ліцензії MATLAB у MathWorks. [9, 10]

1.2.3 Платформа AWS IoT Core

AWS IoT Core – ця платформа надає можливість пристроям спілкуватися один з одним або з хмарними додатками AWS. Також надає інструментарій, необхідний для оброблення, використання та керування даних, що надходять з пристроїв. Має можливість розгортання власних хмарних додатків, які можуть спілкуються з пристроями та керувати ними. AWS IoT Core використовує різні протоколи зв'язку: MQTT, MQTT. За допомогою транспортного рівня безпеки (TLS) версії 1.2 і наскрізного шифрування, платформазахищає комунікації. [11, 12]

1.2.4 Платформа UnaConnect

UnaConnect - дозволяє розгорнути, інтегрувати та підтримувати велику кількість пристроїв у системі. За допомогою цієї платформи зменшужться технічний тягар керування пристроями, що допомагає зосередитися на важливих для бізнесу деталях. Будьякі повідомлення декодуються, приймаються та зберігаються в хмарному рішенні UnaConnect.

Дешифровані повідомлення можуть бути надіслані на будь-яку комерційну інформаційну панель IoT або в JSONна інщі системи. [13]

1.2.5 Платформа Google Cloud IoT

Google Cloud призначена для розробки IoT на основі хмарних платформ Google. Платформа дозволяє обробляти та отримувати дані з пристрою.Для реєстрації, моніторингу та налаштування пристроїв використовуються диспетчера пристроїв. Для підключення пристрою та зв'язку платформа використовує протоколи MQTT і HTTP. Прийом даних та маршрутизацію повідомлень виконується за допомогою Cloud Pub/Sub.

Безпечну аналітику даних забезпечує Google BigQuery.

Google Data Studio надає можливість візуалізації дані, шляхом створення звітів або інформаційних панелей. [14, 15, 16]

1.2.6 Платформа Cisco IoT Cloud

Cisco IoT Cloud Connect - інструмент для мобільних операторів. Cisco надає надійне обладнання для IoT таке як: точки доступу, комутатори, маршрутизатори, шлюзи тощо.

Cisco IoT Control Center забезпечує управління стільниковим підключенням, також дозволяюче інтегрувати ваші пристрої IoT до одного рішення SaaS.

Розробка бізнес-додатків проводиться за допомогою розширеного корпоративного рішення яке забезпечує централізоване керування мережею та швидке розгортання.

Edge Intelligence використовується для обробки даних, за допомогою розділення потоків даних локального або багато хмарного середовища.

Захистом від кібератак, а також забезпечення віддаленого доступу, та інші послуги безпеки виконує Cisco IoT Threat Defense. [17, 18]

1.2.7 Платформа Oracle IoT

Хмарна служба IoT Oracle - платформа послуга (PaaS) для підключення пристроїв до хмари. [19]

Основні функції :

- Створення програм та підключати їх до пристроїв із JavaScript, Java, Android, iOS, C POSIX та REST API
- Інтеграція з корпоративними програмами, сервісами або іншими службами Oracle
- Інструменти аналізу для зниження, за допомогою фільтрації, вхідних даних
- Автоматична синхронізація даних з Oracle Business Intelligence Cloud Service

1.2.8 Платформа ThingWorx

ThingWorx - платформа IoT, яка використовується для виробництва, обслуговування та інженерії. Платформа вирішує проблеми в різних галузях, від моніторингу та обслуговування до оптимізації активів.

Основні функції ThingWorx:

- Доступ до джерел даних завдяки поширенню промислових комунікацій
- Потужні інструменти, програми для швидкого створення рішень IoT

- Статистика зі складних промислових даних IoT для активної оптимізації операцій та запобігання проблемам, в режимі реального часу
- Повний контроль над пристроями, процесами та системами

Основні приклади використання:

- Моніторинг активів
- Дистанційне обслуговування

1.3 Огляд сучасних мікроконтролерів

1.3.1 Мікроконтролер ESP8266

ESP8266 - мікроконтролер китайської фірми виробника Espressif Systems, обладнаний Wi-Fi інтерфейсом, а також пам'яті на кристалі не має енергонезалежності. Програми виконаннуються за допомогою зовнішнього SPI ПЗУ, шляхом динамічного завантаження необхідних розділів програми до інструкцій кешу. Завантаження є апаратним і прозорим для програміста. Мікроконтролер підтримує до 16 МБ зовнішньої пам'яті. Можливий стандартний, подвійний або чотириразовий інтерфейс SPI.

Виробник не надає документацію на внутрішню периферію мікроконтролера. Замість цього він надає набір бібліотек і API, які дозволяють програмісту отримати доступ до периферійних пристроїв. Оскільки ці бібліотеки використовують оперативну пам’ять дуже

інтенсивно, виробник не вказує в документах точний обсяг оперативної пам’яті на мікросхемі, а лише приблизну оцінку обсягу оперативної пам’яті, що залишається у користувача після підключення бібліотек – близько 50 Кбайт. Ентузіасти, які досліджують бібліотеку ESP8266, припускають, що вона містить 32 КБ кешу інструкцій і 80 КБ даних RAM. [1, 2, 3, 8]

1.3.2 Мікроконтролер ARDUINO UNO

ARDUINO UNO - це звичайна плата мікроконтролера з відкритим вихідним кодом на основі мікроконтролера ATmega328P. Він включає в себе все необхідне для роботи з мікроконтролером: 14 цифрових входів/виходів (6 з них можна використовувати як ШІМ), 6 аналогових входів, кварцовий резонатор 16 МГц, роз'єм USB, роз'єм живлення, роз'єм внутрішнього програмування (ICSP) і кнопка скидання. Щоб почати, просто підключіть адаптер змінного / постійного струму, акумулятор або підключіть його до комп’ютера за допомогою USB-кабелю.
На відміну від усіх попередніх плат Arduino, Uno використовує мікроконтролер ATmega16U2 як перетворювач інтерфейсу USB-UART замість мікросхеми FTDI. [4, 5]

1.4 Аналіз популярних IoT платформ та мікроконтролерів

Виходячи з усього вищесказаного, можна підсумувати вибір системи IoT і мікроконтролера, найкраще це буде видно в табличній версії. [20, 21]

Таблиця 1

Порівняння IoT платформ

Назва	Наявність безкоштовної версії	Протоколи обміну даними	Інструменти для розробки додаткових функцій
ThingsBoard IoT Platform	+	MQTT, CoAP і HTTP	+
Thingspeak IoT Platform	+	MQTT і HTTP	+
AWS IoT Core	-	MQTT, CoAP і HTTP	-
UnaConnect IoT Platform	-	MQTT і HTTP	-
Google Cloud IoT	-	CoAP і HTTP	+
Cisco IoT Cloud	-	MQTT і HTTP	-
Oracle IoT	-	MQTT і HTTP	-
ThingWorx	-	MQTT і HTTP	-

Таблиця 2

Порівняння мікроконтролерів

Назва	Робоча напруга	WIFI модуль	Інтерфейс	Flash-пам'ять
ESP 8266	2,2…3,6 В.	+	SPI	4*16 МБайт
ARDUINO UNO R3 ATMEGA328	5В	-	ШІМ	32 КБ

Наведені вище таблиці (таблиці 1, 2) показують особливості вибраних систем IoT та мікроконтролерів. Тепер настав час вибрати одну систему IoT. Для нас дуже важливо мати можливість додавати будь-які функції та віджети в нашу систему та різноманітні протоколи зв'язку. Виходячи з результатів нашого аналізу, представлених у таблиці, видно, що дві системи відповідають нашим критеріям. Такими системами є: **ThingsBoard** і **AWS**. Тепер розглянемо інші характеристики цих засобів. AWS є повністю комерційною системою, тобто для її використання потрібні гроші. У свою чергу, **ThingsBoard** надає скорочену версію, або повний доступ до коду безкоштовно. Що стосується мікросхем, то нам потрібно відразу встановити модуль WIFI, пам'яті вистачить. Це рішення для нас - **ESP8266.**

Виходячи з усього вищесказаного, ми вибираємо **ThingsBoard** та **ESP8266**.

