Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: *Комп'ютеризована система віддаленого керування альтернативним охолодженням приміщень для зберігання запасів продуктів*

Виконав: студент *IV* курсу, групи *СІ-41*
спеціальності *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

		Олійник М. Є.
	(підпис)	(прізвище та ініціали)
Керівник		*Луцик Н. С.*
	(підпис)	(прізвище та ініціали)
Нормоконтроль		*Тиш Є. В.*
	(підпис)	(прізвище та ініціали)
Завідувач кафедри		*Осухівська Г. М.*
	(підпис)	(прізвище та ініціали)
Рецензент		
	(підпис)	(прізвище та ініціали)

Тернопіль
2023

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет *комп'ютерно-інформаційних систем і програмної інженерії*
(повна назва факультету)

Кафедра *комп'ютерних систем та мереж*
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
Осухівська Г.М.
(підпис) (прізвище та ініціали)
« _2_ » _____03_____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня *бакалавр*
(назва освітнього ступеня)

за спеціальністю *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

студенту *Олійнику Михайлу Євгеновичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Комп'ютеризована система віддаленого керування альтернативним охолодженням приміщень для зберігання запасів продуктів*

Керівник роботи *Луцик Надія Степанівна, доктор філософії, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*28*» *02* *2023 року* № *4/7-238*

2. Термін подання студентом завершеної роботи *22.06.2023 р.*

3. Вихідні дані до роботи *Технічне завдання*

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз технічного завдання
2. Проектна частина
3. Практична система
4. Безпека життєдіяльності, основи охорони праці.
Висновки. Список використаних джерел

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Функціональна схема комп'ютеризованої системи
2. Схема електрична принципова
3. Діаграма послідовностей
4. Логічна схема архітектури сервера комп'ютеризованої системи
5. Блок-схема функції аналізу стану повітря

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності,	*Пилипець М. І., д.т.н. проф. каф. МТ*		
основи охорони праці			

7. Дата видачі завдання 02.03.2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	*Розробка та затвердження технічного завдання*	*01.03-02.03*	*Виконано*
2	*Аналіз технічного завдання та побудова можливих рішень*	*04.03-10.03*	*Виконано*
3	*Аналіз складського приміщення*	*11.03-15.03*	*Виконано*
4	*Розробка структурної та функціональної схеми*	*16.03-24.03*	*Виконано*
5	*Розробка діаграми послідовностей та архітектури серверної частини розроблюваної системи*	*25.03-31.03*	*Виконано*
6	*Розробка схеми електричної принципової*	*01.04-04.04*	*Виконано*
7	*Вибір елементної бази*	*05.04-09.04*	*Виконано*
8	*Моделювання апаратної складової комп'ютеризованої системи*	*10.04-30.04*	*Виконано*
9	*Реалізація програмного забезпечення комп'ютеризованої системи*	*01.05-25.05*	*Виконано*
10	*Налаштування та тестування реалізованої системи*	*26.05-31.05*	*Виконано*
11	*Безпека життєдіяльності, основи охорони праці*	*01.06-04.06*	*Виконано*
12	*Оформлення кваліфікаційної роботи*	*05.06-12.06*	*Виконано*
13	*Оформлення графічної частини*	*13.06-15.06*	*Виконано*
14	*Попередній захист кваліфікаційної роботи*	*16.06-19.06*	*Виконано*
15	*Захист кваліфікаційної роботи*	*22.06.2023*	

Студент _____ *Олійник М. Є.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Луцик Н. С.*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Комп’ютеризована система віддаленого керування альтернативним охолодженням приміщень для зберігання запасів продуктів // Кваліфікаційна робота бакалавра // Олійник Михайло Євгенович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп’ютерно-інформаційних систем і програмної інженерії, кафедра комп’ютерних систем та мереж, група СІ-41 // Тернопіль, 2023 // с. – 123, рис. – 46, аркушів А1 – 5, бібліогр. – 19.

Ключові слова: КОМП’ЮТЕРИЗОВАНА СИСТЕМА, ВІДДАЛЕНЕ КЕРУВАННЯ, АЛЬТЕРНАТИВНЕ ОХОЛОДЖЕННЯ, МІКРОКОНТРОЛЕР, ДАВАЧ ТЕМПЕРАТУРИ.

Кваліфікаційна робота присвячена розробці комп'ютеризованої системи віддаленого керування альтернативним охолодженням приміщень для зберігання запасів продуктів. Пояснювальна записка складається із чотирьох розділів.

В першому розділі проведено аналіз вимог до комп’ютеризованої системи. Було розглянуто можливі рішення для виконання поставленого завдання та проведено оцінку актуальності впровадження розроблюваної системи.

В другому розділі було обґрунтовано вибір апаратної та програмної складових для реалізації комп’ютеризованої системи.

В третьому розділі описана практична частина, де проводилась розробка програмного і апаратного забезпечення та його подальше тестування.

В четвертому розділі розглянуті питання безпеки життєдіяльності та охорони праці.

ANNOTATION

Computerized remote control system for alternative cooling of premises for storing food stocks // Qualification work // Mykhailo Oliinyk // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information System and Software Engineering, Department of Computer Systems and Networks, group CI-41 // Ternopil, 2023 // p. – 123, fig. – 46, sheets A1 – 5, ref. – 19.

Keywords: COMPUTERIZED SYSTEM, REMOTE CONTROL, ALTERNATIVE COOLING, MICROCONTROLLER, TEMPERATURE SENSOR.

The qualification work is devoted to the development of a computerized system for remote control of alternative cooling of premises for storing product stocks. The explanatory note consists of four sections.

The first section analyzes the requirements for a computerized system. Possible solutions for the task were considered and the relevance of the implementation of the developed system was assessed.

In the second section, the choice of hardware and software components for the implementation of the computerized system was substantiated.

The third section describes the practical part where software and hardware development and its further testing were carried out.

The fourth section discusses the issues of life safety and labor protection.

ЗМІСТ

					КС КРБ 123.050.00.00 ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розробив		Олійник М.Є.			Комп’ютеризована система віддаленого керування альтернативним охолодженням приміщень для зберігання запасів продуктів	Літ.	Арк. 6	Акркушів
Перевірив		Луцик Н. С.						
Рецензент						ТНТУ, каф. КС, гр. СІ-41		
Н. Контр.		Тиш Є. В.						
Затвердив		Осухівська Г.М.						

СПИСОК СКОРОЧЕНЬ

ІСР – Інтегроване середовище розробки

ПЗ – Програмне забезпечення

ЦП – Центральний процесор

СКБД – Система керування базами даних

JWT – JSON Web Token

ВСТУП

У сучасному світі зберігання продуктів стає все більш важливою задачею, особливо в умовах змін клімату. Одним із ключових аспектів забезпечення якості та тривалості зберігання продуктів є правильне охолодження приміщень, де вони знаходяться. Традиційні системи охолодження, які використовуються на багатьох підприємствах та складах, є неефективними з точки зору енергоспоживання. Вони вимагають значних витрат електроенергії для підтримки постійно низької температури у приміщенні, не зважаючи на умови навколишнього середовища.

Метою даної кваліфікаційної роботи є розробка комп'ютеризованої системи віддаленого керування альтернативним охолодженням приміщень для зберігання запасів продуктів. Основним принципом цієї системи є використання природних ресурсів, а саме холодне повітря зовнішнього середовища. Таким чином, комп’ютеризована система буде автоматично регулювати температуру забезпечуючи оптимальні умови для зберігання продуктів без необхідності постійного використання енергоємних систем охолодження.

Для досягнення поставленої мети необхідно виконати такі завдання:

– провести аналіз погодних умов для ефективної реалізації даної системи;

– створити систему давачів, механізму вентиляції та автоматизованого мікроконтролера для ефективного регулювання температури;

– розробити алгоритм роботи та програмне забезпечення;

– провести налаштування та тестування розробленої комп’ютеризованої системи.

Об'єктом дослідження даного проєкту є процес забезпечення оптимальних умов температури в приміщенні для зберігання продуктів з використанням природних ресурсів та комп’ютеризованої системи управління. Комп'ютеризована система віддаленого керування альтернативним охолодженням приміщень для зберігання запасів продуктів буде ефективним і стійким рішенням, яке забезпечить оптимальні умови зберігання продуктів з використанням мінімальних енергетичних ресурсів.

Особливістю даної системи охолодження є впровадження нового підходу для охолодження складських приміщень, з використанням природних ресурсів та автоматизованої системи. Це допоможе досягнути кращих показників енергоефективності та зменшити негативний вплив на навколишнє середовище.

Впровадження розробленої системи на різноманітних підприємствах та складах, що займаються зберіганням продуктів, дозволить зменшити витрати на енергію та забезпечити ефективне зберігання продуктів без шкідливого впливу на навколишнє середовище.

Під час виконання кваліфікаційної роботи використовувались наступні методи дослідження:

– аналітичний метод: для аналізу кліматичних умов та оцінки актуальності даної систем охолодження;

– експериментальний метод: для проведення досліджень та оцінки ефективності розробленої комп'ютеризованої системи;

– моделювання: для створення моделі та симуляції роботи системи охолодження.

Необхідний для реалізації інструментарій включатиме сервіси для побудови блок-схем, електрично принципових схем і алгоритмів роботи. Інтегровані середовища розробки програмного забезпечення та спеціалізоване обладнання для створення прототипу апаратної складової системи.

Результати дослідження можуть мати велику практичну цінність для підприємств, які займаються зберіганням продуктів, забезпечуючи енергоефективність та зменшення негативного впливу на довкілля.

РОЗДІЛ 1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

На сьогоднішній день є різноманітна кількість пристроїв охолодження приміщень для зберігання запасів продуктів. Опишемо деякі існуючі системи охолодження та їхні недоліки:

Компресорні системи охолодження: використовують компресори для охолодження приміщень. Вони ефективні, але гучні та споживають багато електроенергії.

– Кондиціонери: часто використовуються для охолодження приміщень. Однак, вони вимагають значних енергетичних та високих експлуатаційних витрат. Також вони не є екологічно чистими через використання охолоджуючих речовин.

– Системи з водяним охолодженням: використовують воду в якості теплопровідника для охолодження приміщень. Ці системи вимагають складної інфраструктури та можуть бути вразливими до проблем з утриманням чистоти води та ризику замерзання.

Розроблювана комп’ютеризована система використовує природні ресурси для охолодження. Коли температура навколишнього середовища нижча за температуру в приміщенні, комп’ютеризована система активує провітрювання, тим самим забезпечуючи природне охолодження. Це зменшує енергетичні витрати та експлуатаційні витрати, так як система не використовує холодильні установки. Крім того, розроблена система екологічно чиста, оскільки не використовує шкідливих речовин чи газів. Вона є малогабаритною та простою для інсталяції, ефективною і зручною в управлінні, забезпечуючи досить стабільне охолодження приміщення, використовуючи природні ресурси.

1.1 Аналіз вимог до комп'ютеризованої системи

За даними технічного завдання до кваліфікаційної роботи, необхідно здійснити малогабаритну, недорогу та ефективну систему охолодження приміщення для зберігання запасів продуктів із можливістю віддаленого керування. Дана система повинна відповідати наступним технічним характеристикам.

Апаратна частина:

– діапазон вимірювання температури повітря від -20 ℃ до 40 ℃;

– можливість локального да віддаленого налаштування та контролю системи;

– постійний аналіз системи для виявлення та реагування на несправності;

– автоматичний контроль системою вентиляції для підтримання оптимального діапазону температури.

Клієнт-серверна частина:

– можливість реєстрації, авторизації та функції додавання пристроїв та приміщень;

– архітектура із можливістю подальшого розширення більшого діапазону пристроїв.

Для реалізації вищесказаних функцій система включатиме різного роду давачі для продуктивного функціонування та ІСР із сучасними бібліотеками розробки ПЗ для побудови розширюваної архітектури та авторизованого доступу до контролю пристроїв. Тому, в кваліфікаційній роботі буде здійснено порівняльний аналіз давачів, архітектурної моделі та бібліотек для їхньої реалізації.

Для підтримання оптимального діапазону температури, комп’ютерна система буде знімати дані з внутрішніх та зовнішніх давачів та порівнюватиме їхні значення. Згідно до виставлених параметрів дана система вмикатиме чи вимикатиме систему вентиляції.

Основною вимогою технічного завдання є забезпечення безперебійної роботи системи, незважаючи на стан підключення до мережі Internet. Саме для цього було враховано локальний та синхронізований режими роботи. Для відображення основної інформації пристрою було передбачено зручне меню. Дане меню відображатиметься на OLED-дисплеї, а для його контролю буде включена клавіатура дотиків із 4 сенсорів. Дана клавіатура дає змогу відкрити панель налаштувань, обрати необхідний режим роботи та виставити основні параметри роботи пристрою, які включають:

– діапазон оптимальної температури;

– максимальний час роботи пристрою;

– час перерви між зупинками;

– максимальний час повторного ввімкнення після попереднього завершення;

– дозвіл на синхронізацію;

– інтервал обміну даними;

– інтервал отримання даних з пристроїв;

– інтервал оцінювання продуктивності системи.

Клієнт-серверна частина дозволятиме користувачу зареєструватись та увійти на свою персональну сторінку. На якій він матиме змогу створювати різні розділи (віртуальні кімнати) та додати у них любий підтримуваний сервером пристрій за допомогою унікального ідентифікатора. Для даної кваліфікаційної роботи ми обмежимось лише одним пристроєм, а саме пристроєм альтернативного охолодження. Для нього буде створена відповідна модель представлення даних, яка відображатиме поточний стан пристрою і включатиме можливість задання діапазону оптимальної температури та вмикати чи вимикати пристрій.

1.2 Аналіз можливих рішень поставленого завдання

1.2.1 Оцінка актуальності впровадження розробленої систем охолодження

З поступовим ростом глобального потепління, необхідно оцінити ефективність та актуальність впровадження даної системи охолодження, яка базується на використанні природних ресурсів. Для цього наведено статистику середньо місячної температури повітря по областях України [2], яка проілюстрована на рисунку 1.1.

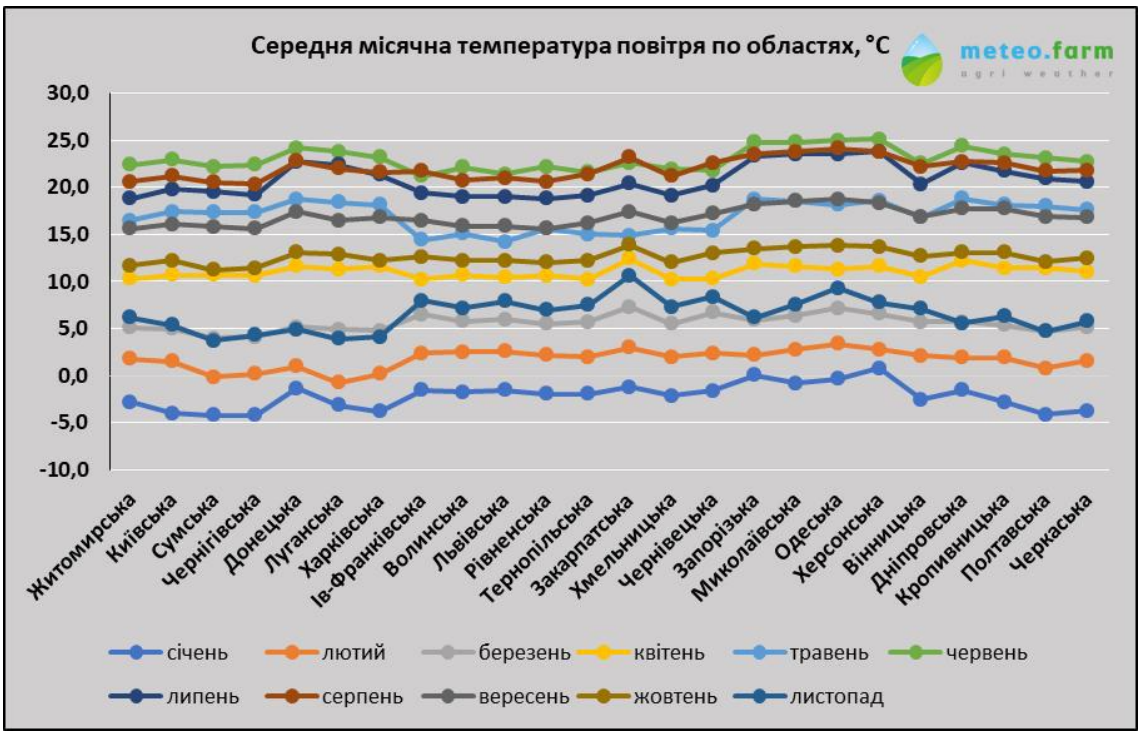


Рисунок 1.1 – Статистика середньомісячної температури по областях України

На рисунку 1.1 показано, що найвищі показники температури припадають на червень.

На рисунку 1.2 представлено статистику за цей місяць детальніше, а саме медіану температурних показників червня за даними від 2013 до 2022 років.

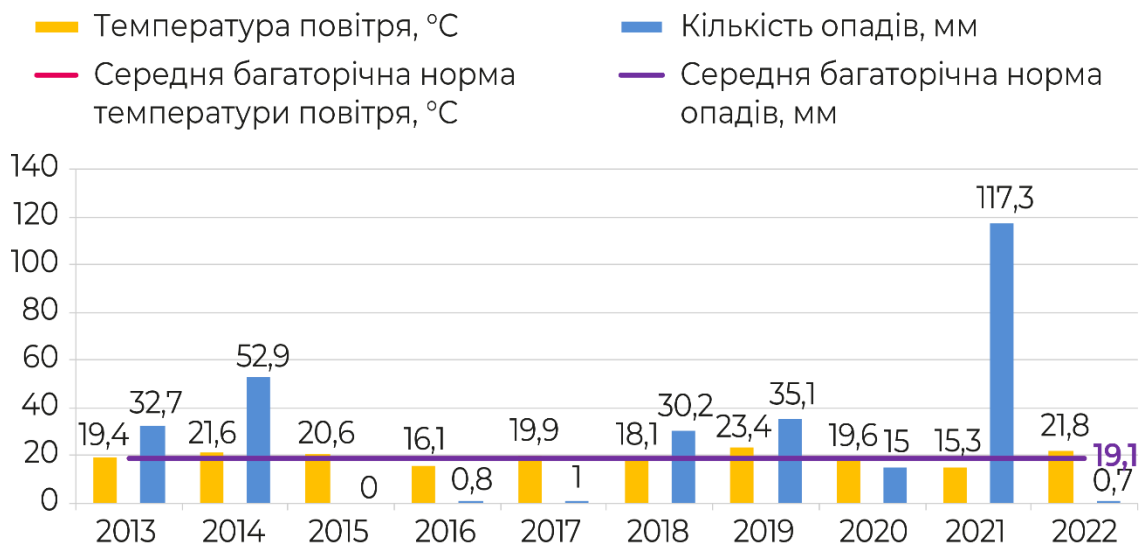


Рисунок 1.2 – Середні значення гідротермічних показників у першій декаді червня, 2013-2022 рр.

Видно, що середні значення температури в найтепліший місяць коливаються в межах 15-24 ℃. Оскільки це середні показники нам необхідно визначити їхні екстремуми в межах дня. На рисунку 1.3 відображено графік показників температури та точки роси впродовж одного місяця.