РОЗДІЛ 2.

Розробка інструментів та програмних засобів для системи “Smart house”

У процесі розробки програмного забезпечення є значний обсяг розробки проекту. Тому для стандартизації процесу розробки потрібна документація. Дизайн високого рівня може бути способом отримати розуміння того, як буде реалізовано загальне рішення.

Високорівневий дизайн може забезпечити основу, щоб краще зрозуміти цю функцію. Ця документація може керувати детальним проектом, щоб відповідати загальним рішенням. Для створення високорівневого дизайну існує багато підходів. Найпростіший спосіб зробити це — створити діаграму для представлення рішення. Документ не міститиме повного пояснення, наприклад, який фреймворк використовувати. Але це більше схоже на загальний потік того, як буде відбуватися процес і яке технічне рішення нам потрібно для виконання вимоги. Перше, що потрібно зробити, це сформулювати вимоги до проекта. В нашому випадку такими вимогами є:

- Отримання детальної інформації о навколишньому середовищі навчальних аудиторій.
- Обмін даними за допомогою протоколів MQTT, CoAP і HTTP.
- Гнучка система оповіщення користувачів в разі виникнення небезпечних ситуацій.
- Відображення інформації в реальному часі, на веб сайті, а також на інформаційних LED панелях

Після того, як ви зрозуміли вимоги, настав час створити дизайн високого рівня. Спочатку ми створюємо (системну) діаграму для розробки нашого рішення. У нашому проекті ми визначимо варіант використання як сервіс. Ми створимо зв’язок між частинами сервісу, щоб створити загальне розуміння системи. Цю діаграму, яка представляє систему ми можемо побачити на рис. 2.1

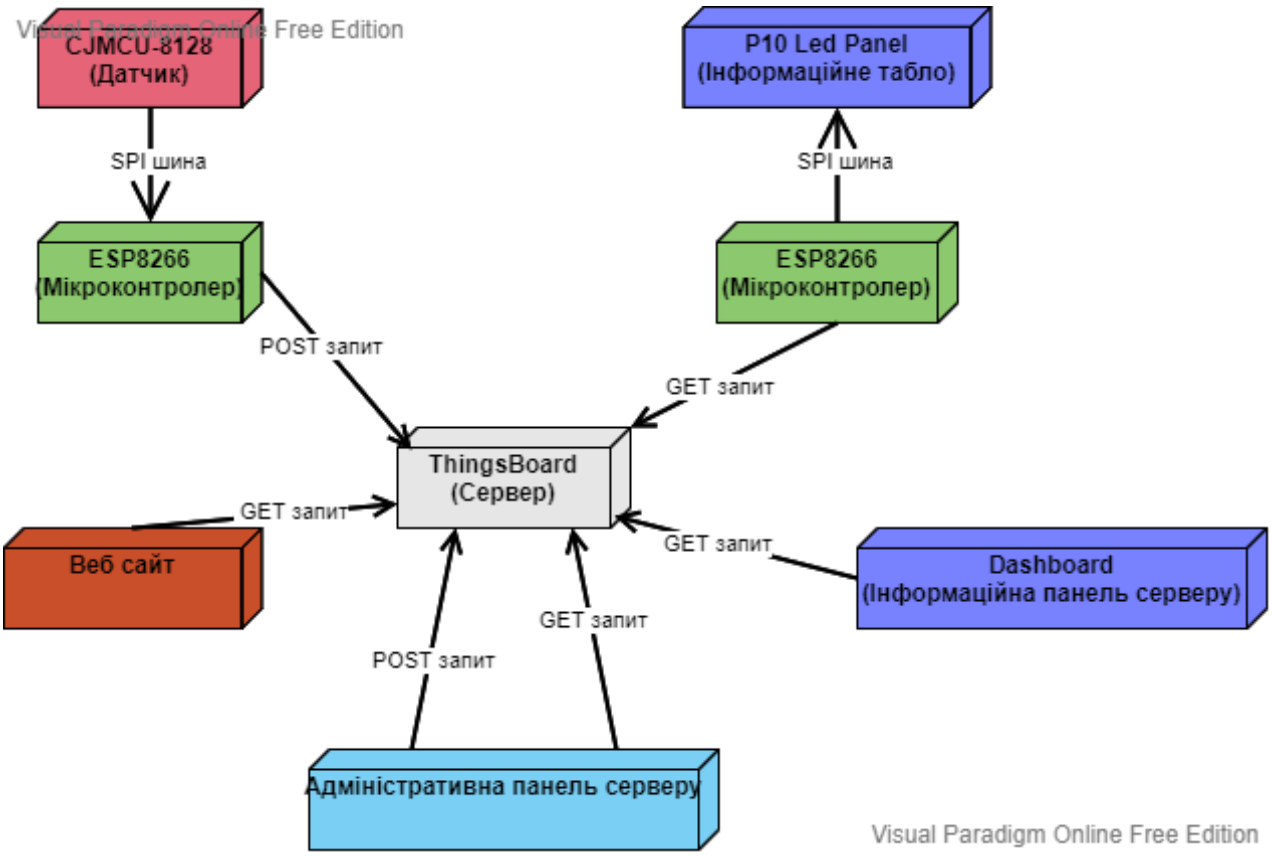
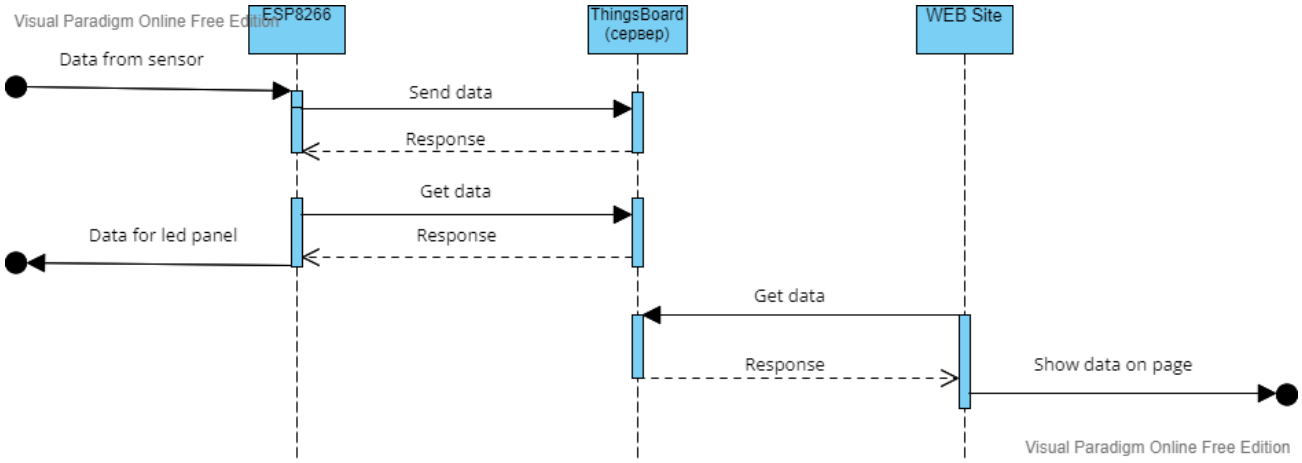


Рис 2.1 Схема системи

Основними об’єктами на рис 2.1є розгалуджені модулі. Наприклад мікроконтролер EXP8266 та датчик CJMCU-8128 , які обьєднані SPI шиною для передачі інформації від датчика до мікроконтролера, який в свою чергу зробить первинну обробку інформації, тобто конвертація данних в необхідний формат, сгенерую JSON файл з інформацією о навколишньому середовищі та відправить його до серверу ThingsBoard. Так я к сервер не може відправляти POST запити до наших мікроконтролерів , другий ESP8266 який підключений до P10 Led Panel кожні 30 секунд виконує GET запит до серверу щоб отримати актуальну інформацію та відобразити її на P10 Led Panel.

ThingsBoard сервер обробляє данні часових послідовності, тобто ті данні які прийшли з певного приладу, та встановлює їм часову мітку, яка буде використанна на Dashboard для створення різноманітних графіків.

Кожен с модулів спілкується з сервером за допомогою REST API, цей формат був обран тому що ми не передаєму дуже велику кількість інформації , й тому



не потебуємо іншого формату обміну данних, хоча сервер може підтримувати обмін за допомогою MQTT та CoAP. Більш детальніше взаємодія модулів показана на рис. 2.2

Рис. 2.2 Діаграма послідовності

По друге важливо встановити які дії можуть виконувати користувачі, та адміністратори. В нашому випадку в користувача залишається невелика кількість дії, а саме:

- Авторизація / Реєстрація
- Налаштування власного облікового запису
- Перегляд доступної інформації на Dashboard

В свою чергу адміністратор може виконувати такі дії як і користувач, а також додаткові дії відповідно до його ролі :

- Створення або редагування облікових записів для користувачів та пристроїв
- Налаштування доступу до інформаційних панелей
- Налаштування серверу

Ці залежності зображені на рис. 2.3

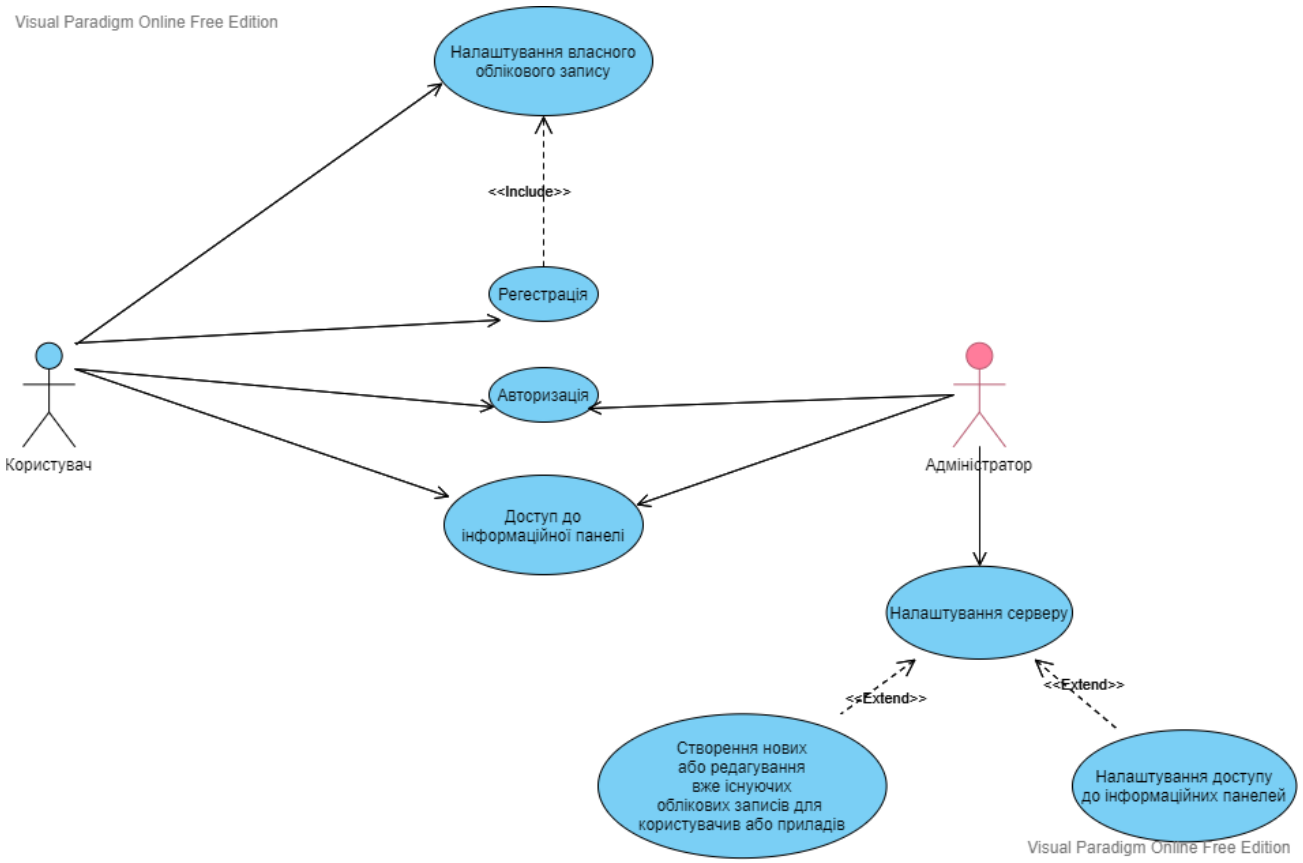


Рис. 2.3 Ролі та їх дії

Зробивши схеми та маючи базову документацію про те, як буде виглядати наша система. Ми можемо отримати більш широке уявлення про функції та почати їх реалізацію.

2.1 Запити до сервера

Спираючись на hld сервіса, яка була описана в розділі 2 , для передачи данних мі використовуємо JSON файли з певною сигнатурою. В нашому випадку ця сигнатура не дуже складна й заключається с основному ключів та самої інформації, що ми можемо побачити на лістингу 1.

```
{       “id”: 509,
        “temperature”: 25.3,
        “humidity”: 62,
        “co2”: 550,
        “polution”: 123,5
}
```

Лістинг 1

Як ми можемо бачити на лістингу 1, наш json файл містить в собі інформацію о навколишньом середовищі тієї чи іншої аудиторії, та номер аудиторії до якої відносятся ці данні.

2.2 Мікроконтролер ESP8266

2.2.1 Призначення ESP8266

ESP8266 - мікроконтролер, призначений для керування електронними пристроями. У нашому випадку це такі пристрої: модуль CJMCU-8128 для збору екологічних даних і світлодіодна панель P10 для відображення інформації в режимі реального часу. Вбудований модуль WIFI дозволяє легко підключатися до будь-якої існуючої мережі WIFI за умови наявності ідентифікатора та пароля цієї мережі, що є перевагою перед іншими подібними мікроконтролерами.

ESP8266 виконує первинну обробку інформації залежно від закодованої програми. Прикладом цього є:

- Перетворення температури з градусів Кельвіна в Цельсій.
- Побудова файлу JSON з даних, отриманих від модуля CJMCU-8128
- Запити на сервер, з певним інтервалом, для оновлення інформації на світлодіодній панелі P10.

2.2.2 CJMCU-8128 модуль для збору екологічних даних

Модуль CJMCU-8128 на основі мікросхеми газового датчика (SSoC) і датчика температури і вологості. Використовує технологію малопотужного зондування для виявлення летких органічних сполук (ЛОС).

Схематична структура плати CJMCU8118 показує, що це не простий датчик, він інтегрований з MCU, тобто його можна обробляти на борту

без втручання господаря, і може надавати інформацію про еквівалентні рівні вуглекислого газу або загальні леткі органічні сполуки (TVOC).

2.2.3 P10 LED модуль для відображення інформації в режимі реального часу

Світлодіодні дисплеї P10 дуже прості в установці та використанні, вони дуже надійні та довговічні. Головною перевагою світлодіодних екранів є їх висока яскравість, стійкість до різних погодних умов, можливість використання вдень і вночі.

2.3. ThingsBoard IoT Platform як інструмент для зберігання, аналізу та відображення даних, отриманих від пристроїв

2.3.1 Призначення ThingsBoard

ThingsBoard — це платформа IoT для збору, обробки, візуалізації та керування пристроями. Платформа підтримує підключення пристроїв за допомогою стандартних протоколів IoT - MQTT, CoAP і HTTP, а також підтримує хмарне і локальне розгортання. ThingsBoard поєднує масштабованість, відмовостійкість і продуктивність, тому ви не втратите свої дані.

2.3.2 Основні сутності ThingsBoard

ThingsBoard надає користувальницький інтерфейс та REST API для керування кількома типами сутностей та їхніми відносинами в Інтернеті речей.