Рисунок 1.3 – Місячний графік показників температури за Червень 2020р.

Отже, найбільше значення мінімуму є 20℃, але з врахуванням теплоізоляції приміщення знайдемо середнє значення мінімумів, а це близько 16℃. Враховуючи, що консервовані продукти повинні зберігатись в межах +5…+18℃, актуальність застосування даної системи охолодження є прийнятною навіть в найтепліший місяць року.

1.2.2 Аналіз складського приміщення

Для забезпечення необхідних вимог до апаратної частини системи охолодження слід проаналізувати приміщення в якому воно працюватиме. А саме визначити його корисний об’єм. Це дозволить розрахувати необхідні габарити та потужність складових компонентів системи вентиляції. Що в свою чергу несе прямий вплив на ефективність роботи розроблюваної системи охолодження.

Для визначення корисного об’єму приміщення було виведено наступну формулу:

$$V_k = V_{\text{П}} - V_O. \tag{1.1}$$

де V_k – корисний об’єм приміщення;

$V_{\text{П}}$ – загальний об’єм приміщення;

V_O – об’єм сторонніх об’єктів всередині приміщення;

Для знаходження загального об’єму приміщення:

$$V_{\text{П}} = S_{\text{П}} * h_{\text{П}}. \tag{1.2}$$

де $S_{\text{П}}$ – площа основи приміщення;

$h_{\text{П}}$ – висота приміщення;

Якщо приміщення нестандартних розмірів, необхідно обраховувати об’єм поділивши приміщення на ділянки. Для кожної із якої підібрати формулу обчислення об’єму. Для отримання повного об’єму приміщення слід просумувати об’єм кожної із його ділянок. Таку саму процедуру слід провести для обрахунку об’єму сторонніх об’єктів приміщення.

РОЗДІЛ 2 ПРОЕКТНА ЧАСТИНА

2.1 Побудова узагальненої структури комп'ютеризованої системи віддаленого керування альтернативним охолодженням приміщень

Згідно вимог до розроблюваної системи було побудовано функціональну схему комп'ютеризованої системи віддаленого керування альтернативним охолодженням приміщень для зберігання запасів продуктів, яка відображає взаємодію між складовими елементами проєкту (рис. 2.1).

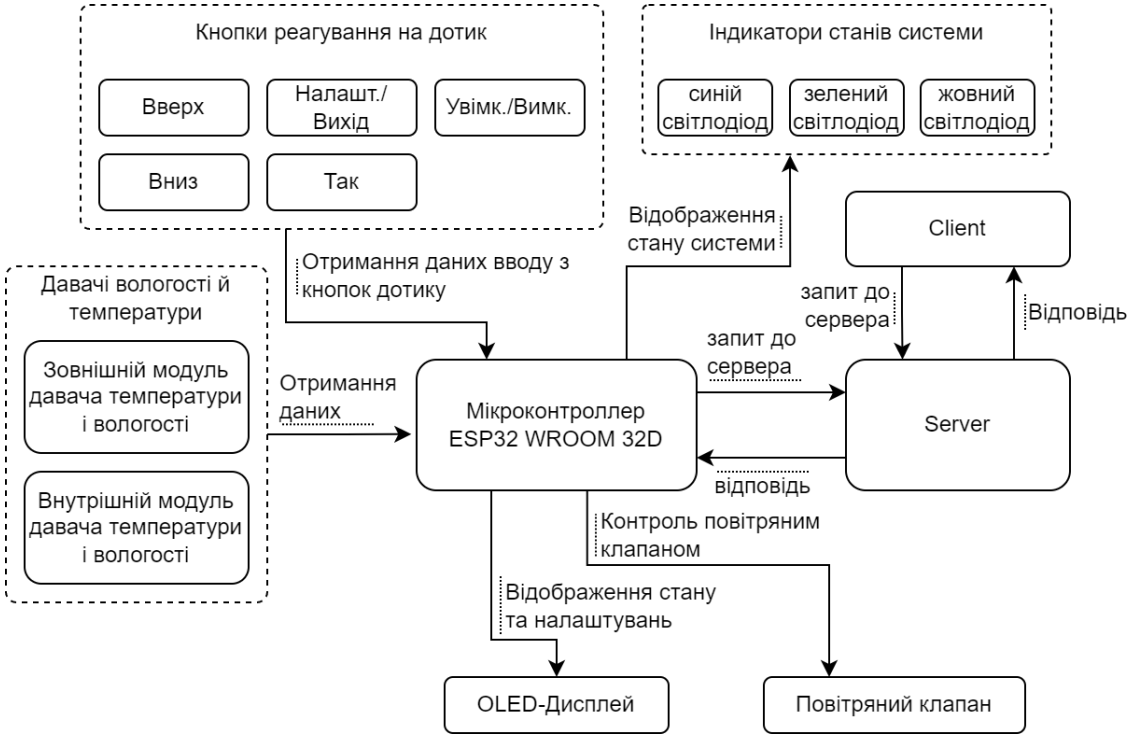


Рисунок 2.1 – Функціональна схема комп'ютеризованої системи охолодження

Структурна схема розроблюваної комп'ютеризованої системи проілюстрована на рисунку 2.2.

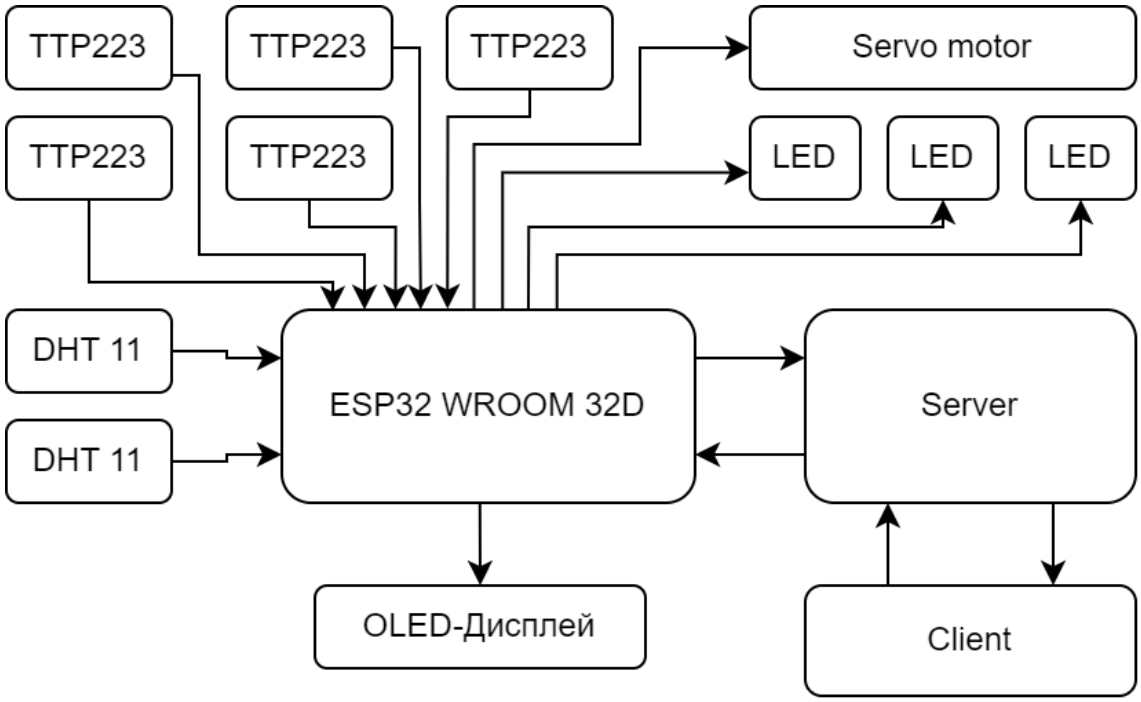


Рисунок 2.2 – Структурна схема комп’ютеризованої системи охолодження

Дана комп’ютеризована система складається з двох давачів температури та вологості повітря DHT11, які аналізують стан повітря всередині приміщення та зовнішнього середовища. Для ввімкнення та налаштування системи є сенсорна клавіатура, яка складається із 5 модулів TTP223. Керування повітряним клапаном здійснюється за допомогою сервопривода SG90. Відображення основних параметрів системи здійснюватиметься на OLED-дисплеї, який підключений за допомогою інтерфейсу 1-Ware.

Обмін даними між мікроконтролером ESP32 WROOM 32D та сервером здійснюватиметься за допомогою протоколу http. Для відображення стану системи на клієнтському web-сайті комп’ютеризована система надсилатиме дані кожні 10 секунд та в залежності від відповіді виконуватиме зміну параметрів.

Взаємодія між клієнтською та серверною частинами відображена у вигляді діаграми послідовностей використання (sequence diagram), яка зображена на рисунку 2.3.

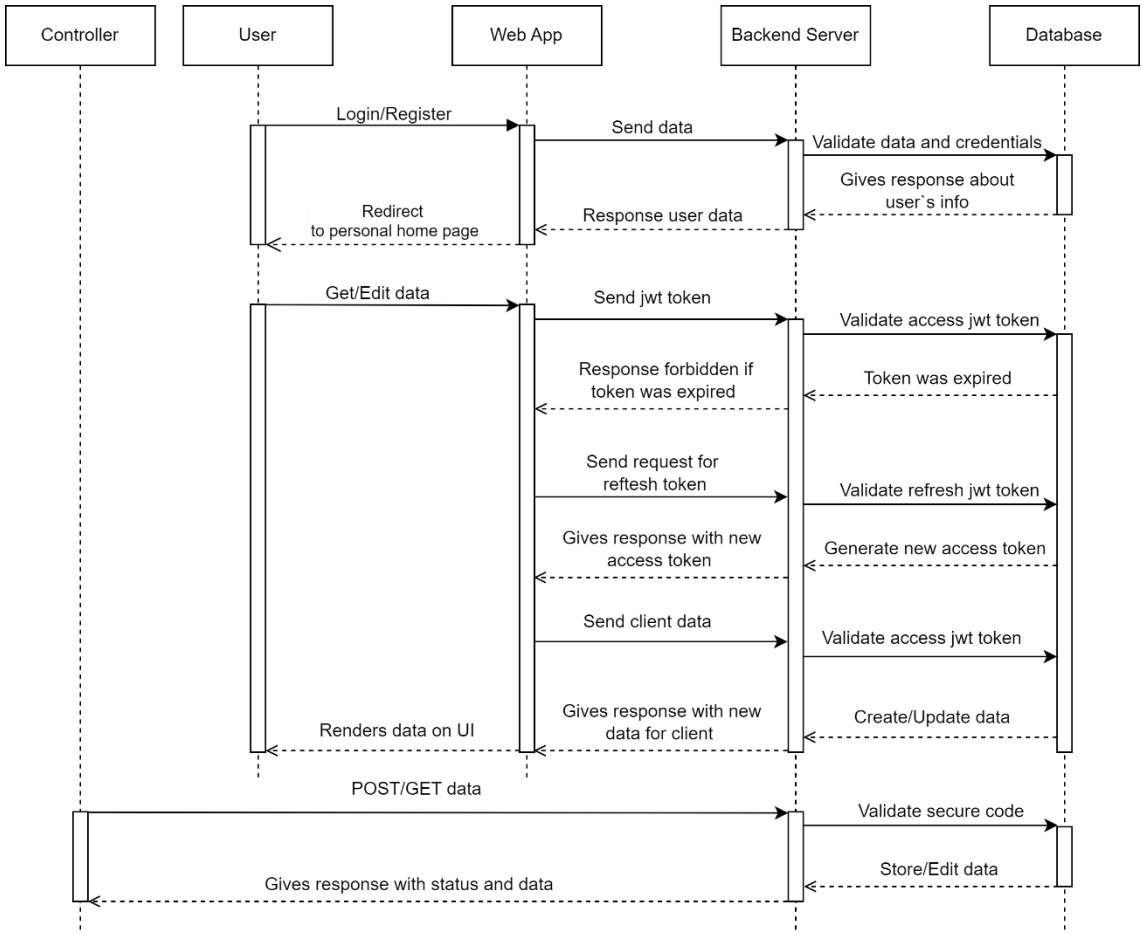


Рисунок 2.3 – Діаграма послідовностей

З даної діаграми можна побачити що спершу користувач реєструється і входить в свою персональну сторінку. Після входу він отримує свій JWT-токен доступу, час дійсності якого складає 1 хв. Кожного разу, як користувач вноситиме якісь зміни на своїй персональній сторінці та надсилатиме їх на сервер, спершу через інтерцептори JWT-токен перевіриться на дійсність, в разі його недійсності буде надіслано запит на отримання нового JWT-токена доступу. Опісля його отримання всі внесені користувачем зміни буде надіслано на сервер для їхнього збереження. В разі дійсності JWT-токена, дані будуть просто надіслані на сервер. Сервер отримує дані, звіряє користувача якого отримує із розшифрованого JWT-токена та в залежності від дозволів користувача вносить зміни у базі даних.

Контролер в свою чергу надсилає секретний код пристрою на сервер, який звіряється в базі даних. За наявності секретного коду пристрою, дані які надсилає пристрій зберігаються у відповідну колекцію бази даних.

Логічна схема архітектури сервера комп'ютеризованої системи віддаленого керування альтернативним охолодженням зображена на рисунку 2.4

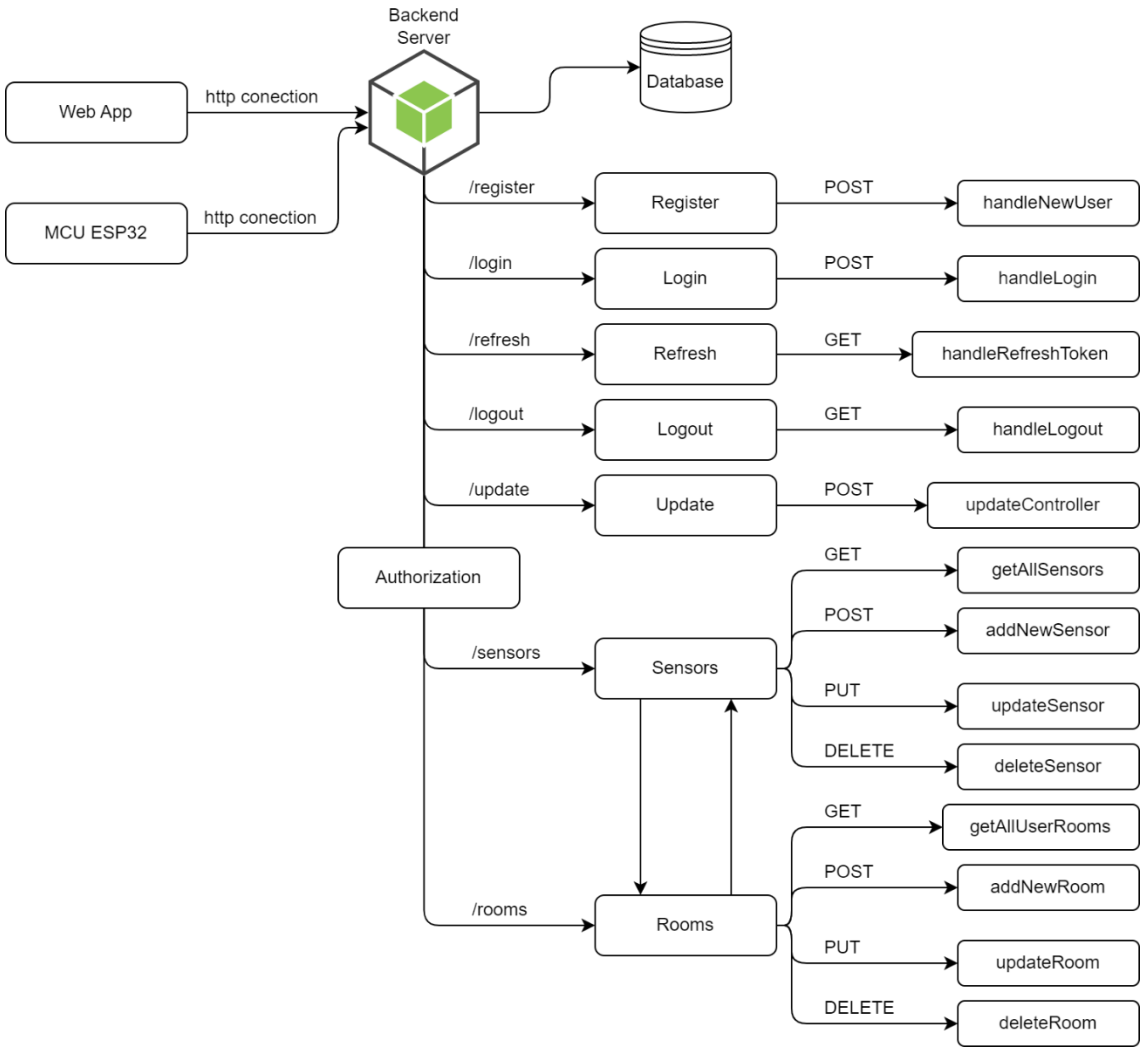


Рисунок 2.4 – Логічна схема архітектури сервера комп'ютеризованої системи віддаленого керування альтернативним охолодженням

Як видно на рисунку 2.4, сервер має доволі великий функціонал, який включає:

- авторизацію користувачів;
- безпечний доступ з використанням JWT токенів;
- створення, отримання, редагування та видалення розділів (кімнат);
- створення, отримання, редагування та видалення пристроїв.

Узагальнена блок-схема роботи апаратного забезпечення проектованої комп’ютеризованої системи проілюстрована на рисунку 2.5.

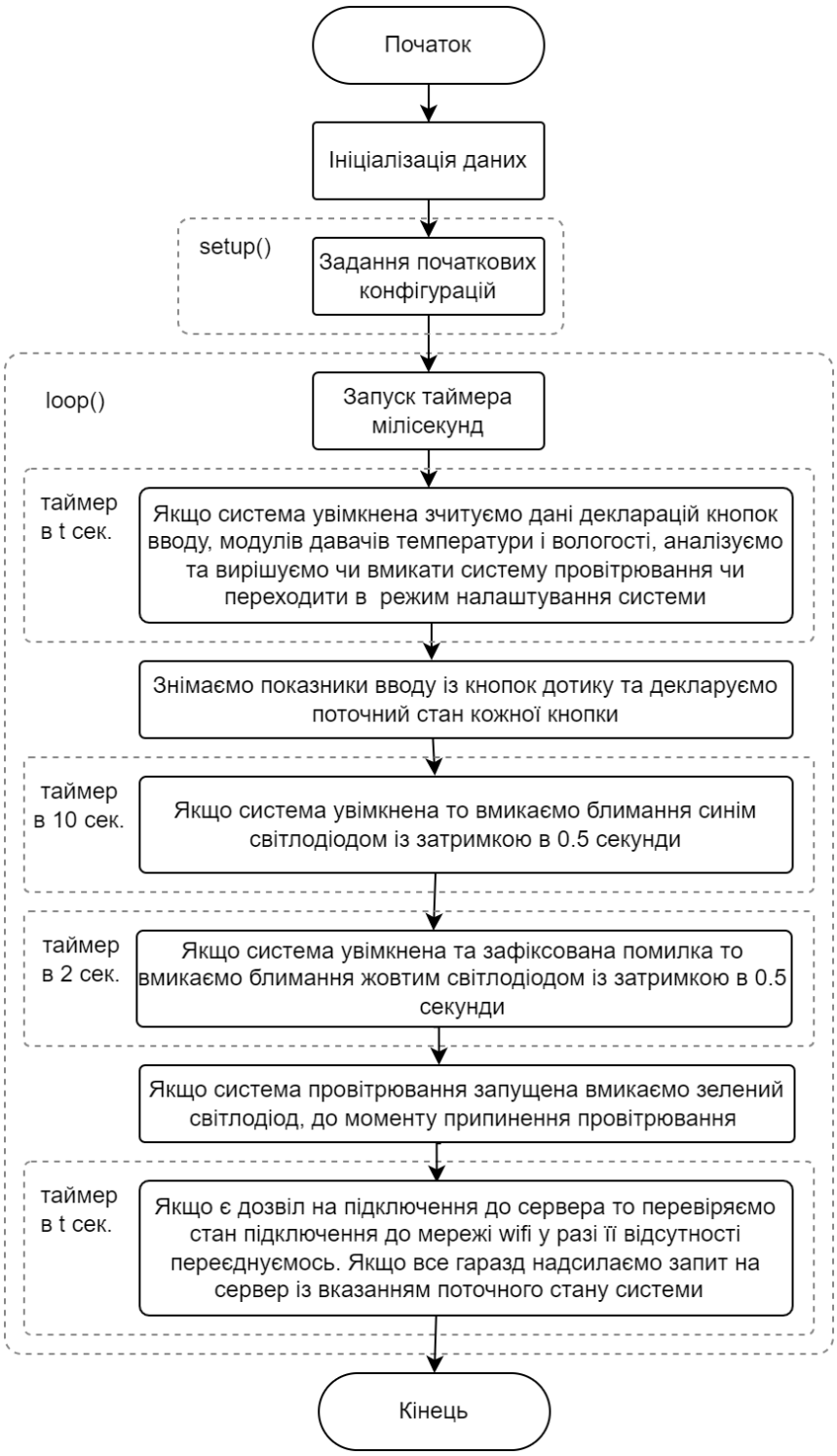


Рисунок 2.5 – Узагальнена блок-схема роботи апаратної складової системи

Після побудови усіх необхідних структурних схем, згідно вимог перейдемо до вибору апаратного та програмного забезпечення для реалізації розроблюваної КС.

2.2 Обгрунтування вибору апаратного забезпечення розробленої комп’ютеризованої системи

2.2.1 Мікроконтролер ESP32 WROOM

Основним керуючим елементом проектованої комп’ютеризованої системи є ESP32 WROOM - потужний і високопродуктивний пристрій з інтегрованим Wi-Fi і Bluetooth. Це дозволяє забезпечувати зв’язок із зовнішніми пристроями та інтернетом. Це дає можливість використовувати мікроконтролер для віддаленого керування проектованою системою охолодження, забезпечуючи зручність і гнучкість управління. Даний контролер базується на двоядерному мікропроцесорі Xtensa LX6 з тактовою частотою 240 МГц та вбудованою флеш пам’яттю на 4 МБ для зберігання програмного забезпечення та даних. Мікроконтролер ESP32 WROOM зображений на рисунку 2.6.

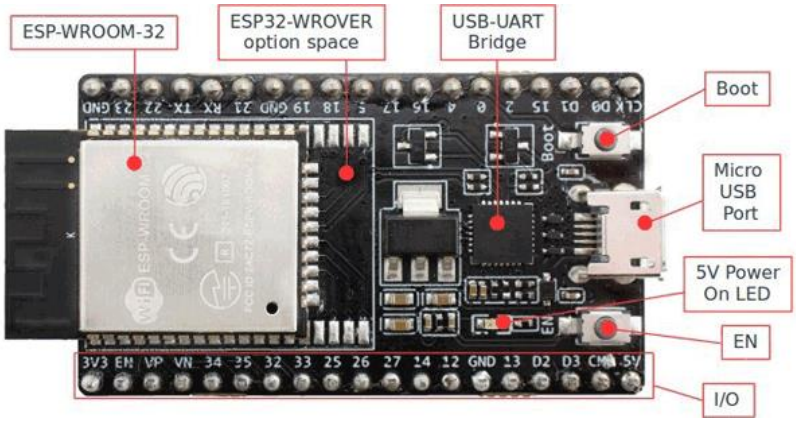


Рисунок 2.6 – Мікроконтролер ESP32 WROOM

ESP32 WROOM має широкий набір периферійних пристроїв, включаючи GPIO порти, ADC (аналого-цифровий перетворювач), UART (універсальний асинхронний приймач-передавач), I2C (двозв'язковий протокол), SPI (послідовна шина) та інші. Це дає нам можливість підключати та керувати різноманітними давачами, актуаторами та іншими зовнішніми пристроями для вимірювання температури, контролю вентиляції та зберігання даних.

Периферійні пристрої мікроконтролер ESP32 WROOM:

– Порт Micro USB, призначений для підключення ESP32 до комп’ютера для програмування за допомогою USB-кабелю.

– Кнопка Boot, призначена для завантаження програми з Arduino IDE в ESP32.

– Кнопка EN, використовується для скидання налаштувань модуля ESP.

– Світлодіод LED, вказівка джерела живлення, світиться червоним під час живлення.

– Контакти вводу/виводу, якщо режими для портів не прописано в коді, тоді параметри портів використовуватимуться за умовчанням.

– Міст USB-UART - це повністю інтегрований контролер USB-UART, який забезпечує підключення USB до пристроїв з інтерфейсом UART [5].

Важливою характеристикою ESP32 WROOM є його висока сумісність з різними програмними засобами. Можна використовувати Arduino IDE та розширення ESP32 для програмування та налагодження мікроконтролера. Це забезпечує зручність розробки та підтримку широкого спектру функціональності. Даний мікроконтролер має низьке споживання енергії, що є критичним для системи, яка повинна працювати тривалий час. Це забезпечує стабільне та ефективне охолодження приміщень для зберігання продуктів, не споживаючи зайву електроенергію.

Інтегрований процесор ESP32 WROOM розроблений із врахуванням можливості масштабування та адаптації. Функціональна блок-схема процесора зображена на рисунку 2.7.

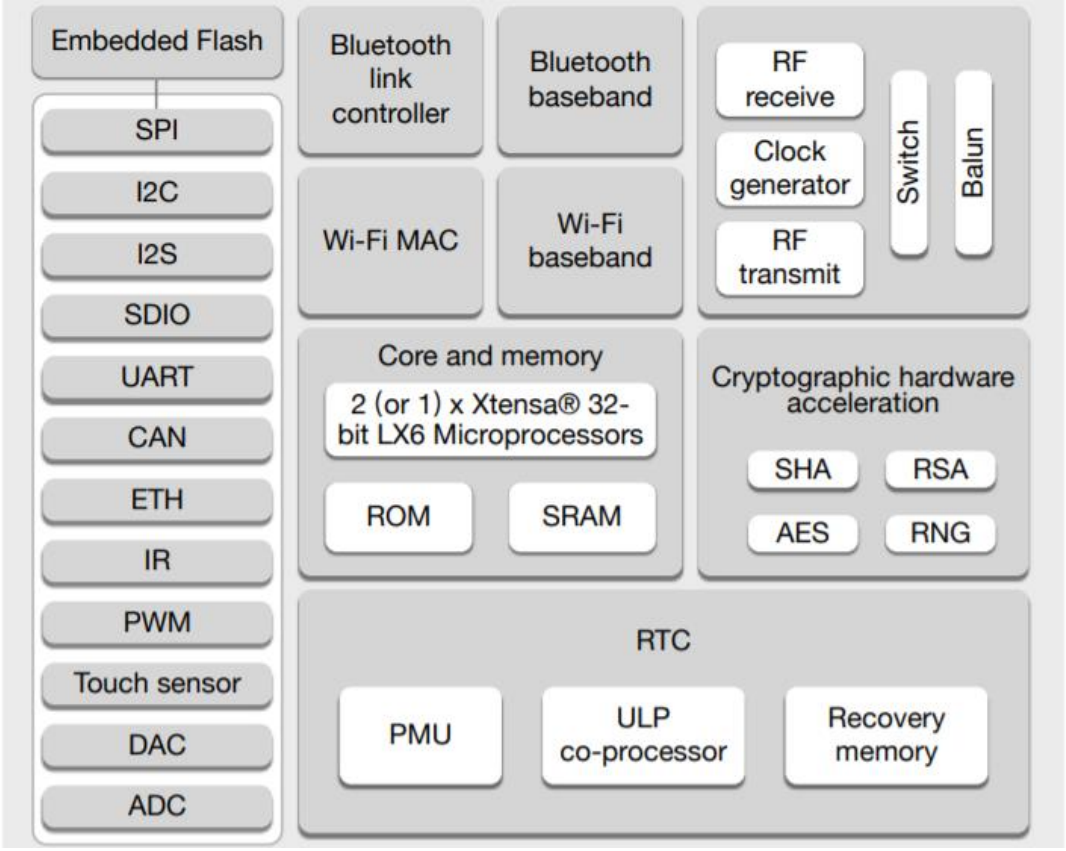


Рисунок 2.7 – Функціональна блок-схема процесора ESP32 WROOM

Центральний процесор містить два ядра, якими можна керувати індивідуально, це дозволить здійснювати асинхронне виконання різних функцій. Тактова частота ЦП регулюється від 80 МГц до 240 МГц.

Проектуючи схему підключень пристроїв до мікроконтролера та для його подальшого програмування в середовищі розробки, необхідно враховувати розпінування мікроконтролера. Схему розпінування мікроконтролера ESP32 WROOM проілюстровано на рисунку 2.8.

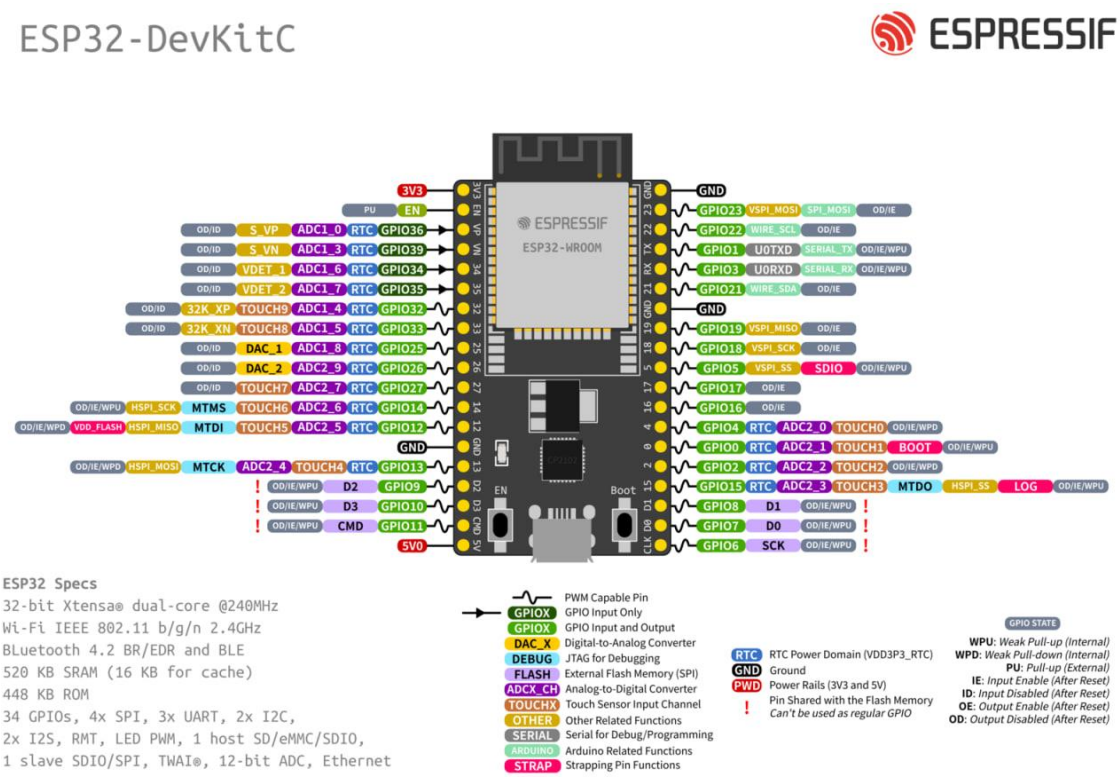


Рисунок 2.8 – Розпінування мікроконтролера ESP32 WROOM

До основних пінів мікроконтролера ESP32 WROOM входять наступні піни та їх функціональність [3]:

– GPIO (General Purpose Input/Output) піни: для підключення та керування зовнішніми пристроями, такими як давачі, актуатори та інші периферійні пристрої.

– UART (Universal Asynchronous Receiver/Transmitter): для послідовної комунікації з іншими пристроями чи мікроконтролерами за допомогою протоколу UART.

– SPI (Serial Peripheral Interface): для здійснення швидкого обміну даними зі зовнішніми пристроями, такими як флеш-пам'ять, сенсори, дисплеї.

– I2C (Inter-Integrated Circuit): для забезпечення зв'язку із периферійними пристроями за допомогою двозв'язкового протоколу I2C.

– ADC (Analog-to-Digital Converter): для перетворення аналогових сигналів в цифровий.

– DAC (Digital-to-Analog Converter): для перетворення цифрових сигналів в аналоговий.

– PWM (Pulse Width Modulation): для генерації сигналів з різною шириною імпульсів. Це корисно при керуванні яскравістю світлодіодів, або швидкістю моторів.

2.2.2 Давач температури та вологості DHT11

Давач температури і вологості DHT11 (рис. 2.9) є доступним пристроєм, який здатний вимірювати температуру та відносну вологість повітря. Основні характеристики давача DHT11 [4]:

– діапазон вимірювання температури: від -20°C до +60°C з точністю ±2°C.

– діапазон вимірювання вологості: від 5% до 95% з точністю ±5%.

– напруга живлення: датчик працює з напругою живлення 3.3-5.5 В.

– частота отримання даних: не більше 1Гц.

DHT11 має вбудований 8-бітний АЦП, який здійснює перетворення аналогових значень температури і вологості та передає результати через цифровий інтерфейс. Для зчитування даних з давача будемо використовувати бібліотеку DHT.h.

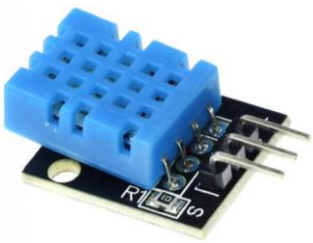


Рисунок 2.9 – Давач температури і вологості DHT11

Для підключення DHT11 містить 3 виводи: VCC (живлення), GND (земля) та Data (дані).

DHT11 є простим, надійним та недорогим у використанні пристроєм для вимірювання температури та вологості.

2.2.3 OLED-дисплей SSD1306

OLED-дисплей SSD1306 (рис. 2.10) - це компактний і високоякісний дисплей на основі OLED-технології (Organic Light-Emitting Diode). Він забезпечує яскраве та контрастне відображення зображень і тексту. Основні характеристики SSD1306 [5]:

– розмір екрану 0.96 дюймів;

– роздільна здатність матриці дисплею128x64;

– спосіб зв'язку можливий через I2C та SPI інтерфейси;

– монохромне відображення у чорно-білому кольоровому варіанті;

– висока яскравість та чіткість.

Однією з ключових переваг OLED-дисплею SSD1306 є його низьке споживання енергії, що дозволяє помірно використовувати електроенергію в пристроях з обмеженим джерелом живлення. Має широкий кут огляду (близько 160°), який дозволяє зручно зчитувати вміст дисплею з різних положень.

Для роботи з OLED-дисплеєм SSD1306 використовується спеціальна бібліотека (Adafruit_SSD1306.h), яка надає зручний інтерфейс для відображення тексту на дисплеї.

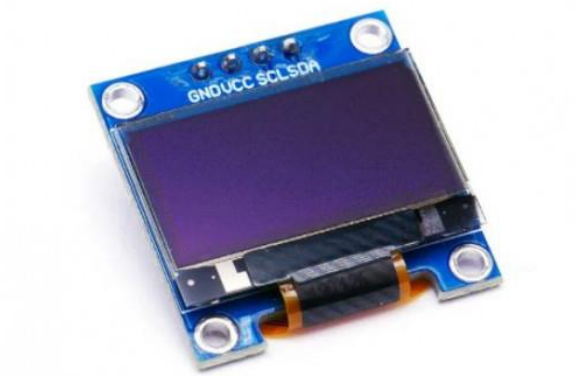


Рисунок 2.10 – Вигляд OLED-дисплея SSD1306

В комп’ютеризованій системі на даному дисплеї відображатиметься вологість і температура всередині приміщення та ззовні, а також меню керування та наявні помилки, які можуть виникнути в процесі роботи системи.

2.2.4 Ємнісна сенсорна кнопка TTP223

Ємнісна кнопка дотику TTP223 (рис. 2.11) – це сенсорна кнопка, яка реагує на дотик пальця та подає на вихід логічний сигнал. Основні характеристики сенсорної кнопки TTP223 [6]:

– плата мікросхеми TTP223;

– напруга живлення від 2.0 до 5.5В;

– час реакції на дотик в активному режимі 60мс;

– максимальна відстань спрацювання сенсора 3мм;

– час реакції на дотик в режимі енергозбереження 220мс;

– розміри модуля 14x11x2 мм.

Рисунок 2.11 – Вигляд ємнісної кнопки дотику TTP223

Модуль TTP223 містить 2 перемикачі A та B, за допомогою яких можна задати наступні параметри сенсора:

– A(розімкнений): високий логічний рівень на виході;

– A(замкнений): низький логічний рівень на виході;

– B(розімкнений): режим спрацювання як в кнопки;

– B(замкнений): режим спрацювання як в перемикача.

TTP223 є хорошим енергоефективним рішенням для розроблюваної системи. Підключення даного модуля здійснюється 3 виводами: VCC (живлення), GND (земля) та I/O (логічний вихід).

2.2.5 Реле модуль AOC456

Модуль AOC456 (рис. 2.12) – це одноканальний 5В-ий реле модуль, для розподілення електричного навантаження. Він застосовується для активації електродвигунів. Даний модуль активується низьким рівнем напруги (low level) та потребує 5-20мА для спрацювання [7].

Рисунок 2.12 – Вигляд реле модуля AOC456

Для підключення AOC456 використовуються 3 виводи: VCC (живлення), GND (земля) та IN (логічний вхід).

2.2.6 Конектор PBD-2.54-2x20

Конектор PBD-2.54-2x20 (рис. 2.13) – використовується для роз'ємного з'єднання між ESP32 та іншими модулями розроблюваної системи. Містить два ряди по 20 пінів у кожному.