Підтримувані сутності:

- **Орендарі (Tenants)**
- **Замовники (Customers)**

- **Користувачі (Users)**
- **Пристрої (Devices)**
- **Активи (Assets)**
- **Entity Views**
- **Сигналізація (Alarms)**
- **Інформаційні панелі (Dashboards)**
- **Вузол правила (Rule Node)**
- **Ланцюжок правил (Rule Chain)**

2.4 Розробка серверу за допомогою ThingsBoard

2.4.1 Параметри конфігурації сервера

Розгортання платформи ThingsBoard на вашому пристрої виконується за допомогою Docker.

Docker -це програмне забезпечення для автоматизації розгортання та керування додатками в контейнерних середовищах. Дозволяє «упакувати» програму з усім її оточенням і залежностями в контейнер, який можна розгорнути в будь-якій системі Linux з підтримкою cgroups в ядрі, і надає набір команд для керування цими контейнерами.

Перш за все, яку базу даних ми будемо використовувати, оскільки ThingsBoard підтримує три типи зображень з одним екземпляром, наприклад:

- tb-postgres
- tb-cassandra
- tb

Виходячи з того, що ми будемо використовувати PostgreSQL, то наш вибір впав на «tb-postgres»

Виходячи з усього вищесказаного, нам потрібно створити файл конфігурації з розширенням yml з конфігурацією нашого сервера, заповнений файл, показаний на рис. 2.4.1.1.

```yaml
version: '2.2'
services:
  mytb:
    restart: always
    image: "thingsboard/tb-postgres"
    ports:
      - "8080:9090"
      - "1883:1883"
      - "7070:7070"
      - "5683-5688:5683-5688/udp"
    environment:
      TB_QUEUE_TYPE: in-memory
    volumes:
      - ~/.mytb-data:/data
      - ~/.mytb-logs:/var/log/thingsboard
```

Рис 2.4.1.1 Приклад конфігурації docker-compose.yml

2.4.2 Налаштувати моделі пристроїв для збереження об’єктів у базі даних та коректної роботи API

Налаштувати моделі пристроїв за допомогою ThingsBoard дуже зручно, оскільки ThingsBoard надає простий і зручний інтерфейс.

Кроки для створення моделі:

- У екземплярі ThingsBoard необхідно відкрити сторінку “Пристрої” (рис. 2.4.2.1).

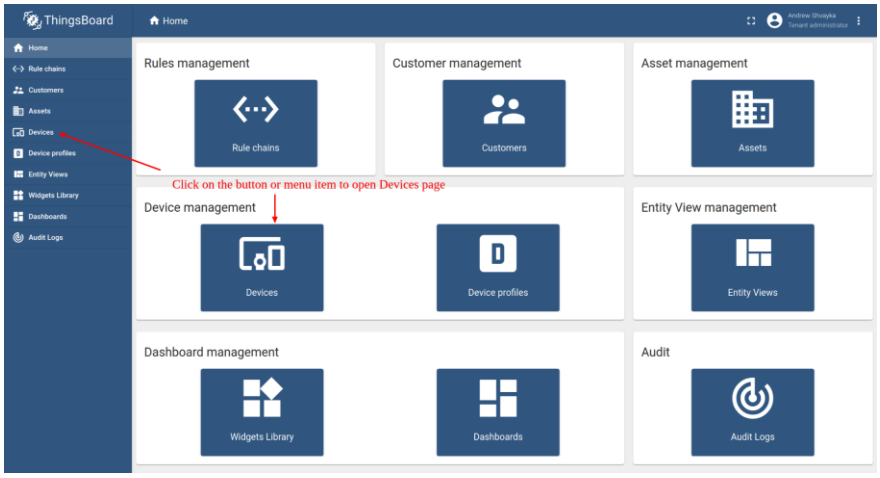


Рис. 2.4.2.1 Наглядна інструкція 1

- Натисніть на «+» у верхньому правому куті таблиці та обрати «Додати новий пристрій» (рис. 2.4.2.2).

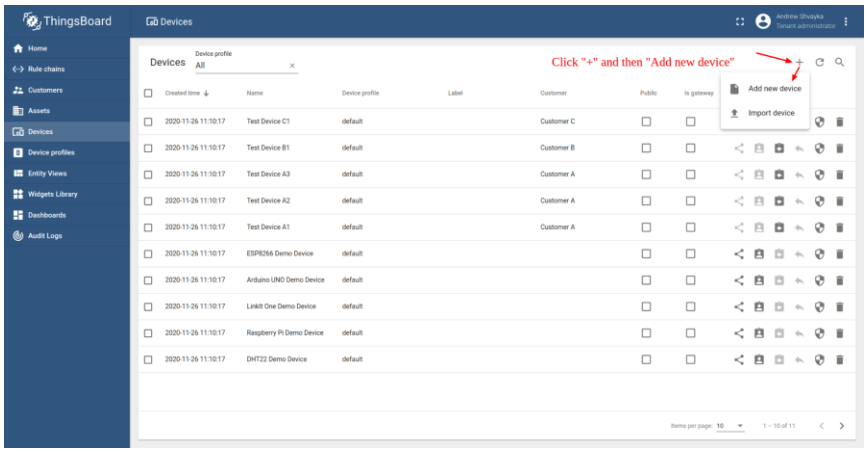


Рис. 2.4.2.2 Наглядна інструкція 2

- Введіть назву пристрою. Наприклад, «Мій новий пристрій». Інших змін зараз не потрібно. Натисніть «Додати», щоб додати пристрій (рис. 2.4.2.3).

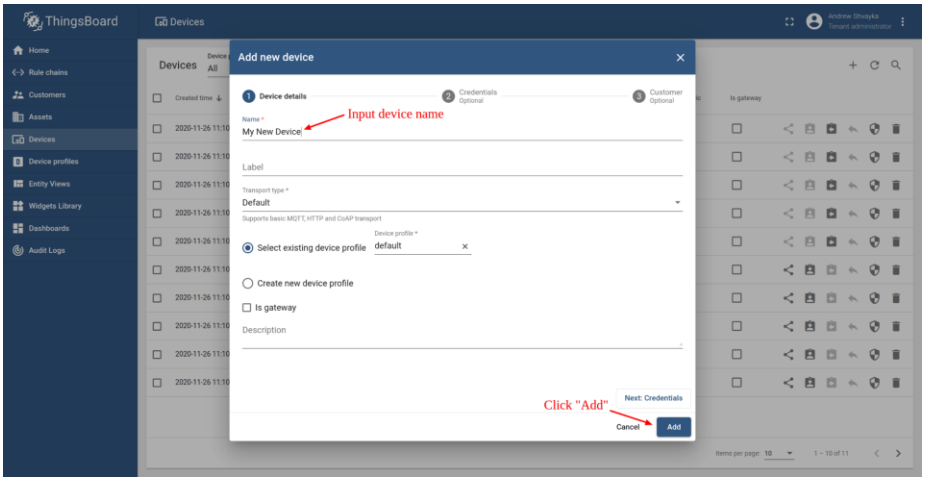


Рис. 2.4.2.3 Наглядна інструкція 3

- Тепер пристрій має бути першим у списку, оскільки таблиця за замовчуванням сортує пристрої за часом створення за (рис. 2.4.2.4).