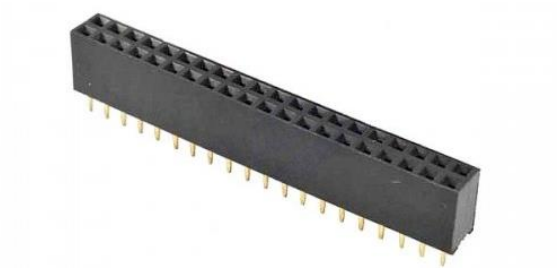


Рисунок 2.13 – Вигляд конектора PBD-2.54-2x20

2.3 Обґрунтування вибору програмного забезпечення для комп’ютеризованої системи віддаленого керування альтернативним охолодженням приміщень

2.3.1 Інтегроване середовище розробки Arduino IDE для програмування однойменних плат

Для програмування мікроконтролера ESP32 WROOM будемо використовувати ICP Arduino IDE. Дане середовище дозволяє програмувати широкий спектр мікроконтролерів Arduino та ESP, а також відображає текстову консоль, яка дуже корисна під час відлагодження мікроконтролерів. Написані за допомогою Arduino IDE програми називають скетчами. Вони мають розширення .ino, а мова програмного коду C, C++.

Інтерфейс ICP Arduino IDE простий і зручний у використанні, містить усі необхідні функції і утиліти для імпортування бібліотек та компіляції програмного коду. До переваг даного середовища можна віднести наступні:

– доступність;

– зручний та зрозумілий інтерфейс;

– сумісність з усіма версіями ОС Windows;

– наявність усіх необхідних інструментів;

– функції збереження, перевірки, пошуку, експорту та заміни скетчів.

Для програмування ESP32 спершу необхідно підключити відповідне розширення. Для цього потрібно перейти в головне меню File -> Preferences та у текстовому полі “Aditional boards manager URLs” ввести наступну адресу: “https://dl.espressif.com/dl/package_esp32_index.json”. Вікно із відкритими налаштуваннями зображене на рисунку 2.14.

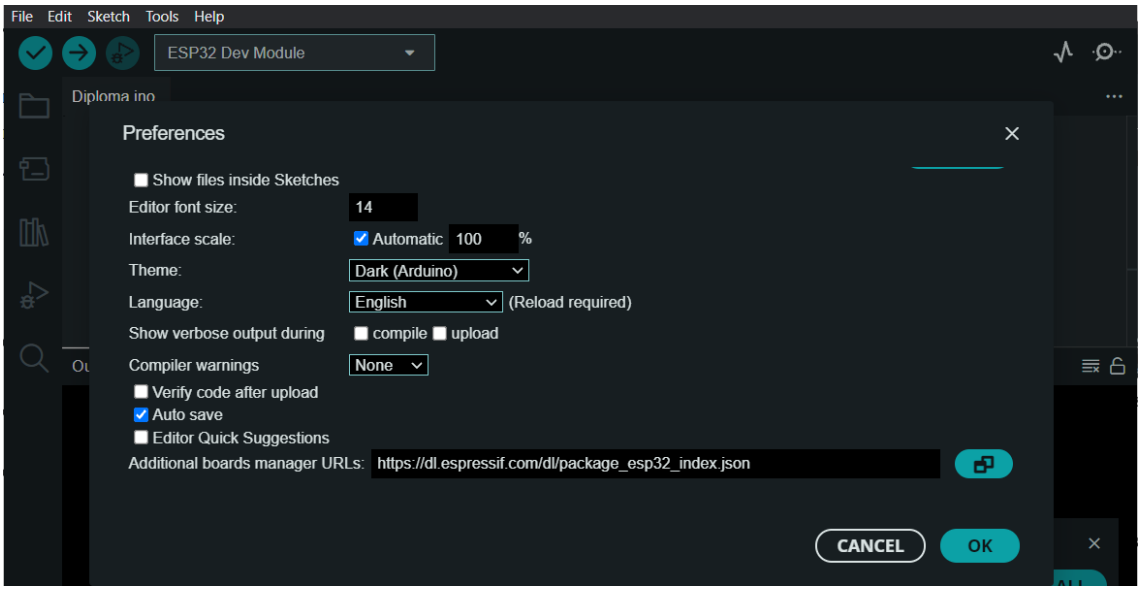


Рисунок 2.14 – Вікно налаштувань ArduinoIDE

Після підключення додаткового розширення у головному меню перейдемо до Tools -> Boards і там з'явиться випадаючий список "esp32". У даному списку необхідно обрати тип плати, для програмування (рис. 2.15).

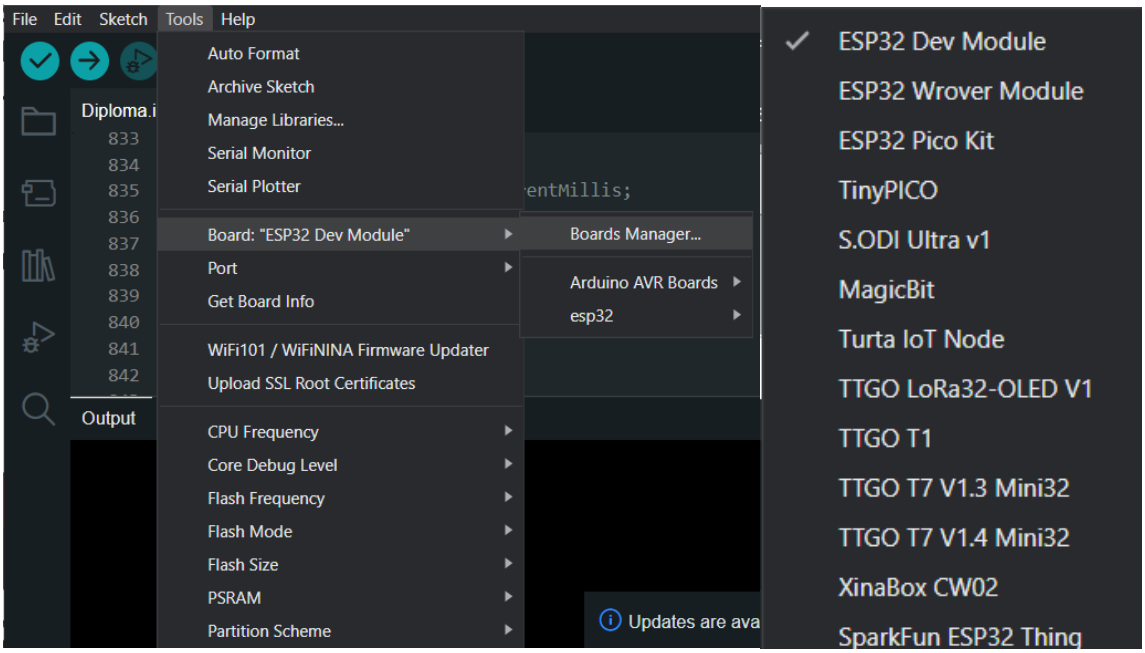


Рисунок 2.15 – Вікно вибору типу плати

Цими налаштуваннями підготовлено проєкт для подальшого написання та відлагодження коду для мікроконтролера ESP32.

2.3.2 Інтегроване середовище розробки Visual Studio Code для створення веб-застосунків

Visual Studio Code – це спрощений але потужний редактор коду, який дозволяє розробляти веб-застосунки та програми для хмарних систем. Даний редактор є безкоштовним та підтримується такими ОС як Windows, Linux та Mac OS. Головною особливістю Visual Studio Code є його велика бібліотека розширень, які дозволяють вносити додаткові функції до наявних. Вона підтримує Node.js, JavaScript, TypeScript і має велику екосистему розширень для інших мов програмування (наприклад, C#, C++, Java, Python, Go і PHP) та середовищ виконання (наприклад, .NET і Unity) [8].

Для реалізації клієнт-серверної частини розроблюваної комп'ютеризованої системи будемо використовувати бібліотеку компонентів React.js, веб-феймворк Express.js та платформу для проектування та виконання мережевих додатків Node.js. Для того аби працювати з ними в Visual Studio Code, спершу інсталюємо Node.js на комп'ютер, після чого з'явиться пакетний менеджер NPM. Яким можна оперувати в командному вікні ОС. Дана утиліта має доступ до усіх доступних пакетів та модулів, які розміщені у хмарному сховищі NPM. Тож розробник має можливість підтягнути будь-який із них для покращення та пришвидшення розробки програмного забезпечення. Пакетний менеджер під час роботи оперується файлом "package.json", в якому зберігається інформація про пакети проєкту, їхню версію та стан. Для того аби розпочати розробку ПЗ із Node.js, необхідно ініціалізувати створений проєкт з його початковими параметрами у файл "package.json". Для цього у командному рядку Visual Studio Code, у директорії проекту, вводимо команду: "npm init".

Після закінчення базових налаштувань проєкту, перейдемо до детальнішого опису бібліотек React.js, Express.js та платформи Node.js. Виділимо їхні особливості та переваги у застосуванні для розробки ПЗ для розроблюваної комп'ютеризованої системи.

React — це декларативна JavaScript-бібліотека з відкритим вихідним кодом, призначена для створення користувацьких інтерфейсів. Розроблена компанією Meta. Вона дає змогу компонувати складні веб-інтерфейси з невеликих окремих компонентів (частин коду). Основною перевагою даної бібліотеки є те, що вона дозволяє створювати односторінкові веб-сайти, без постійного перезавантаження усієї сторінки при будь-яких змінах. Це стало можливим за допомогою віртуального DOM сторінки , який є точною копією реального DOM сторінки у браузері. Віртуальний DOM зберігається в пам’яті та синхронізується з реальним DOM. В разі фіксації змін у віртуальному DOM, вони передаються реальному DOM для оновлення конкретного компонента. Це забезпечує оптимальну та швидку роботу веб-сторінки, без її перезавантажень.

Також у React.js є файлове розширення .jsx, яке дозволяє компонувати гіпертекст HTML з кодом JavaScript. Тобто код розмітки сторінки там само, де і код компонента. Завдяки цьому процес створення компонентів став значно зручнішим та швидшим.

Не менш важливу роль при розробці з використанням React, відіграє концепція життєвого циклу компонента. З якого можна виділити основні три етапи: Mount Update та Unmount (рис. 2.16).

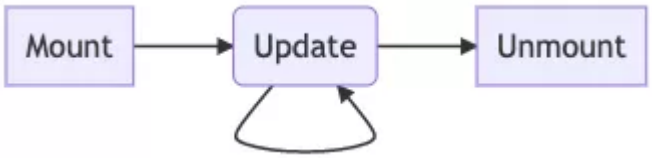


Рисунок 2.16 – Життєвий цикл компонента React

Update поділяється ще на три частини: Render, Precommit, Commit. Їх зображено на рисунку 2.17.

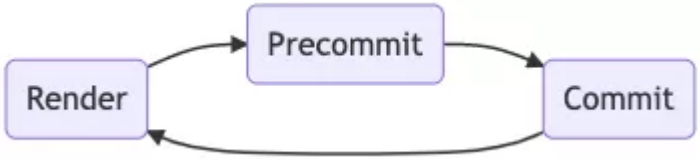


Рисунок 2.17 – Структура етапу Update

На етапі Mount (монтування) відбувається ініціалізація компонента та його стану, опісля чого він монтується в структурі DOM. На етапі Update (оновлення компонента) відбувається рендеринг (render) компонента і підготовка до внесення змін в структуру DOM (проміжний етам precommit), а тоді безпосередньо внесення змін до структури DOM (commit). На останньому етапі Unmount (демонтування) відбувається вилучення компонента зі структури DOM.

Express.js – це гнучкий фреймворк для створення серверної частини веб-застосунків. Він являється програмним каркасом серверної частини до якого легко та зручно підключаються додаткові бібліотеки. Для розробки серверної частини комп'ютеризованої системи віддаленого керування альтернативним охолодженням, буде застосовано цей фреймворк та підключено до нього різні плагіни, такі як mongoose для керування СКБД MongoDB, bcrypt для шифрування JWT токена, axios для забезпечення http комунікації та інші.

В основу серверної частини express покладено MVC (Model-View-Controller) модель архітектури. Ця модель містить три складові, які складаються з бізнес-логіки, моделі представлення та інтерфейсу відображення.

Model – це база даних елементів, яка містить структуру збереження та представлення даних. View – це інтерфейс користувача на якому відображаються усі компоненти з якими він може безпосередньо взаємодіяти. Controller – це міст між моделюю та інтерфейсом, який реагує на всі зміни внесені користувачем та відправляє їх у модель для збереження.

Node.js – це однопоточне програмне середовище виконання з відкритим кодом для запуску веб-додатків, які написані мовою JavaScript. Вона розроблена на базі рушія V8, компанією Google. Головною превагою даної платформи є те, що JavaScript код можна запустити практично у будь-якому середовищі. Node.js дає змогу створювати як frontend так і backend частини веб-додатку. Тобто, повноцінні веб-сайти можуть працювати використовуючи єдиний “стек”. А це робить розробку та обслуговування значно легшими та швидшими.

Оскільки Node.js має відкритий вихідний код, працювати в ньому можна безкоштовно. Основними перевагами цього середовища є те, що він є гнучким і простим у використанні, легко розгортається у проєкті, а весь його функціонал дозволяє оптимізувати та пришвидшити роботу проєкту.

Основні характеристики Node.js:

– Реалізований мовою JavaScript, яка являється найпоширенішою мовою програмування у світі. Це робить Node.js простим та зрозумілим для вивчення.

– Асинхронність: сервер написаний на Node.js, може опрацьовувати декілька різних процесів паралельно, не блокуючи один одного. Ця властивість робить його дуже швидким і забезпечує кращий інтерфейс користувача.

– Однопотокове опрацювання запитів: робота починається з отримання запиту і закінчується виконанням необхідного завдання із надсиланням відповіді клієнту назад. Це запобігає повторному завантаженню запитів і зменшує час їхнього опрацювання.

– Сумісність з різними платформами: можна використовувати на різних ОС, таких як Windows, Mac OS та Linux.

– Швидка потокова передача даних: працює на базі рушія виконання JavaScript V8. Що значно прискорює роботу середовища та забезпечує дуже швидку потокову передачу даних для веб-застосунків.

2.3.3 СКБД ModgoDB

ModgoDB – це документо-орієнтована система керування базами даних. Її використовують для швидкого створення додатків та стартапів з особливою бізнес-логікою [11]. ModgoDB створена у 2007 році розробниками Google. Дану СКБД використовують Adobe, eBay, Squarespace Version та інші великі корпорації.

На відміну від реляційних СКБД де інформація жорстко пов'язана та структурована таблицями, ModgoDB (рис. 2.18) зберігає дані в колекціях у вигляді JSON об'єктів, у яких також є ієрархія, але не така жорстка. Основною перевагою ModgoDB є його масштабованість, що дозволяє доповнювати об'єкти за необхідності. Для прикладу, якщо у користувача не одна email адреса, а декілька, то додаткове поле можна додати цьому користувачу без підключення додаткових розширень чи інших допрацювань. ModgoDB чудово підходить для даних, які змінюються з часом.

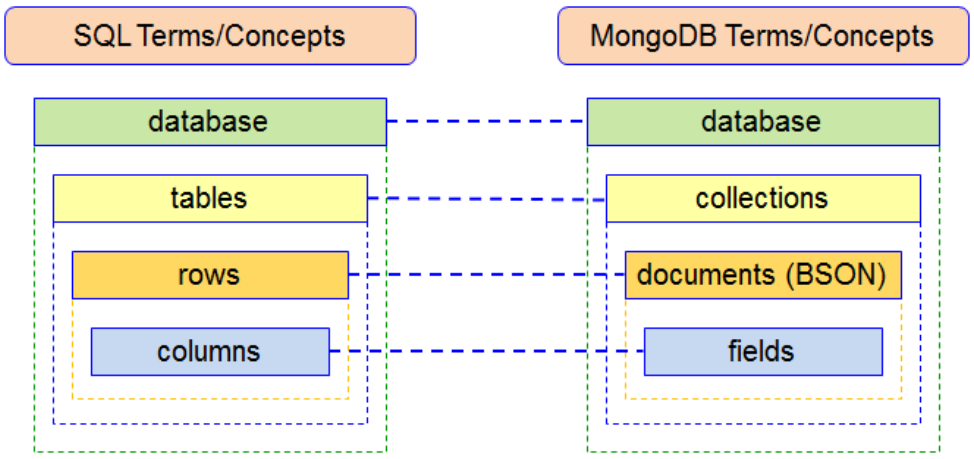


Рисунок 2.18 – Порівняння SQL з MongoBD

Для розроблюваної комп'ютеризованої системи створено в MongoDB три колекції для зберігання даних користувача (колекція users), кімнати (колекція rooms) та всі додані пристрої (колекція controllers).

2.4 Проєктування комп’ютерного засобу

Для проєктування комп’ютерного засобу скористаємось Wokwi - безплатним сервісом для прототипування та відлагодження вбудованих проектів Arduino та ESP32. Цей сервіс не потребує інсталяції, оскільки проектувати схеми можна одразу в браузері. Вигляд інтерфейсу Wokwi зображений на рисунку 2.19.

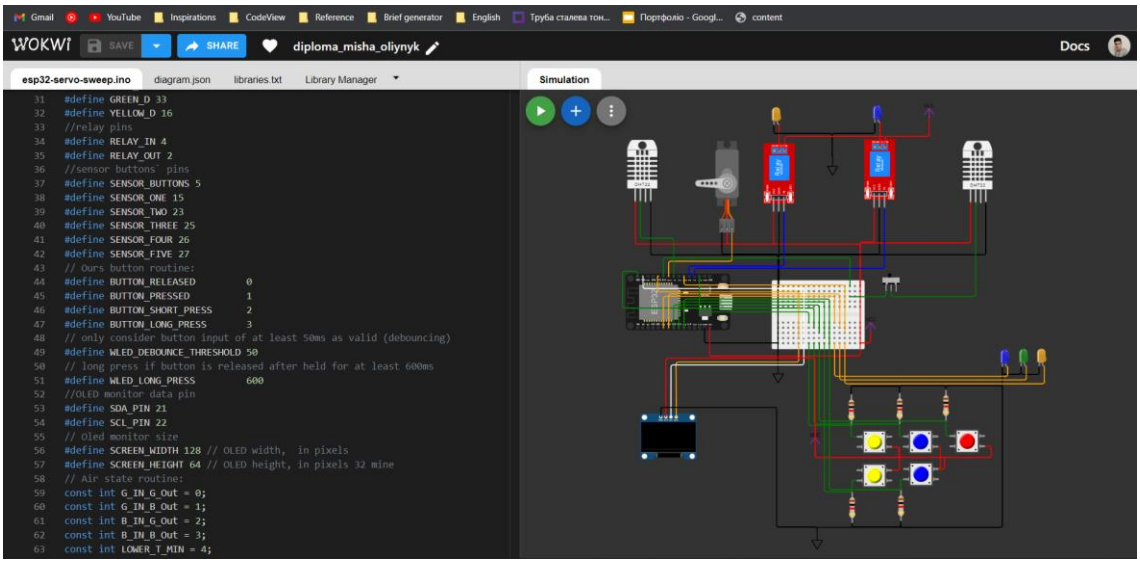


Рисунок 2.19 – Вигляд інтерфейсу симулятора Wokwi

Його інтерфейс доволі простий та зрозумілий. Екран поділений на дві вертикальні частини. Зліва знаходиться текстовий редактор програмного коду, а праворуч вікно конструктора для додавання компонентів та утворення зв’язків між ними. Також інструментарій Wokwi включає менеджера бібліотек, відображення підключених бібліотек та JSON файлу структури проєкту.

Розробимо комп’ютерну модель розроблювальної системи (рис. 2.20). Побудована модель дозволить написати та протестувати програмний код. По завершенні тестувань якого, безпечно перенести його в фізичний мікроконтролер ESP32, дозволяючи, уникнути можливих некоректних з’єднань та програмних помилок, в результаті яких може вийти з ладу фізична ESP32.

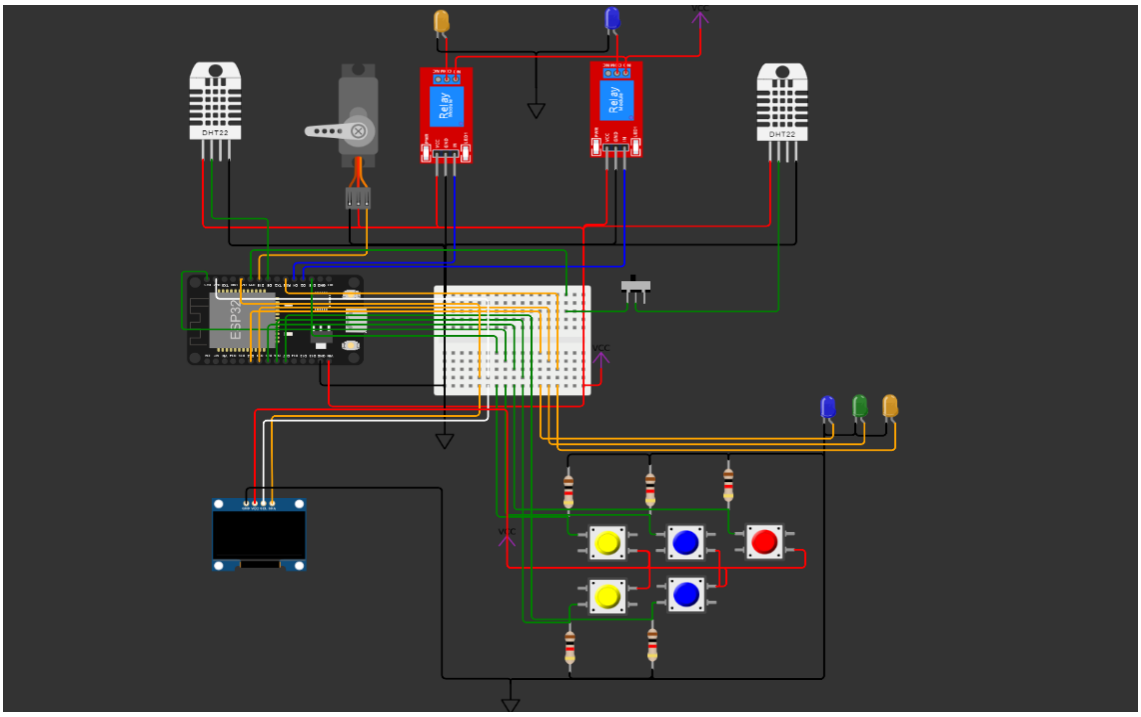


Рисунок 2.20 – Модель КС в середовищі Wokwi

Опісля створеної моделі ініціалізуємо необхідні бібліотеки. Для цього в розділі менеджера бібліотек (рис. 2.21) знайдемо їх та підключимо до проєкту.

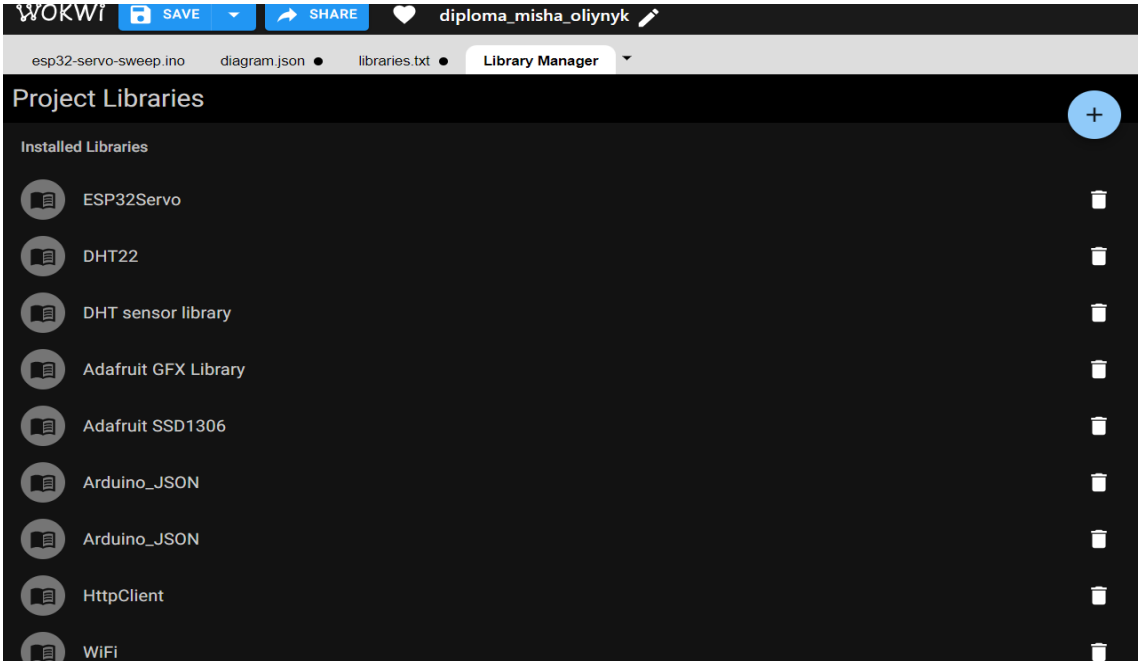


Рисунок 2.21 – Список підключених бібліотек у Wokwi

Тепер можна переходити до практичної частини, а саме написанню ПЗ.

РОЗДІЛ 3 ПРАКТИЧНА ЧАСТИНА

3.1 Реалізація проектних рішень програмного забезпечення для розробленої комп'ютеризованої системи

Для створення ПЗ комп'ютеризованої системи віддаленого керування альтернативним охолодженням приміщення для зберігання запасів продуктів, програмне забезпечення було поділено на три частини: ПЗ клієнтської частини, ПЗ серверної частини та ПЗ головного керуючого пристрою комп'ютеризованої системи: мікроконтролера ESP32 WROOM.

3.1.1 Реалізація серверної частини комп'ютеризованої системи

Моделлю архітектури веб-сервера є MVC. Спершу після ініціалізації npm, додаємо всі необхідні розширення та бібліотеки. Їхній список зображений на рисунку 3.1.

```
12  "dependencies": {
13    "axios": "^1.3.5",
14    "bcrypt": "^5.1.0",
15    "cookie-parser": "^1.4.6",
16    "cors": "^2.8.5",
17    "date-fns": "^2.29.3",
18    "dotenv": "^16.0.3",
19    "express": "^4.18.2",
20    "jsonwebtoken": "^9.0.0",
21    "mongoose": "^7.0.3",
22    "uuid": "^9.0.0"
23  },
24  "devDependencies": {
25    "nodemon": "^2.0.22"
26  }
```

Рисунок 3.1 – Список розширень проєкту

Після підключення розширень перейдемо до створення каталогів. Було створено наступні папки (рис. 3.2).

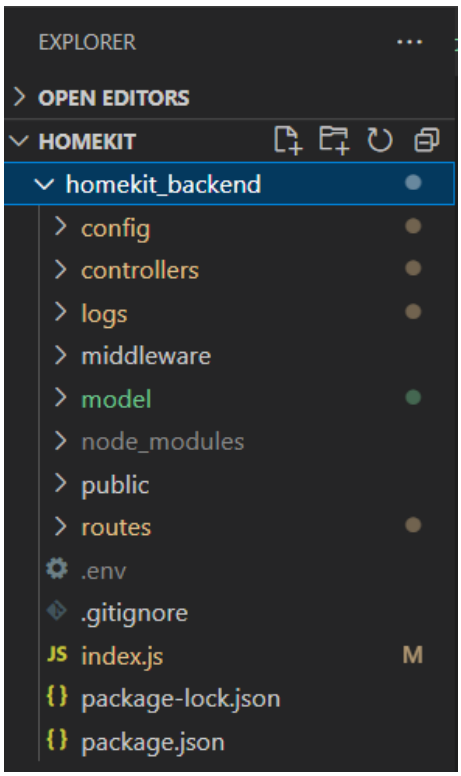


Рисунок 3.2 – Список каталогів проєкту Node.js

У каталозі config знаходяться конфігурації сервера (рис. 3.3), а саме список дозволених ip-адрес, параметри cors, функція синхронізації з базою даних та список ролей користувача.

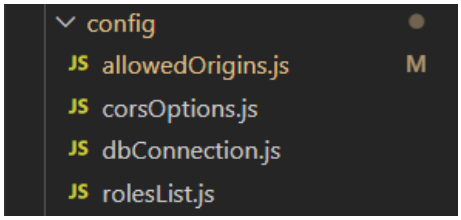


Рисунок 3.3 – Список конфігураційних файлів

Наступним каталогом є controllers. В ньому розміщені всі наявні контролери сервера, які запускаються в разі отримання відповідного запиту до конкретного із них. Список усіх контролерів зображений на рисунку 3.4.

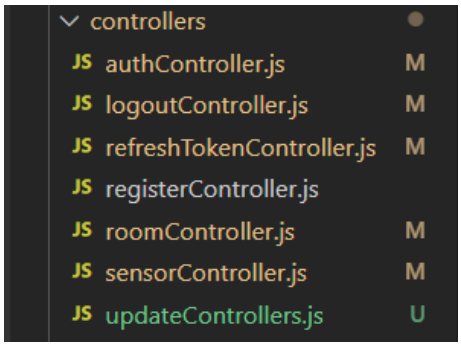


Рисунок 3.4 – Список контролерів сервера

В каталозі logs знаходяться файли логування, які зберігають інформацію про запити, які були отримані сервером та помилки під час їхнього опрацювання. Файли логування досить корисні при тестуванні та налагодженні сервера. Самі функції логування знаходяться в каталозі middleware.

Каталог middleware містить файли проміжного виконання (рис. 3.5), саме тому у нього така назва. Оскільки сервер опрацьовує потоки, проміжні файли розміщуються в порядку їхньої необхідності. Тобто при отриманні запиту від клієнтської частини спочатку він пройде middleware logEvents, а опісля middleware credentials з перевіркою конфігурацій cors. Після цього запит потрапляє до маршрутизації контролерів (routes). Routes поділяються на загальнодоступні та авторизованого доступу. Тому між ними розміщений middleware verifyJWT та verifyRoles, які пропускають тільки авторизовані запити з обмеженням по ролях.

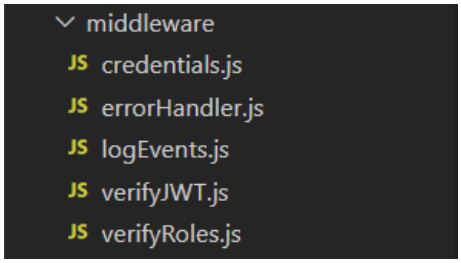


Рисунок 3.5 – Список middleware функцій

У каталозі model, знаходяться моделі представлення та збереження даних для СКБД MongoDB (рис. 3.6). Це моделі для зберігання інформації про користувачів, створених кімнат та доданих пристроїв.

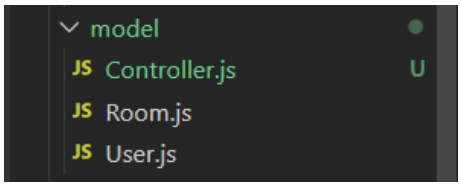


Рисунок 3.6 – Список моделей

Останній каталог routes (рис. 3.7) містить маршрути (endpoint) куди будуть направлятись запити від клієнтської частини.

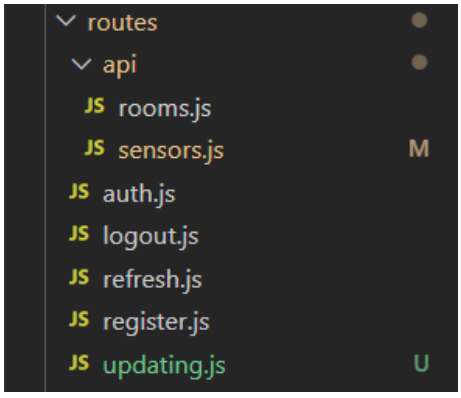


Рисунок 3.7 – Список маршрутів каталогу routes

В підкаталозі api знаходяться маршрути авторизованого доступу. Це маршрути до контролерів кімнат та пристроїв, оскільки оперувати ними можна тільки авторизованому користувачу.

Методи опрацювання запитів та їхня послідовність прописана у головному файлі index.js. У якому спершу імпортуємо, вищеописані функції та розширення (ліст. 3.1).

Лістинг 3.1 – Імпорт необхідних розширень та функцій

```
require('dotenv').config();
const express = require('express');
const app = express();
const path = require('path');
const cors = require('cors');
const corsOptions = require('./config/corsOptions');
const {logger} = require('./middleware/logEvents');
const errorHandler = require('./middleware/errorHandler');
const verifyJWT = require('./middleware/verifyJWT');
const cookieParser = require('cookie-parser');
const credentials = require('./middleware/credentials');
const mongoose = require('mongoose');
const connectDB = require('./config/dbConnection');
const PORT = process.env.PORT || 5500;
```

Наступним кроком відбувається з’єднання із базою даних MongoDB:

Лістинг 3.2 – Встановлення з’єднання між сервером та базою даних

```
//connect to MongoDB
connectDB();
```

Розпочинаємо опрацьовувати запити проміжними функціями. При надходженні на сервер запит спершу логується, де вказується його ip-адреса, час та метод звернення. Тоді він проходить перевірку на дозвіл доступу, після чого відбувається декодування url-адреси запиту. З якої витягується необхідний маршрут та дані, які було відправлено:

Лістинг 3.3 – Послідовність опрацювання запиту middleware функціями

```
//custom maiddleware logger
app.use(logger);
// Handle options credentials check before CORS!
// and fetch cookies credentials requirement
app.use(credentials);
//Cross Origin Resource Sharing
app.use(cors(corsOptions));
// built-in middleware to handle urlencoded from data
app.use(express.urlencoded({extended: false}));
//built-in middleware for json
app.use(express.json());
//middleware for cookies
app.use(cookieParser());
```

Після проміжних функцій запит переходить до вибору необхідного маршруту:

Лістинг 3.4 – Маршрути загального доступу

```
app.get('/', (req, res) => {
    res.send("Server is running now");
});
//routes
app.use('/register',require('./routes/register'));
app.use('/login',require('./routes/auth'));
app.use('/refresh',require('./routes/refresh'));
app.use('/logout',require('./routes/logout'));
app.use('/updating',require('./routes/updating'));
```

Для доступу до наступних маршрутів необхідно пройти авторизацію JWT-токена та перевірку ролей користувача:

Лістинг 3.5 – Маршрути авторизованого доступу

```
app.use(verifyJWT);
app.use('/sensors',require('./routes/api/sensors'));
app.use('/rooms',require('./routes/api/rooms'));
```

Якщо при надсиланні був вказаний невірний маршрут то він потрапляє на кінцевий маршрут сервера, який повертає відповідну помилку у відповідь:

Лістинг 3.6 – Кінцевий маршрут для некоректних запитів

```
app.all('*',(req, res) => {
    res.status(404);
    if(req.accepts('html')){
        res.send("anything in there");
    }else if(req.accepts('json')){
        res.json({error: "404 Not found"});
    } else {
        res.type('txt').send("404 Not found");
    }
});
```

Якщо в процесі виникли помилки, вони виявляються за допомогою middleware errorHandler та логуються у log-файл:

Лістинг 3.7 – Логування помилок в процесі опрацювання запитів

```
app.use(errorHandler);
```

Якщо зв’язок із базою даних присутній то порт сервера буде відкритим та очікувати на запити від клієнтської частини. Дана перевірка здійснюється один раз при запуску сервера:

Лістинг 3.8 – Перевірка з’єднання із базою даних

```
mongoose.connection.once('open', () => {
    console.log('Connected to MongoDB');
    app.listen(PORT, () => console.log(`Server running on port
${PORT}`));
});
```

Сценарій де користувач заходить на свою персональну сторінку:

– Користувач вводить свої персональні дані, такі як ім’я та пароль і натискає кнопку увійти.

– Клієнтська частина сформує та відправить запит до сервера на маршрут /login. В тілі запиту закріпить json-об’єкт зі вказаними параметрами імені та паролю.

– Коли сервер бачить запит, спершу він перевіряє дозвіл на доступ ip-адреси відправника. Це перевірка cors із переглядом списку дозволених ip-адрес із файлу конфігурацій.

– Якщо запит отримав дозвіл на доступ, його url-адреса розшифровується. З шифру отримується маршрут на який надіслано запит, тип запиту та дані, які було розміщено у тіло запиту.

– Зі списку наявних маршрутів обирається маршрут /login та запускається контролер handleLogin, оскільки він закріплений за даним маршрутом. Даний контролер надсилає модель користувача до бази даних для пошуку користувача. Якщо користувача було знайдено то у відповідь клієнтській частині, контролер відправить ролі користувача та його JWT-токени доступу.

3.1.2 Реалізація клієнтської частини комп’ютеризованої системи

Для ініціалізації проєкту React, в консолі вікна Visual Studio Code, потрібно ввести наступну команду: “npm init react-app homekit”. Після якої додати декілька розширень, а саме axios, web-vitals та react-tranzition-group. Список усіх розширень зображений на рисунку 3.8.

```
"dependencies": {
  "@testing-library/jest-dom": "^5.16.5",
  "@testing-library/react": "^13.4.0",
  "@testing-library/user-event": "^13.5.0",
  "axios": "^1.3.5",
  "react": "^18.2.0",
  "react-dom": "^18.2.0",
  "react-router-dom": "^6.10.0",
  "react-scripts": "5.0.1",
  "react-transition-group": "^4.4.5",
  "web-vitals": "^2.1.4"
},
```

Рисунок 3.8 – Список розширень проєкту в React

Після ініціалізації проєкту створено наступні каталоги:

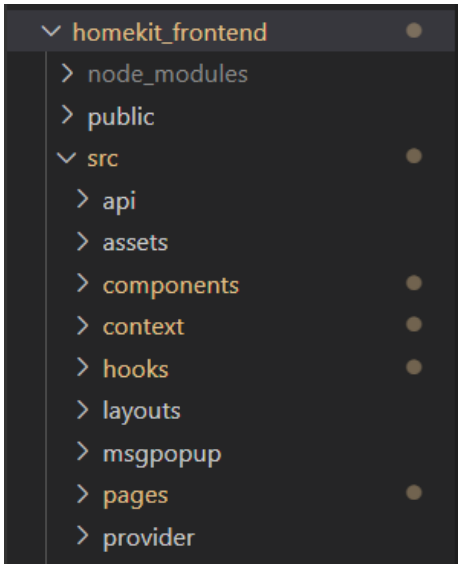


Рисунок 3.9 – Список каталогів проєкту в React

В каталозі api знаходиться базова функція створення http зв’язку axios. А в каталозі assets, зберігаються всі іконки та ілюстрації. Вміст даних каталогів зображений на рисунку 3.10.