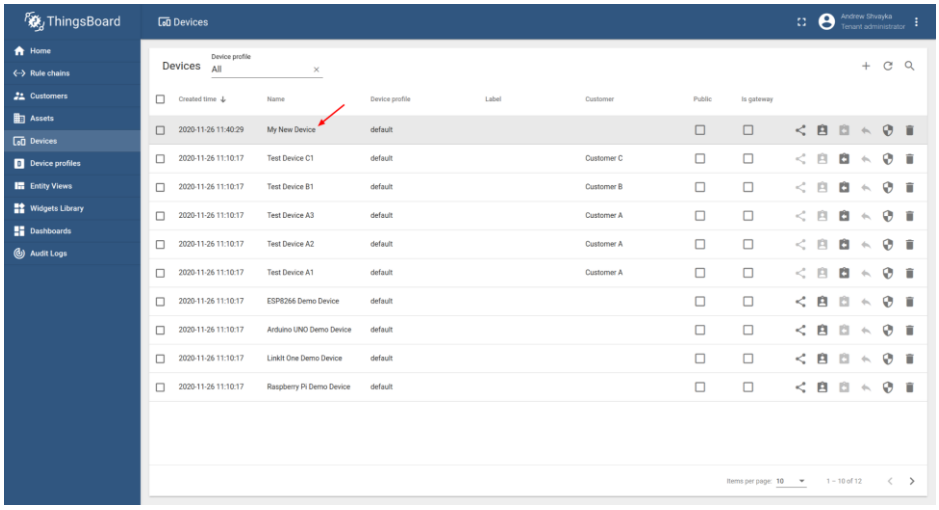


Рис. 2.4.2.4 Наглядна інструкція 4

2.4.3 Налаштування Dashboard для відображення аналітики отриманої інформації

ThingsBoard надає гнучкий інструмент для створення інформаційних панелей, стандартних віджетів і можливість створювати власні або імпортування віджета у форматі JSON. Щоб відобразити зміни температури та вологості, найкраще використовувати діаграму, яка дозволяє оцінити зміну продуктивності з часом. Вам також потрібно буде додати віджет до таблиці, який показує вашу поточну ефективність або деякі результати вашої екологічної оцінки.

Щоб створити типову таблицю, потрібно виконати деякі дії, а саме:

- Увійдіть в режим редагування, натисніть кнопку «Додати новий віджет».
- Виберіть набір віджетів «Карти», виберіть вкладку «Останні значення» та клацніть заголовок віджета «Об'єкти».
- Натиснути «Додати», щоб додати джерело даних. Віджет може мати кілька джерел даних.
- Виберіть псевдонім об'єкта. Клацніть поле введення праворуч. З’явиться автозаповнення доступними точками даних. Виберіть потрібну точку даних і натисніть «Додати».

Після того, як таблиця буде сформована, вона буде виглядати так само як показано на рис. 2.4.3.1

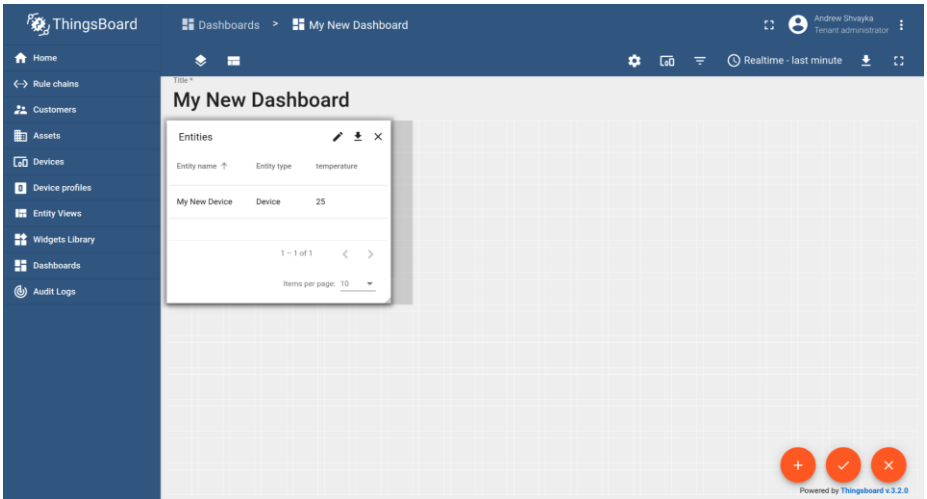


Рис.2.4.3.1 Вигляд типового віджету таблиці

Існує також кілька кроків, які ви можете зробити, щоб створити діаграму:

- Увійти до режима редагування.
- Натиснути «Додати новий віджет» у нижньому правому куті екрана.
- Натиснути «Створити новий віджет».
- Виберіть Діаграми. Натисніть на віджет графіка "Тимчасові рядки".
- Натисніть кнопку Додати джерело даних.
- Виберіть псевдонім MyDevice. Виберіть «температура». Натисніть «Додати».
- Перетягніть віджет у потрібне місце. Змініть розмір віджета. Застосувати зміни.

Віджет діаграми за замовчуванням виглядає як стандартний (рис. 2.4.3.2, рис. 2.4.3.3), але ThingsBoard дозволяє налаштовувати та створювати нові віджети.

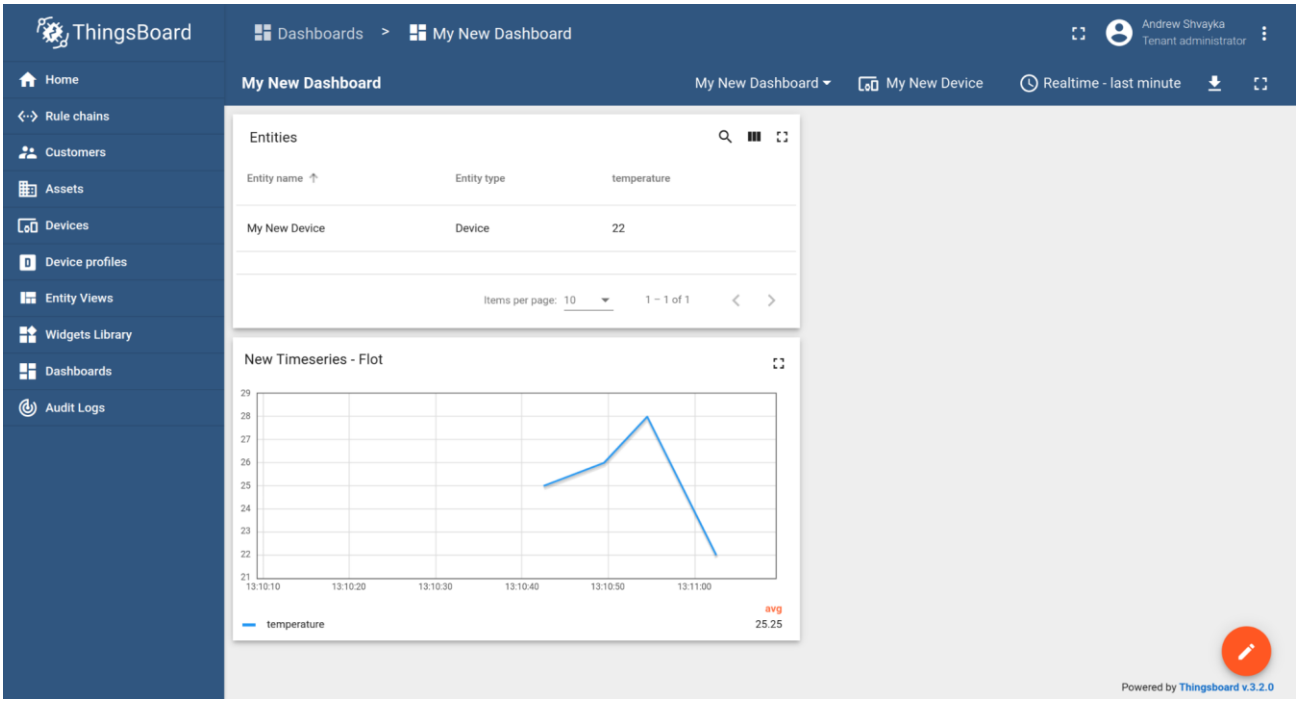


Рис. 2.4.3.2 Вигляд типового віджету графіку 1

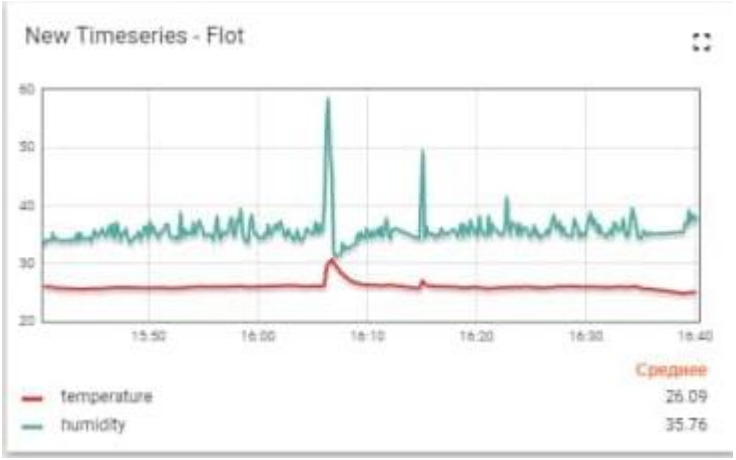


Рис. 2.4.3.3 Вигляд типового віджету графіку 2

2.5. Розробка програмного забезпечення для мікроконтролера ESP8266

Однією з частин нашого дослідження є мікроконтролер ESP8266, а також додаткові до нього модулі, а саме:

- CJMCU-8128
- P10 LED Panel

З початку потрібно налаштувати мікроконтролера до WIFI мережі, для цього потрібно в блоці “setup” прописати наступні команди, що можно побачити в лістингу 1.

```
Serial.begin(115200);
Wire.begin();//Serial connection
WiFi.begin("WKSU", "wksu2019");
```

Лістинг 1

Наступним кроком є ініціалізація підключених датчиків з модуля CJMCU-8128, які ми бачимо на рис. 2.5.1

```
Serial.println("CCS811 test");
if (!ccs811.begin()) {
  Serial.println("Failed to start sensor! Please check your wiring.");
  while (true);
}

Serial.println("BMP280 test");     /* --- SETUP BMP on 0x76 ------ */
if (!bmp280.begin(0x76)) {
  Serial.println("Could not find a valid BMP280 sensor, check wiring!");
  while (true);
}

Serial.println("Si7021 test!");     /* ---- SETUP SI702x ----- */
if (!SI702x.begin()) {
  Serial.println("Did not find Si702x sensor!");
  while (true);
}
Serial.print("Found model ");
switch (SI702x.getModel()) {
  case SI_Engineering_Samples:
    Serial.print("SI engineering samples"); break;
  case SI_7013:
    Serial.print("Si7013"); break;
  case SI_7020:
    Serial.print("Si7020"); break;
  case SI_7021:
    Serial.print("Si7021"); break;
  case SI_UNKNOWN:
  default:
    Serial.print("Unknown");
}
Serial.print(" Revision(");
Serial.print(SI702x.getRevision());
Serial.print(")");
Serial.print(" Serial #"); Serial.print(SI702x.sernum_a, HEX); Serial.println(SI702x.sernum_b, HEX);
```

Рис 2.5.1 Ініціалізація модулів

Після підключення до модулів WIFI та ініціалізації все, що вам потрібно зробити, це отримати данні, обробити та відправити їх на сервер. Зчитування даних показано в листингу 2.

```
eco2=ccs811.geteCO2();
etvoc=ccs811.getTVOC();
temperature =bmp280.readTemperature();
humidity = SI702x.readHumidity();
pressure = bmp280.readPressure() / 100;
```

Лістинг 2

Щоб передати дані на сервер, спочатку потрібно створити файл JSON і заповнити його даними, як показано в листингу 3.

```
StaticJsonDocument<1000> postData;
postData["humidity"]=humidity;
postData["temperature"]=temperature;
postData["pressure"]=pressure;
postData["co2"]=eco2;
postData["tvoc"]=etvoc;
```

Лістинг 3

Після створення файлу JSON все, що вам потрібно зробити, це створити HTTPClient і додати необхідний заголовок і сервер адрес, як показано в листингу 4.

```
String address=
"http://test1.kspu.edu/api/v1/"+device_token+"/telemetry/";
HTTPClient http;
http.begin(address);
http.addHeader("Content-Type", "application/json");
auto httpCode = http.POST(postData.as<String>());
String payload = http.getString();
http.end();
```

Лістинг 4

P10 LED Panel підключено до аналогічного мікроконтролера ESP8266, тобто аналогічного підключення до мережі WIFI, як у листингу 1. Щоб P10 LED Panel могла відображати текст, написаний кирилицею, нам потрібно було розробити хеш-таблицю для символів кирилиці, кожен символ має розмірність 5/7 пікселів або діодів (як на світлодіодній панелі P10). Світлодіодна панель P10 використовує з'єднання SPI і контролер DMD2. Перш за все, нам потрібно ініціалізувати контролер і створити текстове поле, як показано в листингу 5, ми повинні зазначити, що аргументи передають кількість панелей по ширині і висоті, тому що P10 підтримує можливість об'єднання панелей в одну велику панель.

```
SPIDMD  dmd(2,1);

DMD_TextBox box(dmd,2,4);
```

Лістинг 5

Оскільки світлодіодна панель P10 має вбудовану стандартну систему передачі тексту, нам потрібно лише розділити текст на масив символів і передати кожен символ функції малювання, як показано в листингу 6, тоді світлодіодна панель P10 зробить усе сама.

```
const char *next = MESSAGE;
 while(*next) {
  box.print(*next);
  delay(420);
  next++;
 }
```

Лістинг 6

2.6. Модифікація університетських web сайтів.

Згідно з останнім з основних завдань роботи, які були описані в вступній частині, ми повинні провести модифікацію університетських web сайтів, а зокрема сайту "*Аналітична система матеріально-технічної бази Херсонського Державного Університету*". Так як особливістю сайту є детальне відображення плану кожного поверху університету, як показано на рис. 2.6.1. Наша модифікація буде заключатися в додаванні спливаючого модального вікна, для кожної аудиторії, яке в свою чергу буду відображати стан навколишнього середовища в аудиторії (температуру, вологість, забруднення повітря чадним газом та іншими сполуками).