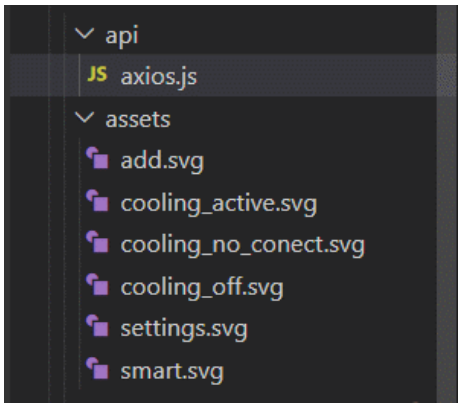


Рисунок 3.10 – Вміст каталогів api та assets

Наступним є каталог компонентів: “components” (рис. 3.11). Він містить підкаталоги усіх створених компонентів, які використовуються для взаємодії із користувачем. Це компоненти реєстрації, входу, налаштування кімнат та пристроїв, додавання кімнат та пристроїв, відображення та видалення їх.

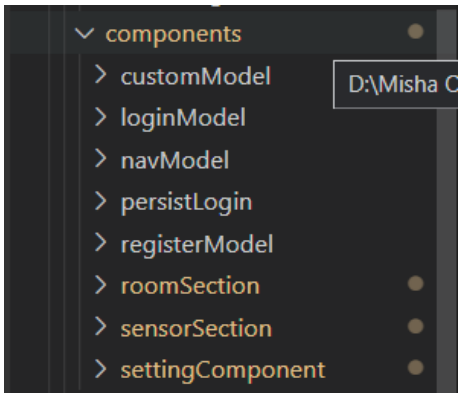


Рисунок 3.11 – Вміст каталогу components

Наступними каталогами є context, hooks та layouts (рис. 3.12). В яких розміщуються капсули (обгортки).

PersistLogin – це обгортка, яка при кожному перезавантаженні сторінки утримує користувача на тій самій сторінці де він знаходився в момент її

перезавантаження. В каталозі hooks знаходяться акцептори перевірки валідності JWT токена через, які проходять авторизовані запити на сервер. В каталозі layouts знаходяться шари відображення вмісту сторінок каталогу pages.

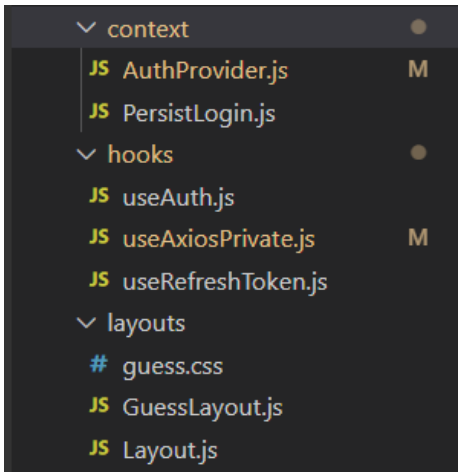


Рисунок 3.12 – Вміст каталогів context, hooks та layouts

В каталозі msgpopup знаходяться функції відображення спливаючих повідомлень. В каталозі pages (рис. 3.13), знаходяться основні сторінки веб-сайту. Це сторінки авторизації, головна сторінка користувача та сторінка сповіщення помилки 404.

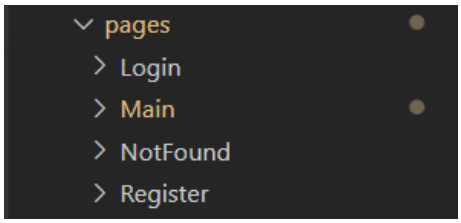


Рисунок 3.13 – Вміст каталогу pages

В каталозі provider, розміщена обгортка доступу для авторизованих користувачів RequireAuth. Даний компонент надає доступ тільки авторизованому користувачу.

Оглянемо вміст головного файлу клієнтської частини index.js (ліст. 3.9) та розглянемо послідовність його роботи.

Лістинг 3.9 – Вміст файлу index.js

```
import React from "react";
import { createRoot } from "react-dom/client";
import "./index.css";
import App from "./App";
import { BrowserRouter, Routes, Route, Navigate } from "react-
router-dom";
import { AuthProvider } from "./context/AuthProvider";
const container = document.getElementById("root");
const root = createRoot(container);
root.render(
  <BrowserRouter>
    <AuthProvider>
      <Routes>
        <Route path="*" element={<App />} />
      </Routes>
    </AuthProvider>
  </BrowserRouter>);
```

В лістингу 3.9, спершу відбувається імпортування необхідних бібліотек, оголошення рендерингу. Всередині якого розміщені базові шари для роботи маршрутів та обгортка AuthProvider, яка задає видимість глобальних змінних на весь компонент App. App – це головний компонент відображення всіх сторінок згідно обраних маршрутів (ліст. 3.10).

Лістинг 3.10 – Структура маршрутів компонента App

```
<Routes>
  <Route path="/" element={<Layout />}>
    <Route element={<GuessLayout />}>
      <Route path="login" element={<Login />} />
      <Route path="register" element={<Register />} />
    </Route>
    <Route element={<PersistLogin/>}>
      <Route element={<RequireAuth allowedRoles = {[5001, 4001,
3001]}/>}>
        <Route path="main" element={<Main />} />
      </Route>
    </Route>
    <Route path="*" element={<NotFound />} />
  </Route>
</Routes>
```

За допомогою Routes, відбувається навігація (маршрутизація) по сторінках клієнтського веб-сайту. Маршрути які огорнуті, вищеописаним RequireAuth provider, мають авторизований доступ. Та впускають користувачів які мають JWT

токен та перелік ролей. Маршрути сторінок авторизації є загальнодоступними. Останній маршрут із шляхом “*” спрацьовує при вказанні невірного шляху маршрутизації. Це може виникнути в разі навігації через URL-панель браузера, де можна випадково ввести невірну адресу.

Коди всіх компонентів клієнтської частини наведені в додатку Е.

3.1.3 Реалізація ПЗ для апаратної частини комп’ютеризованої системи

Програмний код для мікроконтролера ESP32 WROOM було реалізовано згідно парадигми ООП. Тобто весь код розбитий на функціональні об’єкти, кожен з яких виконує конкретні функції. Всі створені функції зображені у лістингу 3.11.

Лістинг 3.11 – Перелік усіх функцій програми для роботи ESP32 WROOM

```
void setAirValve(bool state)
void setCoolerSystem(bool state)
bool isSensorPressed(uint8_t b)
void showSensorStatus(int sensorStatus)
void initSensors()
void pollSensors(int n)
int getLastPolledSensorStatus(uint8_t btn)
int getLastPolledTwoSensorStatus(uint8_t btn1, uint8_t btn2)
void displayData(String str)
int getSensorData()
int setAirStateProtocol()
int analyzeAirStateProtocol(unsigned long current_time)
void settingsMenu()
void displayAirData(float in_T, int in_H, float out_T, int out_H)
String httpGETRequest()
void jsonParsing(String str)
```

Дані функції взаємодіють одна з одною. Це відбувається через передачу параметрів при їхньому виклику, або при повернені даних після виконання їхньої роботи.

Призначення функцій розробленої комп’ютеризованої системи:

– setAirValve(state): контролює повітряний клапан;

– setCoolerSystem(state): контролює систему вентиляції;

– isSensorPressed(b): перевіряє сенсори на наявність дотику;

– showSensorStatus(sensorStatus): відображає стан активації сенсорних кнопок;

– initSensors(): ініціалізує порти до яких підключені сенсорні кнопки;

– pollSensors(n): аналізує кожну підключену сенсорну кнопку та декларує її стан;

– getLastPolledSensorStatus(btn): повертає попереднє значення стану сенсорної кнопки;

– getLastPolledTwoSensorStatus(btn1,btn2): повертає попереднє значення стану двох сенсорних кнопок;

– displayData(String str): відображає текстовий рядок на OLED-дисплей;

– getSensorData(): отримує значення з давачів температури та вологості;

– setAirStateProtocol(): повертає протокол стану температурних умов згідно встановлених параметрів системи;

– analyzeAirStateProtocol(current_time): аналізує отриманий протокол та контролює усією системою вентиляції згідно налаштованих умов.

– settingsMenu() - налаштування системи;

– displayAirData(in_T, in_H, out_T, out_H): відображає поточний стан температури та вологості;

– httpGETRequest(): встановлює зв’язок та обмін даними із сервером;

– jsonParsing(String str): декодує JSON-об’єкт та аналізує його вміст.

Головною функцією керування системою вентиляції даного пристрою є analyzeAirStateProtocol. Вона має особливий алгоритм роботи який дозволяє підтримувати оптимальний стан температури всередині складського приміщення. Блок-схема функції analyzeAirStateProtocol зображена на рисунку 3.14.

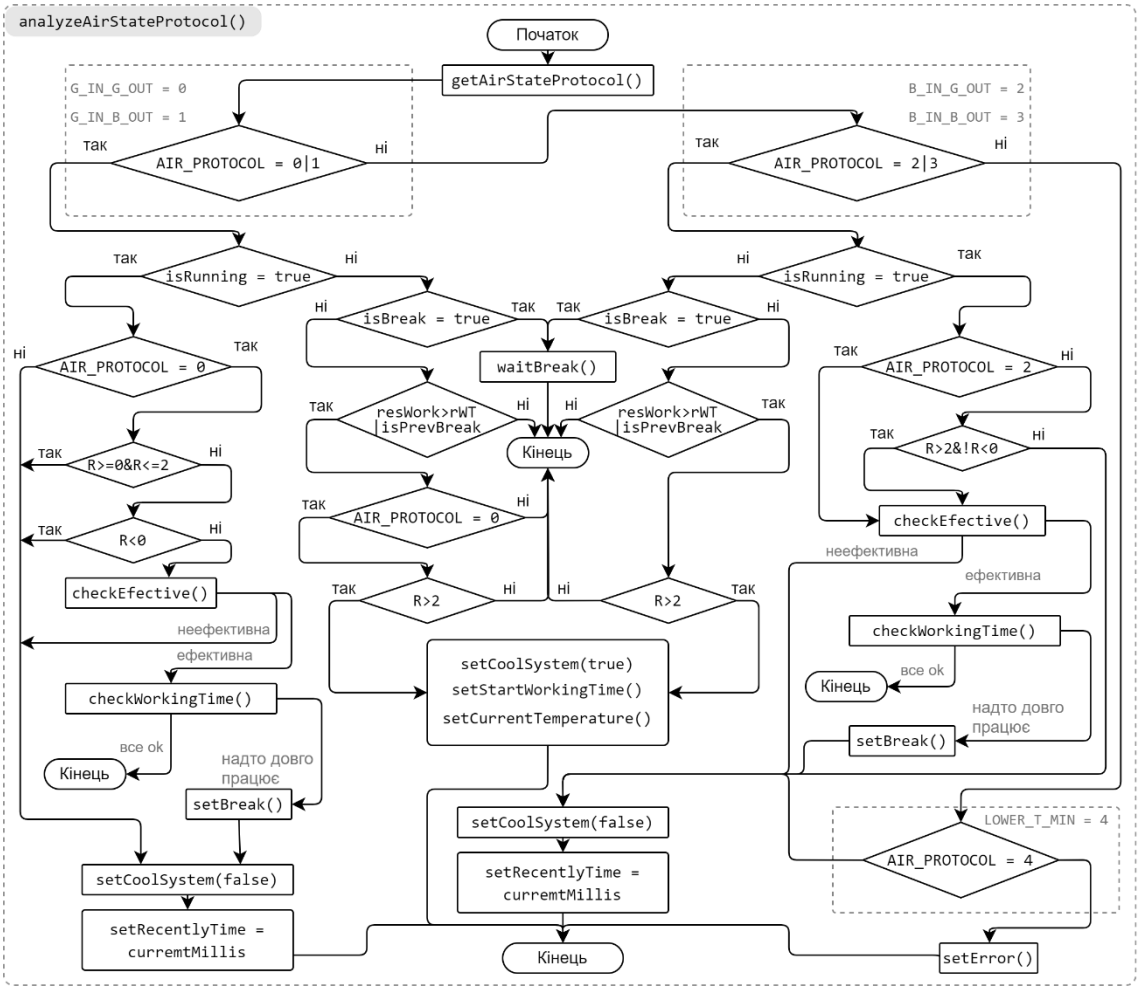


Рисунок 3.14 – Блок-схема функції analyzeAirStateProtocol

Принцип її роботи наступний, спершу відбувається звернення до функції setAirStateProtocol, яка повертає протокол стану повітря. Є п’ять варіантів даного протоколу, а саме G_IN_G_OUT (температура зовні та всередині в межах норми), G_IN_B_OUT (температура всередині в межах норми, зовні - ні), B_IN_G_OUT (температура зовні в межах норми, всередині - ні), B_IN_B_OUT (температура зовні та всередині не в межах норми), LOWER_T_MIN (температура всередині нижче мінімальної). Після отримання протоколу стану повітря, відбувається перехід в ділянку G_IN, B_IN або LOWER_T_MIN в залежності від отриманого протоколу.

На ділянках G_IN та B_IN, спершу відбувається перевірка чи система охолодження активна, а тоді перевіряється чому система активна, а чому ні. На ділянці G_IN, в разі активності системи здійснюється уточнення протоколу, а саме чи рівний він G_IN_G_OUT. Якщо ні – це означає, що температура всередині в

межах норми, зовні – ні. В такому разі негайно вимикається система вентиляції, перекривається повітряний клапан та завершується робота функції. Якщо протокол рівний G_IN_G_OUT, то температура зовні та всередині в межах норми. В такому разі здійснюється перевірка різниці між зовнішньою та внутрішньою температурами повітря. Якщо різниця в межах 2-ох градусів то систему можна вимикати, якщо різниця більше ніж 2 градуси здійснюється перехід до функції аналізу продуктивності системи. Якщо система працює і стан повітря всередині покращується то вона ефективна, в противному випадку ні. Якщо здійснюється ефективне охолодження то перевіряється час роботи системи вентиляції. В разі його досягнення система вимикає вентиляцію та переходить в режим перерви опісля чого функція завершує роботу.

На ділянці G_IN, в разі неактивності системи, перевіряється чи не стоїть режим перерви. Якщо режим перерви активований то функція завершує роботу. В разі завершення часу перерви або часу затримки після попереднього завершення роботи, уточняється протокол стану повітря. Якщо протокол G_IN_B_OUT, то функція завершує свою роботу. Якщо протокол G_IN_G_OUT, здійснюється перевірка різниці між зовнішньою та внутрішньою температурами повітря. Якщо різниця в межах 2-ох градусів то функція завершує роботу, якщо різниця більше ніж 2 градуси здійснюється активація системи охолодження і декларація температури та часу на момент її активації.

На ділянці B_IN, здійснюються подібні маніпуляції що й у G_IN. Основна відмінність полягає у функції обрахунку продуктивності. В разі неефективності системи при протоколі B_IN_G_OUT, здійснюється логічний вивід “1” на реле напруги, яке підвищує напругу на вході системи вентиляції. Це здійснюється для підвищення потужності системи провітрювання.

Весь код для мікроконтролера ESP32 WROOM наведений у додатку Д.

3.2 Моделювання апаратної частини комп’ютеризованої системи

На основі розробленої комп’ютерної моделі системи у середовищі Wokwi, було спроектовано фізичну модель системи. Оскільки фізична модель мікроконтролера ESP32 WROOM відрізняється від моделі ESP32 середовища Wokwi, схема з’єднань буде дещо відрізнятись. Схема електрична принципова комп’ютеризованої системи віддаленого керування альтернативним охолодженням, зображена у додатку В.

Після проведення вимірювань приміщення було обчислено його об’єм (рис. 3.15). Він складає 8000л. або 8м^3. Дане приміщення утеплене по усьому контуру стіни, теплоізолятором товщиною 10см. Тому тепловтрати цього приміщення є незначними. Найбільша втрата тепла може відбуватись через вікно та двері при їхньому відкритті. В приміщені ще знаходяться полиці та продукти об’єм яких близько 1.5м^3. Отже корисний об’єм приміщення сягає 6.5 м^3. Продуктивність підібраної системи провітрювання рівна 300 л./хв. Це означає, що приблизно за 22хв. приміщення буде повністю провітрене, а в межах 1-2 год. – температура повітря буде збалансованою відносно зовнішньої температури середовища.

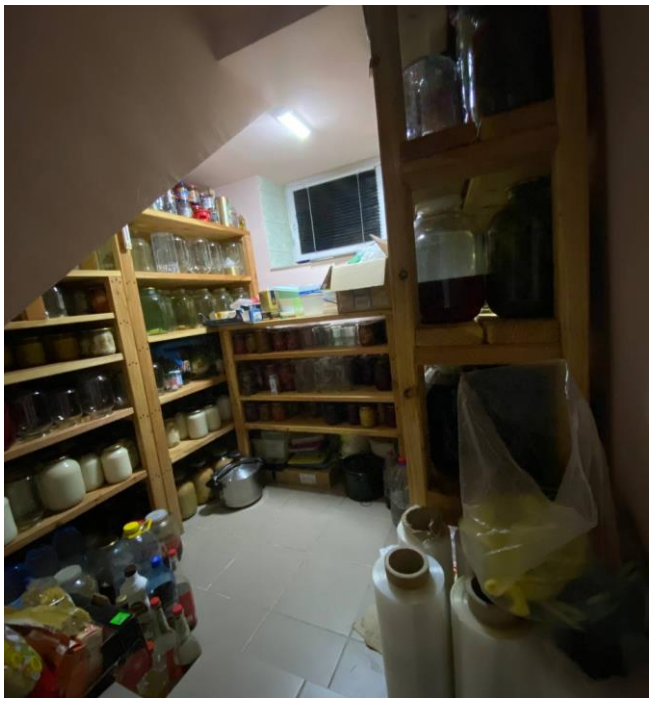


Рисунок 3.15 – Вигляд складського приміщення

Після вимірювань складського приміщення було спроектовано та розроблено корпус комп’ютеризованої системи альтернатибного охолодження із врахуванням усіх кріплень та повітряних каналів. Корпус складається з кришки, основного каркасу та змінних модульних елементів (рис. 3.16).

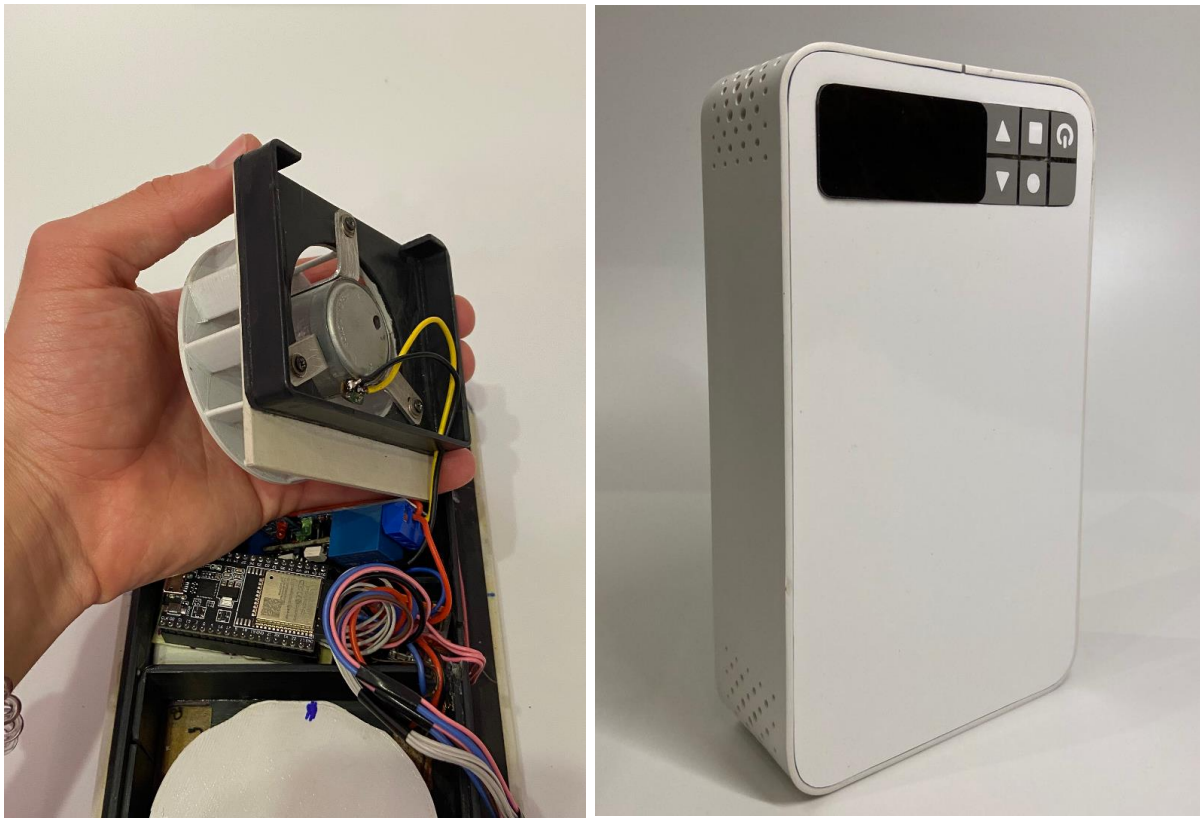


Рисунок 3.16 – Елементи розробленої комп’ютеризованої системи

Наступним кроком стане програмування головного керуючого мікроконтролера розробленої комп’ютеризованої системи ESP32 WROOM.

3.3 Налаштування та тестування реалізованої системи

Після реалізації та тестування програмного коду в середовищі Wokwi, можна перейти до програмування контролера. Для того аби прошити мікроконтролер ESP32 за допомогою програми Arduino IDE, слід завантажити драйвер USD-UART cp2102. Даний драйвер призначений для прошивки мікроконтролерів з UART мостом cp2102. Його можна завантажити з офіційного сайту silicon labs (рис. 3.17) за наступним посиланням: https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads.

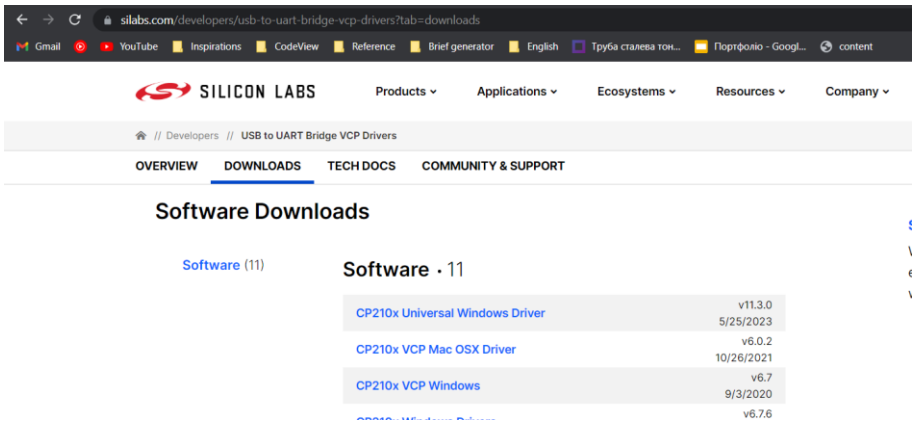


Рисунок 3.17 – Сторінка завантаження драйвера

Після завантаження та інсталяції драйвера переходимо до Arduino IDE. Підключимо спершу мікроконтролер до комп'ютера. Після підключення COM-порт має визначитись автоматично (рис. 3.18).

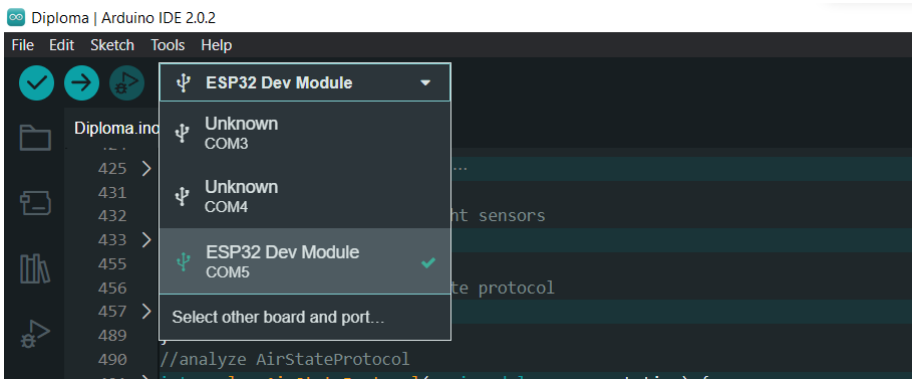


Рисунок 3.18 – Відображення COM-порту

Після того, як COM-порт обраний, натискаємо кнопку у вигляді галочки, яка розміщена з лівого верхнього боку робочого вікна програми Arduino IDE. Після компіляції, код почне записуватись на мікроконтролер ESP32 WROOM.

Після прошивки мікроконтролера, перейдемо до тестування комп'ютеризованої системи та перевіримо усі її складові в спільній роботі. Для цього у локальному режимі запустимо сервер та клієнтський веб-сайт.

Для запуску сервера та клієнтського веб-сайту, в командному рядку необхідно ввести наступні команди: "npm run" dev для сервера та "npm start" для веб-сайту. Після запуску в браузері відкриється клієнтська веб-сторінка, а саме сторінка авторизації. Яка призначена для реєстрації та входу на персональну сторінку користувача.

Персональна сторінка користувача містить панель керування із можливостю додати нову кімнату або пристрій та налаштувати їх. Опція "Додати" складається із двох параметрів, а саме додати кімнату та додати пристрій. Результат додавання розробленої комп'ютеризованої системи охолодження, зображений на рисунку 3.19.

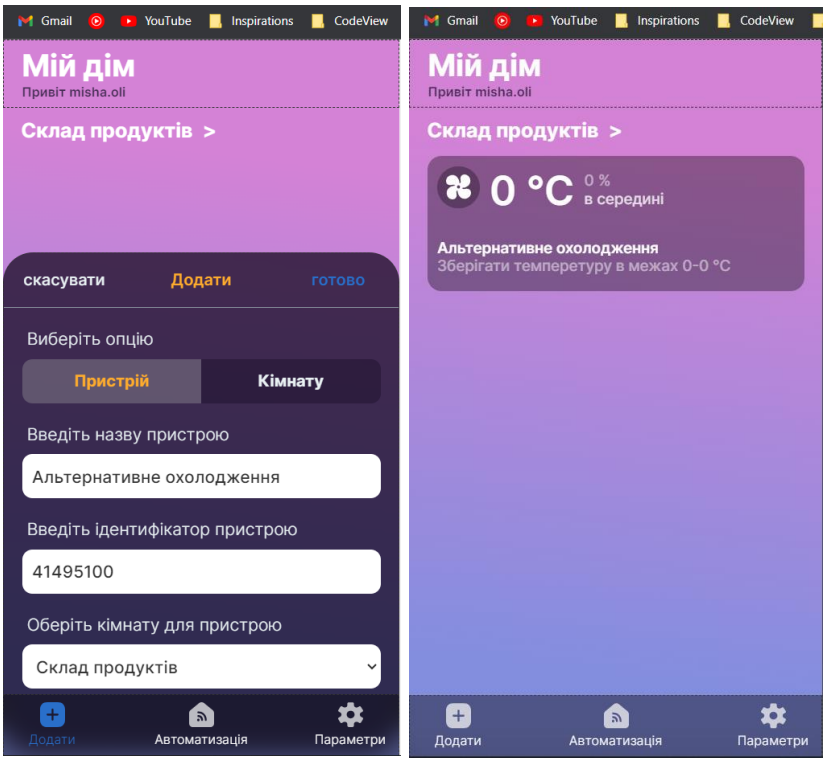


Рисунок 3.19 – Результат додавання пристрою до кімнати

Якщо пристрій увімкнений та підключений до сервера за декілька секунд на веб-сторінці відображатиметься стан повітря та параметри пристрою. Натиснувши на пристрій відкриється меню його налаштувань.

В панелі налаштування пристрою можна змінити параметри мінімальної допустимої і максимальної допустимої температури та увімкнути чи вимкнути пристрій. Результат зміни параметрів пристрою зображений на рисунку 3.20. Також варто зауважити що при вимкнені пристрою показники температури всеодно будуть зніматись із давачів. Це зроблено для того, щоб користувач міг бачити стан температури і зреагувати в разі її критичного значення. Дана опція не споживає багато енергоресурсів тому є досить доцільною з практичної сторони використання. Коли пристрій охолодження увімкнув систему вентиляції на клієнтському веб-сайті буде відображатись анімація обертання іконки вентилятора.

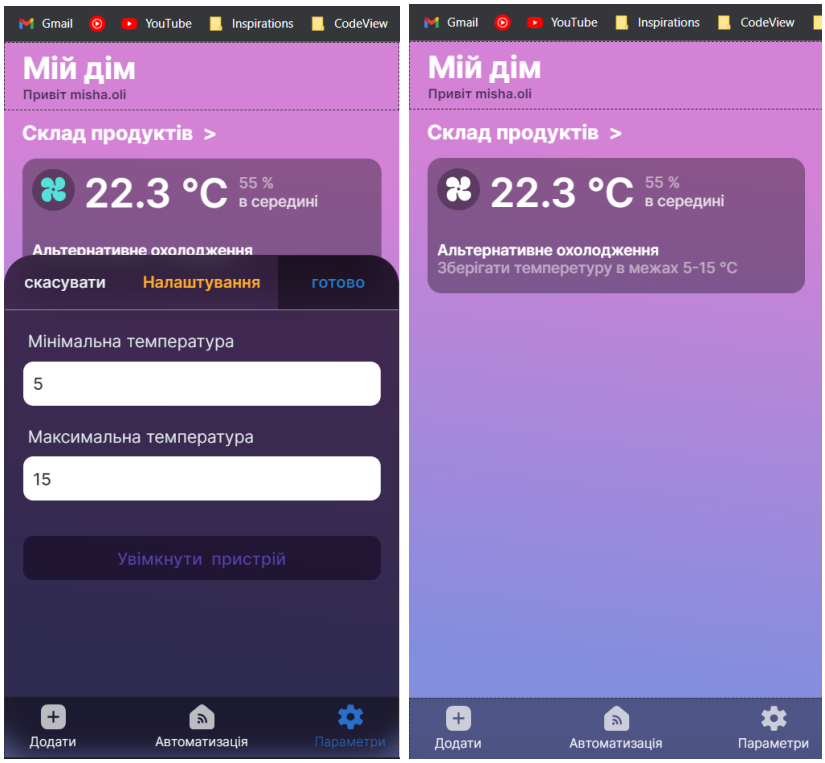


Рисунок 3.20 – Результат зміни параметрів пристрою

Також при натисканні на назву кімнати, відкриється меню налаштування кімнати у якому можна перейменувати або видалити її. Якщо просто натиснути на кнопку налаштувань, відкриється панель із можливістю вийти з персональної сторінки. Вихід передбачає видалення JWT refresh-токена із бази даних (рис. 3.21). Це унеможливить вхід стороннім особам, на персональну сторінку користувача за допомогою перехопленого JWT refresh-токена.

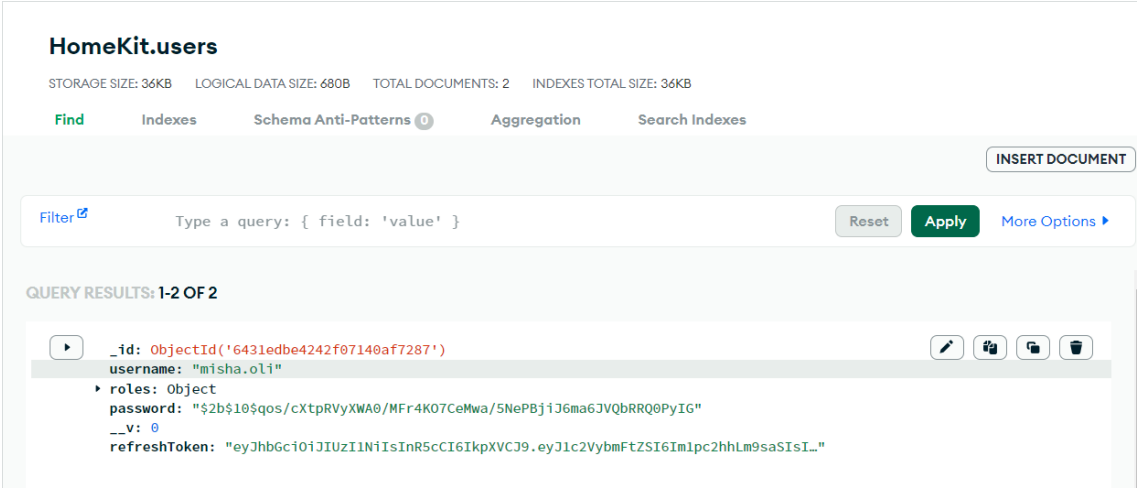


Рисунок 3.21 – Дані поля refreshToken перед виходом з облікової сторінки

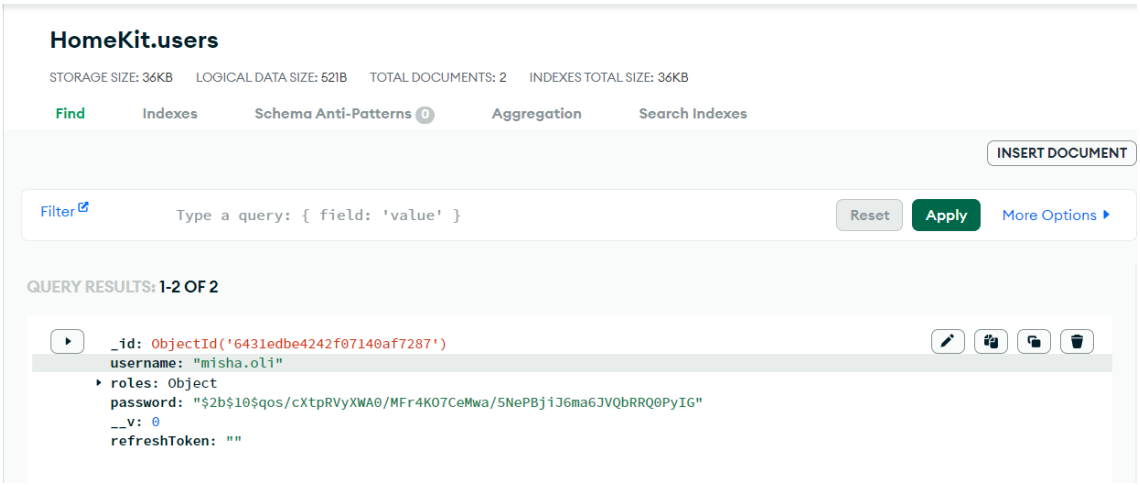


Рисунок 3.22 – Дані поля refreshToken після виходу з облікової сторінки

Як видно з рисунку 3.22, після виходу користувача з облікової сторінки, поле refreshToken стало пустим.

РОЗДІЛ 4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Розробка захисту від пожеж та вибухів в системах опалення, вентиляції та кондиціонування повітря

Безпека життєдіяльності є надзвичайно важливою для забезпечення безпеки та захисту людей, майна та навколишнього середовища. В системах опалення, вентиляції та кондиціонування повітря існує потенційна загроза пожежі та вибуху, яка може призвести до серйозних наслідків. Розробка ефективних заходів безпеки є необхідною для запобігання подібним аваріям та забезпечення безпеки приміщень та користувачів.

Загрози пожежі та вибуху в системах опалення, вентиляції, освітлення та кондиціонування повітря:

– причини пожеж та вибухів в цих системах можуть включати неправильну експлуатацію, несправності обладнання, неправильне встановлення, недотримання технічних норм та стандартів;

– потенційні наслідки пожеж та вибухів включають пошкодження майна, травми та загрозу життю людей, втрату продуктивності та забруднення навколишнього середовища.

Розробка заходів безпеки:

1) аналіз ризиків:

– проведення оцінки ризиків для ідентифікації потенційних загроз пожежі та вибуху;

– визначення вразливих місць та слабких точок систем опалення, вентиляції, освітлення та кондиціонування повітря.

2) проектування системи:

– врахування протипожежних та противибухових вимог при проектуванні систем;

– встановлення високоякісного та сертифікованого обладнання та матеріалів;

– забезпечення належного проведення монтажних робіт та дотримання вимог технічних норм і стандартів.

3) пожежна безпека:

– встановлення автоматичних систем попередження пожеж та пожежогасіння;

– регулярне технічне обслуговування та перевірка протипожежного обладнання;

– забезпечення наявності ефективних шляхів евакуації та планів дій у разі пожежі.

4) противибухова безпека:

– врахування вимог до захисту від вибуху при виборі обладнання та матеріалів;

– застосування вибухозахищених систем та пристроїв;

– використання заземлення та видалення небезпечних речовин.

5) контроль та навчання:

– регулярний контроль технічного стану систем та обладнання;

– навчання персоналу щодо протипожежної та противибухової безпеки;

– проведення навчальних пожежних та евакуаційних тренувань.

Розробка заходів безпеки в системах опалення, вентиляції та кондиціонування повітря згідно ДБН В.2.5-67:2013 “Опалення, вентиляція та кондиціонування”, передбачають аналіз ризиків, проектування систем з урахуванням протипожежних та противибухових вимог, застосування протипожежного, навчання персоналу та контроль технічного стану систем. Реалізація цих заходів в розробці комп’ютеризованій системі охолодження дозволить забезпечити безпеку приміщень та зменшити ризик виникнення пожеж.

4.2 Допомога при попаданні чужорідних тіл у вухо, ніс, очі та дихальні шляхи

При роботі з комп’ютеризованою системою альтернативного охолодження, слід бути уважним та знати як поводитись в разі потрапляння чужорідного тіла у вухо, ніс, очі та дихальні шляхи. Оскільки вона провітрює приміщення та накопичує пилюку в фільтрах.

В разі потрапляння чужорідних тіл до зовнішнього слухового проходу. Розрізняють два види чужорідних тіл - живі й неживі. Живі - це комахи (клопи, таргани, мошки, мухи), неживі - дрібні предмети (ґудзики, намистини, горох, кісточки від ягід, насіння, шматки вати), які потрапляють в зовнішній слуховий прохід.

Неживі чужорідні тіла, як правило, не викликають ніяких больових відчуттів і знаходження їх у вусі не веде до яких-небудь серйозних наслідків. Тому невідкладної допомоги в таких випадках не потрібно. Всякі спроби оточуючих або самого потерпілого видалити стороннє тіло можуть лише сприяти подальшому проштовхування цих тіл в глиб слухового проходу, тому видалення сторонніх тіл неспеціалістами категорично забороняється.