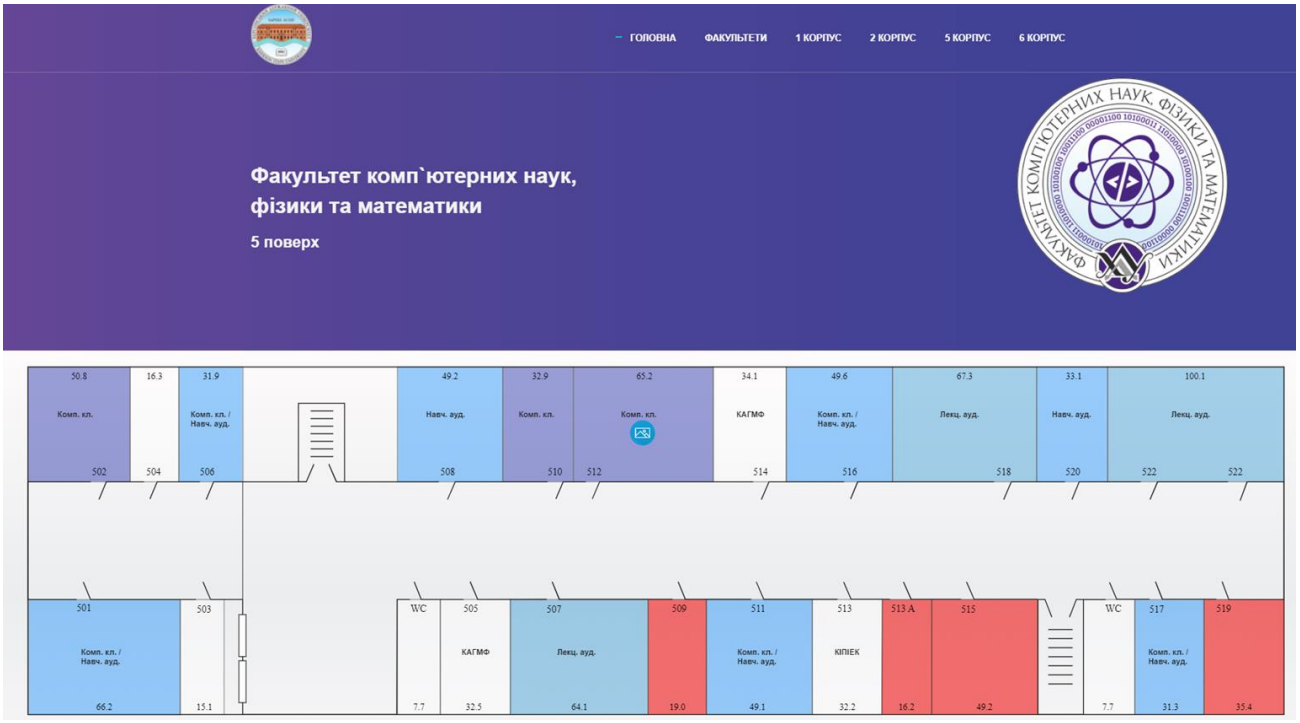


Рис. 2.6.1 Типовий вигляд сторінки сайту

Так як сайт розроблено за допомогою JS [22,23,24] , HTML [25,26,27] та CSS [28,29,30] ми будемо використовувати їх засоби для виконання поставленого завдання. Для початку додаємо до тегу, який відповідає за аудиторію, аргумент *data-toggle="modal"* як це показано на лістингу 1.

```
<g>
        <rect x="986" y="379.7" class="stv" width="89" height="175.6"
                id="d9" data-toggle="modal"/>
        <text transform="matrix(1 0 0 1 1022.2258 398.8375)"
                class="st5">509 </text>
</g>
```

Лістинг 1

Додавання аргументу *data-toggle="modal"* до тегу, необхідно щоб в надалі була можливість відслідковувати взаємодію з цим елементом, що нам знадобиться щоб отримувати інформацію о елементі, а саме ID аудиторії, тобто її номер, що в свою чергу необхідно для отримання інформації з серверу. Далі ми створюємо заготовку модального вікна, як показано на рис. 2.6.2.

```
<div id="tempModal" class="modal" >
    <div class=" modal-dialog modal-lg">
    <div class="modal-content">
      <div class="modal-header">
        <span class="close">&times;</span>
        <h2 style="font-size: 18px; text-align: center;" id="roomNumber"></h2>
      </div>
      <div class="modal-body">
                <div class="item active">
                    <table>
                        <tr>
                            <td>
                                <img width="25" height="25" src="img/temp.png">
                            </td>
                            <td style="padding-left:10px">
                                <p id="TempField" style="color:□#000000"></p>
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <img width="25" height="25" src="img/hum1.png">
                            </td>
                            <td style="padding-left:10px">
                                <p id="HumField" style="color:□#000000"></p>
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <img width="25" height="25" src="img/co2.png">
                            </td>
                            <td style="padding-left:10px">
                                <p id="CO2Field" style="color:□#000000"></p>
                            </td>
                        </tr>
                        <tr>
                            <td>
                                <img width="25" height="25" src="img/polution.png">
                            </td>
                            <td style="padding-left:10px">
                                <p id="PolutField" style="color:□#000000"></p>
                            </td>
                        </tr>
                    </table>
                </div>
      </div>
    </div>
  </div>
```

Рис. 2.6.2 Заготовка модального вікна

Наступним кроком є створення скрипту на мові JavaScript , який буде обробляти взаємодію з елементами у яких є аргумент *data-toggle="modal"* , буде

отримувати інформацію о стані навколишнього середовища аудиторії з серверу та буде додавати цю інформацію до модального вікна, що показано на лістингу 2.

```
document.addEventListener('click', function(e) {
    if (e.target.dataset.toggle === 'modal') {
                modal = document.getElementById('tempModal');
                        var roomNumber =
            document.getElementById('roomNumber');
                         var tempField = document.getElementById('TempField');
                        var humField = document.getElementById('HumField');
                        var co2Field = document.getElementById('CO2Field');
                         var polutField = document.getElementById('PolutField');
            var roomData = getRoomData(e.target.id);
                        roomNumber.innerHTML = "Стан середовища в
                аудиторії " + roomData[0];
                         tempField.innerHTML = "Температура: " + roomData[1]
+ " °C";
                humField.innerHTML = "Вологість приміщення: "+
roomData[2] + " %";
                        co2Field.innerHTML = "Чадний газ: "+ roomData[3] + "
ppm";
                polutField.innerHTML = "Якість повітря (AQI): " +
roomData[4];
            var btn = document.getElementById(e.target.id);
                        var span =
document.getElementsByClassName("close")[0];
                         btn.onclick = function() {modal.style.display = "block";}
                        span.onclick = function() {modal.style.display = "none";}
                        window.onclick = function(event) {
                        if (event.target == modal) { modal.style.display = "none";}
                        }}
                });
```

Лістинг 2

Функція *“getRoomData”* , це раніше реалізована функція, яка дозволяє отримати дані з серверу. Отже після виконання приведених вище функцій, утворюється модальне вікно, яке можно побачити на рис. 2.6.3

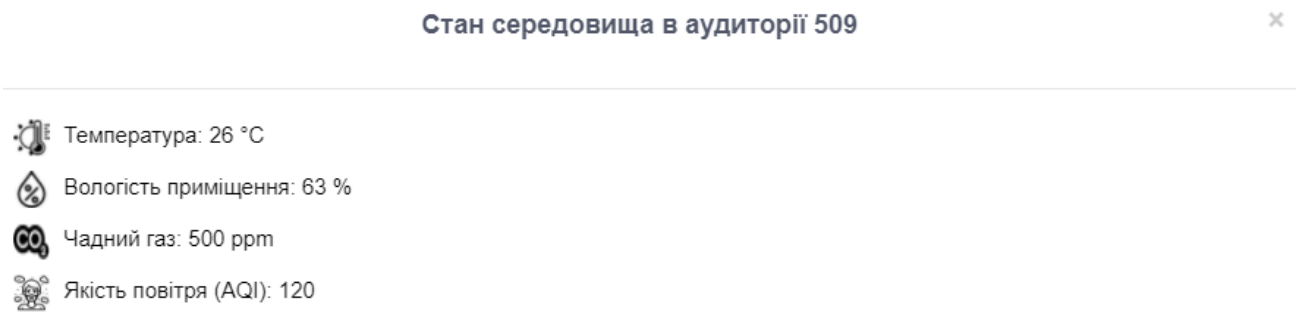


Рис. 2.6.3 Модальне вікно

А також наведемо до вашої уваги загальний вигляд модального вікна на фоні сторінки сайту , дивитися рис. 2.6.4

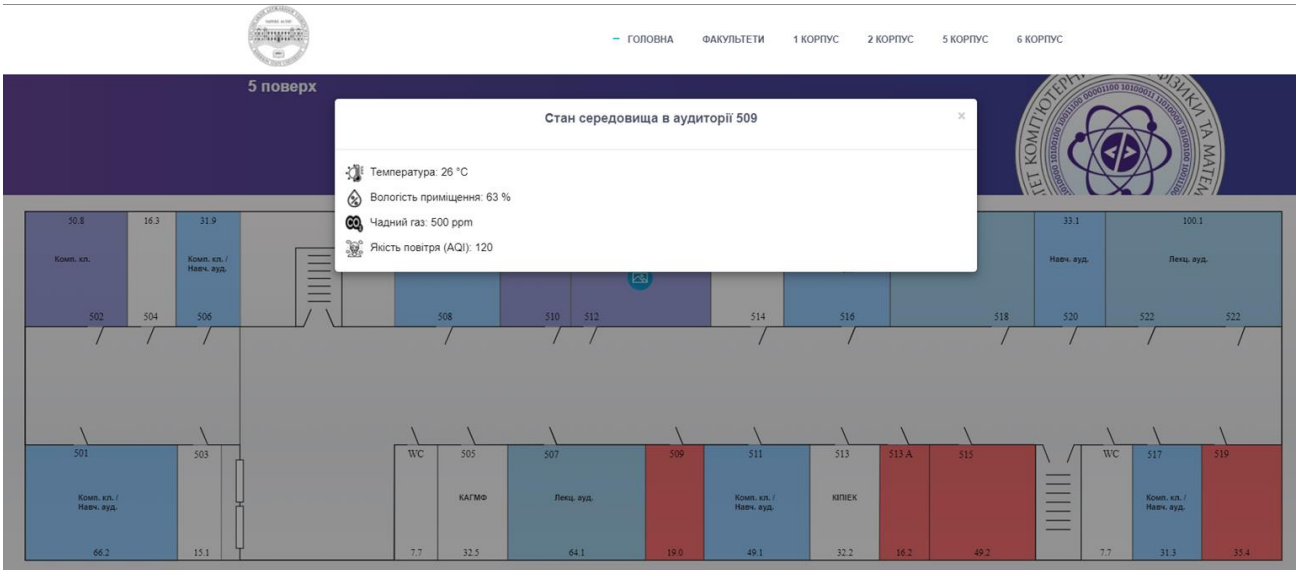


Рис. 2.6.4 Загальний вигляд модального вікна на фоні сторінки сайту

ВИСНОВКИ

Під час дослідження було проаналізовано типову літературу з цього питання. Розроблено програмне забезпечення для ESP8266 та модулів, які дозволяють отримувати, відображати та надсилати екологічну інформацію. Також було проаналізовано документацію ThingsBoard IoT Platform, що дозволило нам розгорнути та налаштувати зручне та гнучке середовище для обробки, зберігання та відображення інформації, отриманої від ESP8266. Такий підхід дозволяє відображати та отримувати актуальну екологічну інформацію в режимі реального часу.

Під час курсової роботи були виконані наступні завдання:

- Аналіз популярних IoT платформ та мікроконтролерів

- Розробка серверу для обміну та збереження інформації:
    - Розгортання та налаштування серверу.
    - Розгортання та налаштування бази даних.
    - Створення Dashboard для аналізу та відображення інформації отриманої від пристрою.
    - Розгортання та налаштування API для обміну даними.

- Розробка системи моніторингу даних о екологічному стані навчальних аудиторій:
    - Сборка інструменту на базі мікроконтролера ESP8266 для збору екологічної інформації.
    - Сборка LED дісплею на базі мікроконтролера ESP8266 об’єднаного з LED панелею P10 та.

- Модифікація університетських web сайтів

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Mehta, Mannan. ESP 8266: a breakthrough in wireless sensor networks and internet of things. International Journal of Electronics and Communication Engineering & Technology, 2015, 6.8: 7-11.
2. Syafa'ah, Lailis, et al. ESP 8266 for Control and Monitoring in Smart Home Application. In: 2019 International Conference on Computer Science, Information Technology, and Electrical Engineering (ICOMITEE). IEEE, 2019. p. 123-128.
3. Sahtyawan, Rama; WICAKSONO, Arief Ikhwan. Application for Control of Distance Lights Using Microcontroller Nodemcu Esp 8266 Based on Internet of Things (IoT). vol, 2020, 3839: 43-50.
4. Badamasi, Yusuf Abdullahi. The working principle of an Arduino. In: 2014 11th international conference on electronics, computer and computation (ICECCO). IEEE, 2014. p. 1-4.
5. Kumar, N. Sathish, et al. IOT based smart garbage alert system using Arduino UNO. In: 2016 IEEE region 10 conference (TENCON). IEEE, 2016. p. 1028-1034.
6. DE PAOLIS, Lucio Tommaso; DE LUCA, Valerio; PAIANO, Roberto. Sensor data collection and analytics with thingsboard and spark streaming. In: 2018 IEEE workshop on environmental, energy, and structural monitoring systems (EESMS). IEEE, 2018. p. 1-6.
7. KADARINA, T. M.; PRIAMBODO, R. Monitoring heart rate and SpO2 using Thingsboard IoT platform for mother and child preventive healthcare. In: IOP conference series: materials science and engineering. IOP Publishing, 2018. p. 012028.
8. AGHENTA, Lawrence Oriaghe; IQBAL, Tariq. Design and implementation of a low-cost, open source IoT-based SCADA system

using ESP32 with OLED, ThingsBoard and MQTT protocol. AIMS Electronics and Electrical Engineering, 2019, 4.1: 57-86.

9. PASHA, Sharmad. ThingSpeak based sensing and monitoring system for IoT with Matlab Analysis. International Journal of New Technology and Research (IJNTR), 2016, 2.6: 19-23.
10. MAUREIRA, Marcello A. Gómez; OLDENHOF, Daan; TEERNSTRA, Livia. ThingSpeak–an API and Web Service for the Internet of Things. World Wide Web, 2011.
11. CALDERONI, Luca. Preserving context security in AWS IoT Core. In: Proceedings of the 14th International Conference on Availability, Reliability and Security. 2019. p. 1-5.
12. KURNIAWAN, Agus. Learning AWS IoT: Effectively manage connected devices on the AWS cloud using services such as AWS Greengrass, AWS button, predictive analytics and machine learning. Packt Publishing Ltd, 2018.
13. DAVOLI, Renzo; SGUEGLIA, Umberto. Sistemi integrati per la domotica e per il risparmio energetico tramite micro elaboratori su singola scheda e basso costo.
14. GUPTA, Deepti, et al. Access control model for google cloud iot. In: 2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS). IEEE, 2020. p. 198-208.
15. PIERLEONI, Paola, et al. Amazon, Google and Microsoft solutions for IoT: architectures and a performance comparison. IEEE access, 2019, 8: 5455-5470.
16. UCUZ, Derya, et al. Comparison of the IoT Platform Vendors, Microsoft Azure, Amazon Web Services, and Google Cloud, from Users'

Perspectives. In: 2020 8th International Symposium on Digital Forensics and Security (ISDFS). IEEE, 2020. p. 1-4.

17. BAKHSHI, Zeinab; BALADOR, Ali; MUSTAFA, Jawad. Industrial IoT security threats and concerns by considering Cisco and Microsoft IoT reference models. In: 2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW). IEEE, 2018. p. 173-178.
18. MOHAMED, Khaled Salah. IoT cloud computing, storage, and data analytics. In: The Era of Internet of Things. Springer, Cham, 2019. p. 71-91.
19. WOO, Sangyeon; SONG, Jeho; PARK, Sungyong. A distributed oracle using intel SGX for blockchain-based iot applications. Sensors, 2020, 20.9: 2725.
20. MACHADO, Gonçalo Carteado Pinho. Aplpication developmente over IoT platform Thingworx. 2018. PhD Thesis. Universidade de Coimbra.
21. VANIN, P. A.; NESTEROV, A. S.; KHOLODILIN, I. Y. Integration of IIoT and AR technologies to educational process through laboratory complex. In: 2018 Global Smart Industry Conference (GloSIC). IEEE, 2018. p. 1-6.
22. CROCKFORD, Douglas. JavaScript: The Good Parts: The Good Parts. " O'Reilly Media, Inc.", 2008.
23. GUHA, Arjun; SAFTOIU, Claudiu; KRISHNAMURTHI, Shriram. The essence of JavaScript. In: European conference on Object-oriented programming. Springer, Berlin, Heidelberg, 2010. p. 126-150.

24. YUE, Chuan; WANG, Haining. Characterizing insecure JavaScript practices on the web. In: Proceedings of the 18th international conference on World wide web. 2009. p. 961-970.
25. GUPTA, Suhit, et al. DOM-based content extraction of HTML documents. In: Proceedings of the 12th international conference on World Wide Web. 2003. p. 207-214.
26. MUSCIANO, Chuck; KENNEDY, Bill. HTML & XHTML: The Definitive Guide: The Definitive Guide. " O'Reilly Media, Inc.", 2002.
27. DUCKETT, Jon. HTML & CSS: design and build websites. Indianapolis, IN: Wiley, 2011.
28. NIXON, Robin. Learning PHP, MySQL & JavaScript. " O'Reilly Media, Inc.", 2021.
29. MEYER, Eric A. CSS: The Definitive Guide: The Definitive Guide. " O'Reilly Media, Inc.", 2006.
30. ALAWAR, Mariam W.; ABU-NASER, Samy S. CSS-Tutor: An intelligent tutoring system for CSS and HTML. 2017.