Живі сторонні тіла можуть викликати неприємні, суб'єктивні відчуття - почуття свердління, печіння, біль.

Невідкладна допомога - необхідно заповнити слуховий прохід рідким маслом, борним спиртом або водою і змусити потерпілого кілька хвилин полежати на здоровій стороні. При цьому комаха гине негайно ж і суб'єктивні розлади проходять. Після зникнення неприємних відчуттів у вусі потерпілого необхідно покласти на хвору сторону. Нерідко разом з рідиною з вуха видаляються і чужорідні тіла. Якщо чужорідне тіло залишається у вусі, то хворого слід доставити до лікаря - отоларинголога.

Потрапляння чужорідного тіла в порожнину носа. Частіше зустрічаються у дітей, які самі заштовхують в ніс дрібні предмети (кульки, намистини, шматки паперу або вати, ягоди, ґудзики).

Сторонні тіла розташовуються в основному в нижніх носових ходах. При наданні першої допомоги дозволяється спробувати видалити стороннє тіло тільки сильним сяканням, закривши щільно ніздрю вільної половини носа.

Якщо це не вдається зробити з 2-3 спроб, необхідно терміново відвести дитину на травматологічний пункт дитячої клініки. Самим застосовувати різні інструменти для видалення стороннього тіла заборонено, так що можна поранити слизову оболонку або заштовхати чужорідне тіло у вузьке кісткове простір, або виштовхнути в носоглотку воно може потрапити в гортань або трахею.

Потрапляння чужорідного тіла в очі. Зазвичай дрібні не гострі предмети (смітинки, мошки, піщинки, вії), затримуючись на кон'юнктиві (слизова оболонка), викликають гостре відчуття печіння в оці, посилення при миганні, сльозотеча. Якщо чужорідне тіло не видалити, виникає набряк кон'юнктиви, почервоніння, порушується функція очей. Сторонній предмет зазвичай розташовується під верхньою або нижньою повікою. Чим раніше видалено чужорідне тіло, тим швидше пройдуть всі викликані явища.

Терти очей не можна, оскільки це ще більше подразнює кон'юнктиву.

Необхідно оглянути очей і видалити смітинку. Спочатку оглядають кон'юнктиву нижньої повіки: хворого просять подивитися вгору, який надає допомогу відтягує нижню повіку вниз, тоді добре видна вся нижня частина кон'юнктиви.

Сторонній предмет видаляють щільним ватним гнотиком, сухим або вологим. Видалення чужорідного тіла з-під верхнього століття дещо складніше - необхідно вивернути повіку назовні кон'юнктивою. Для цього хворого просять направити погляд вниз, який надає допомогу, захопивши двома пальцями правої руки верхню повіку, відтягує її вперед і вниз, потім вказівним пальцем лівої руки, накладеним поверх верхньої повіки, відкриває її рухом знизу вверх. Після видалення стороннього тіла хворого просять подивитися вгору і вивернута повіка повертається самостійно в звичайне положення.

Категорично забороняється видалення сторонніх тіл, вторгуючись в рогівку. Це можна зробити тільки в медичному закладі.

Потрапляння чужорідного тіла в дихальні шляхи може привести до повної закупорки і розвитку асфіксії. Найбільш часто сторонні тіла дихальних шляхів спостерігаються у дітей У дорослих в дихальні шляхи частіше потрапляє їжа: у випадках, коли людина розмовляє під час їжі, або при захворюваннях надгортанника, коли він нещільно в момент ковтання закриває вхід в гортань. Чужорідні тіла, що знаходяться в роті, при глибокому вдиху разом з повітрям проникають в гортань і трахею, що викликає напад різкого кашлю. Сторонній предмет часто в момент кашлю видаляється. При великих сторонніх тілах може виникнути спазм голосових зв'язок, тоді тіла стають міцно фіксованими, а просвіт голосової щілини повністю закритим, що викликає задуху.

Якщо різке і сильне відкашлювання не приводить до видалення стороннього тіла, то роблять спроби видалити його активно. Постраждалого укладають животом на зігнуте коліно, голову опускають вниз якнайнижче і ударами рукою по спині стрясають грудну клітку, здавлюють при цьому епігастральну область. При відсутності ефекту потерпілого укладають на стіл, голову різко відгинають назад і через відкритий рот оглядають область гортані. При виявленні стороннього тіла його захоплюють пінцетом, пальцями, корнцангом і видаляють. Постраждалого слід доставити в лікувальний заклад. При повному закритті дихальних шляхів, що розвилася асфіксії і неможливості видалити стороннє тіло, єдина міра порятунку - екстрена трахеотомія.

Перша медична допомога. Не треба намагатися витягти сторонній предмет пальцем або пінцетом, найчастіше це приводить до проштовхування його в нижні відділи дихальної системи. Якщо постраждала дитина, то краще всього перевернути її головою вниз і постукати долонею по спині на рівні лопаток. Якщо постраждав дорослий, то встати позаду нього, попросити його нахилитися вперед і долонею різко вдарити його між лопаток. Якщо потерпілий втратив свідомість, то його слід укласти на бік і також вдарити між лопаток долонею. Не можна бити між лопаток кулаком або ребром долоні! У цьому випадку можна пошкодити хребет.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було розроблено комп'ютеризовану систему віддаленого керування альтернативним охолодженням приміщень для зберігання запасів продуктів. Для її реалізації було використано ряд технологій, таких як React.js, Node.js, Express.js та MongoDB.

При створенні комп'ютеризованої системи віддаленого керування альтернативним охолодженням було:

– досліджено та проаналізовано існуючі методи та засоби охолодження приміщень;

– здійснено оцінку актуальності впровадження розробленої комп'ютеризованої системи;

– створено комп'ютерну модель комп'ютеризованої системи за допомогою сервісу Wokwi;

– розроблено узагальнену структурну та функціональну схеми комп'ютеризованої системи;

– спроектовано схему електричну принципову апаратної складової комп'ютеризованої системи та описано її елементну базу;

– спроектовано комп'ютеризовану систему, з використанням мікроконтролера ESP32 WROOM 32U, сенсорних кнопок, OLED-дисплею, LED-індикаторів, 5В-го реле модуля та конекторів;

– розроблено програмне забезпечення для апаратної та клієнт-серверної частини комп'ютеризованої системи;

– налаштовано та протестовано комп'ютеризовану систему.

Розроблена комп'ютеризована система виконує віддалене керування альтернативним охолодженням приміщення для зберігання запасів продуктів. При тестуванні розробленої системи не було виявлено критичних помилок. Всі функції працюють коректно та без збоїв.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Паламар М.І., Стрембіцький М.О., Паламар А.М. Проектування комп'ютеризованих вимірювальних систем і комплексів. Навчальний посібник. Тернопіль: ТНТУ. 2019. 150 с.

2. Січень 2020 року був аномально теплим і сухим. URL: https://kurkul.com/news/19569-sichen-2020-roku-buv-anomalno-teplim-i-suhim (дата звернення 01.03.2023)

3. ESP32-DevKitC V4 Getting Started Guide. URL: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/get-started-devkitc.html (дата звернення 10.03.2023)

4. Датчик вологості та температури DHT11. URL: https://arduino.ua/prod185-datchik-vlajnosti-i-temperatyri-dht11 (дата звернення 10.03.2023)

5. OLED дисплей 0.96" I2C 128x64. URL: https://arduino.ua/prod1263-oled-displei-modyl-belii (дата звернення 10.03.2023)

6. TTP223 TOUCH KEY сенсорний датчик для Arduino. URL: https://arduino.ua/prod264-ttp223-touch-key-sensornii-datchik-dlya-arduino]: (дата звернення 10.03.2023)

7. Модуль реле 5В 10А низького рівня. URL:https://arduino.ua/prod1706-modyl-rele-5v-10a-nizkogo-yrovnya-low-level (дата звернення 10.03.2023)

8. Використання розширення Visual Studio Code. URL: https://learn.microsoft.com/uk-ua/power-apps/maker/portals/vs-code-extension (дата звернення 05.04.2023)

9. Вступ в React, якого нам не вистачало. URL: https://devzone.org.ua/post/vstup-v-react-yakogo-nam-ne-vistachalo (дата звернення 25.04.2023)

10. Що таке Node JS простими словами. URL: https://dan-it.com.ua/uk/blog/chto-jeto-takoe-node-js-prostymi-slovami/ (дата звернення 30.04.2023)

11. Розробка ECOMMERCE проектів TECHNOLOGIES MONGO DB. URL: https://brander.ua/technologies/mongo-db (дата звернення 14.05.2023)

12. Технічна документація Express.js. URL: https://expressjs.com/ (дата звернення 03.05.2023)

13. Технічна документація бібліотеки компонентів React.js. URL: https://legacy.reactjs.org/docs/ (дата звернення 25.04.2023)

14. Технічна документація Node.js. URL: https://nodejs.org/en/docs (дата звернення 30.04.2023)

15. Технічна документація MongoDB. URL: https://www.mongodb.com/docs/ (дата звернення 14.05.2023)

16. Осухівська Г.М., Тиш Є.В., Луцик Н.С., Паламар А.М. Методичні вказівки до виконання кваліфікаційних робіт здобувачів першого (бакалаврського) рівня вищої освіти спеціальності 123 «Комп'ютерна інженерія» усіх форм навчання. Тернопіль, ТНТУ. 2022. 28 с.

17. Ларіоник Р.В., Луцик Н.С., Паламар А.М. Система для моніторингу якості атмосферного повітря на базі IoT. Матеріали IX науково-технічної конференції "Інформаційні моделі, системи та технології" Тернопільського національного технічного університету імені Івана Пулюя (Тернопіль, 8–9 грудня 2021 року), Тернопіль: ТНТУ. 2021. С. 116.

18. Palamar A., Karpinski M., Palamar M., Osukhivska H., Mytnyk M. Remote Air Pollution Monitoring System Based on Internet of Things. In CEUR Workshop Proceedings, 2022. Vol. 3309. P. 194-204.

19. ДБН В.2.5-67:2013 "Опалення, вентиляція та кондиціонування". Київ. 2013.

Додаток А

Технічне завдання

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний технічний університет імені Івана Пулюя
Факультет комп'ютерно-інформаційних систем і програмної інженерії

Кафедра комп'ютерних систем та мереж

"Затверджую"
Завідувач кафедри КС
_____ Осухівська Г.М.
"___" _____2023 р

КОМП'ЮТЕРИЗОВАНА СИСТЕМА ВІДДАЛЕНОГО КЕРУВАННЯ АЛЬТЕРНАТИВНИМ ОХОЛОДЖЕННЯМ ПРИМІЩЕНЬ ДЛЯ ЗБЕРІГАННЯ ЗАПАСІВ ПРОДУКТІВ

ТЕХНІЧНЕ ЗАВДАННЯ

на _8_ листках

Вид робіт: Кваліфікаційна робота

На здобуття освітнього ступеня «Бакалавр»

Спеціальність 123 «Комп'ютерна інженерія»

«УЗГОДЖЕНО»	«ВИКОНАВЕЦЬ»
Керівник кваліфікаційної роботи	Студент групи СІ-41
_____ Луцик Н. С.	_____ Олійник М. Є.
«____» _____ 2023 р.	«____» _____ 2023 р.

Тернопіль 2023

1 Загальні відомості

1.1 Повна назва та її умовне позначення

Повна назва теми кваліфікаційної роботи: Комп’ютеризована система віддаленого керування альтернативним охолодженням приміщень для зберігання запасів продуктів.

Умовне позначення кваліфікаційної роботи: КС КРБ 123.050.00.00.

1.2 Виконавець

Студент групи СІ-41, факультету комп’ютерно-інформаційних систем і програмної інженерії, кафедри комп’ютерних систем та мереж, Тернопільського національного технічного університету імені Івана Пулюя, Олійник Михайло Євгенович.

1.3 Підстава для виконання роботи

Підставою для виконання кваліфікаційної роботи є наказ по університету (№4/7-238 від 28.02.2023 р.)

1.4 Планові терміни початку та завершення роботи

Плановий термін початку виконання кваліфікаційної роботи – 28.02.2023 р.

Плановий термін завершення виконання кваліфікаційної роботи – 22.06.2023 р.

1.5 Порядок оформлення та представлення результатів роботи

Порядок оформлення пояснювальної записки та графічного матеріалу здійснюється у відповідності до чинних норм та правил ІСО, ГОСТ, ЕСКД, ЕСПД та ДСТУ.

Представлення проміжних результатів роботи з виконання кваліфікаційної роботи здійснюється у відповідності до графіку, затвердженого керівником роботи. Попередній захист кваліфікаційної роботи відбувається при готовності роботи на 90% , наявності пояснювальної записки та графічного матеріалу.

Пред’явлення результатів кваліфікаційної роботи відбувається шляхом захисту на відповідному засіданні ЕК, ілюстрацією основних досягнень за допомогою графічного матеріалу.

2 Призначення і цілі створення системи

2.1 Призначення системи

Призначенням комп’ютеризованої системи є охолодження складського приміщення використовуючи природні ресурси, такі як холодне повітря зовнішнього середовища. Управління системою може здійснюється через персональну сторінку користувача або у локальному режимі.

Основні задачі, які повинна виконувати комп’ютеризована система:

– реєстрація користувачів у системі;

– забезпечення авторизованого доступу;

– можливість віддаленого налаштування комп’ютеризованої системи;

– можливість локального налаштування комп’ютеризованої системи;

– підтримання оптимальних умов температури в складському приміщенні;

– відстеження та реагування комп’ютеризованою системою на помилки в разі їх появи;

2.2 Мета створення системи

Метою даної кваліфікаційної роботи є розробка комп'ютеризованої системи віддаленого керування альтернативним охолодженням приміщень для зберігання запасів продуктів.

2.3 Характеристика об'єкту

Об'єкт, для якого розробляється дана система, є складське приміщення, об’єм якого 8м^3. Приміщення містить один дверний та один віконний проєм. Приміщення облаштоване по контуру стелажами для зберігання консервованих продуктів.

3 Вимоги до системи

3.1 Вимоги до системи в цілому

Забезпечення безперебійної роботи системи, незважаючи на стан підключення до мережі Internet.

3.1.1 Вимоги до структури та функціонування системи

Структура комп’ютерної системи альтернативного охолодження складається з:

– веб-сервера;

– клієнт сервера;

– бази даних;

– комп’ютеризованої системи охолодження.

Основні функціональні вимоги характеризуються наступними критеріями:

– точність вимірювання;

– надійність;

– захищеність;

– зручність монтажу та модернізації;

– контрольованість.

3.1.2 Вимоги до способів та засобів зв’язку між компонентами системи

Основна вимога, яка ставиться до способів та засобів інформаційного обміну – це їх узгодженість.

3.1.3 Вимоги по діагностуванню системи

Діагностування комп’ютеризованої системи віддаленого керування альтернативним охолодженням приміщень для зберігання запасів продуктів проводиться у відповідності до регламенту, визначеного підприємством, що впроваджуватиме дану систему.

3.1.4 Перспективи розвитку та модернізації системи

Передбачаються перспективи розвитку комп’ютеризованої системи, що включають масштабованість і взаємодію з іншими підсистемами для їхнього контролю.

3.2 Показники призначення

Комп’ютеризована система повинна передбачати можливість масштабування. Можливості масштабування повинні забезпечуватися засобами використовуваного базового програмного і технічного забезпечення.

3.2.1 Вимоги до надійності

Комп’ютеризована система повинна забезпечувати працездатність та відновлення своїх функцій при виникненні наступних ситуацій:

– при збоях в системі електропостачання апаратної частини;

– при помилках в підключенні до мережі Internet;

– при помилках а роботі апаратних засобів;

Для захисту апаратури від стрибків напруги і комутаційних завад повинні застосовуватися мережні фільтри.

3.3 Вимоги до безпеки

Зовнішні елементи технічних засобів системи, що перебувають під напругою, повинні мати захист від випадкового дотику, а самі технічні засоби мати занулення або захисне заземлення .

Система електроживлення повинна забезпечувати захисне вимикання при перевантаженнях і коротких замиканнях в колах навантаження, а також аварійне ручне вимикання.

Загальні вимоги пожежної безпеки повинні відповідати нормам на побутове електрообладнання. У разі пожежі не має виділятися отруйних газів і димів. Після зняття електроживлення має бути доступне застосування будь-яких засобів пожежогасіння.

3.4 Вимоги до захисту інформації від несанкціонованого доступу

Система повинна забезпечувати захист від несанкціонованого доступу на рівні не нижче встановленого вимогами, що пред'являються до категорії 1Д по класифікації документа, що діє, “Автоматизовані системи. Захист від несанкціонованого доступу до інформації. Класифікація автоматизованих систем”.

Компоненти підсистеми захисту від НСД повинні забезпечувати:

– ідентифікацію користувача;

– перевірку повноважень користувача при роботі з системою;

– розмежування доступу користувачів.

Рівень захищеності від несанкціонованого доступу повинен відповідати вимогам класу захищеності згідно вимогам документу “Засоби обчислювальної техніки. Захист від несанкціонованого доступу до інформації. Показники захищеності від несанкціонованого доступу до інформації”.

4 Вимоги до документації

Документація повинна відповідати вимогам ЄСКД та ДСТУ

Комплект документації повинен складатись з:

– пояснювальної записки;

– графічного матеріалу:

1) функціональна схема комп’ютеризованої системи.

2) схема електрична принципова.

3) діаграма послідовностей.

4) архітектура сервера комп’ютеризованої системи

5) блок-схема функції аналізу стану повітря.

*Примітка: У комплект документації можуть вноситися міни та доповнення в процесі розробки.

5 Стадії та етапи проектування

Таблиця 1 – Стадії та етапи виконання кваліфікаційної роботи бакалавра

№ етапу	Назва етапу виконання кваліфікаційної роботи	Термін виконання
1	Розробка та затвердження технічного завдання	01.03-02.03.2023
2	Аналіз технічного завдання та побудова можливих рішень	04.03-10.03.2023
3	Аналіз складського приміщення	11.03-15.03.2023
4	Розробка структурної та функціональної схеми	16.03-24.03.2023
5	Розробка діаграми послідовностей та архітектури серверної частини розроблюваної системи	25.03-31.03.2023
6	Розробка схеми електричної принципової	01.04-04.04.2023
7	Вибір елементної бази	05.04-09.04.2023
8	Моделювання апаратної складової комп’ютеризованої системи	10.04-30.04.2023
9	Реалізація програмного забезпечення комп’ютеризованої системи	01.05-25.05.2023
10	Налаштування та тестування реалізованої системи	26.05-31.05.2023
11	Безпека життєдіяльності, основи охорони праці	01.06-04.06.2023
12	Оформлення кваліфікаційної роботи	05.06-12.06.2023
13	Оформлення графічної частини	13.06-15.06.2023
14	Попередній захист кваліфікаційної роботи	16.06-19.06.2023
15	Захист кваліфікаційної роботи	22.06.2023.2023

6 Додаткові умови виконання кваліфікаційної роботи

Під час виконання кваліфікаційної роботи у дане технічне завдання можуть вноситися зміни та доповнення.

Додаток Б

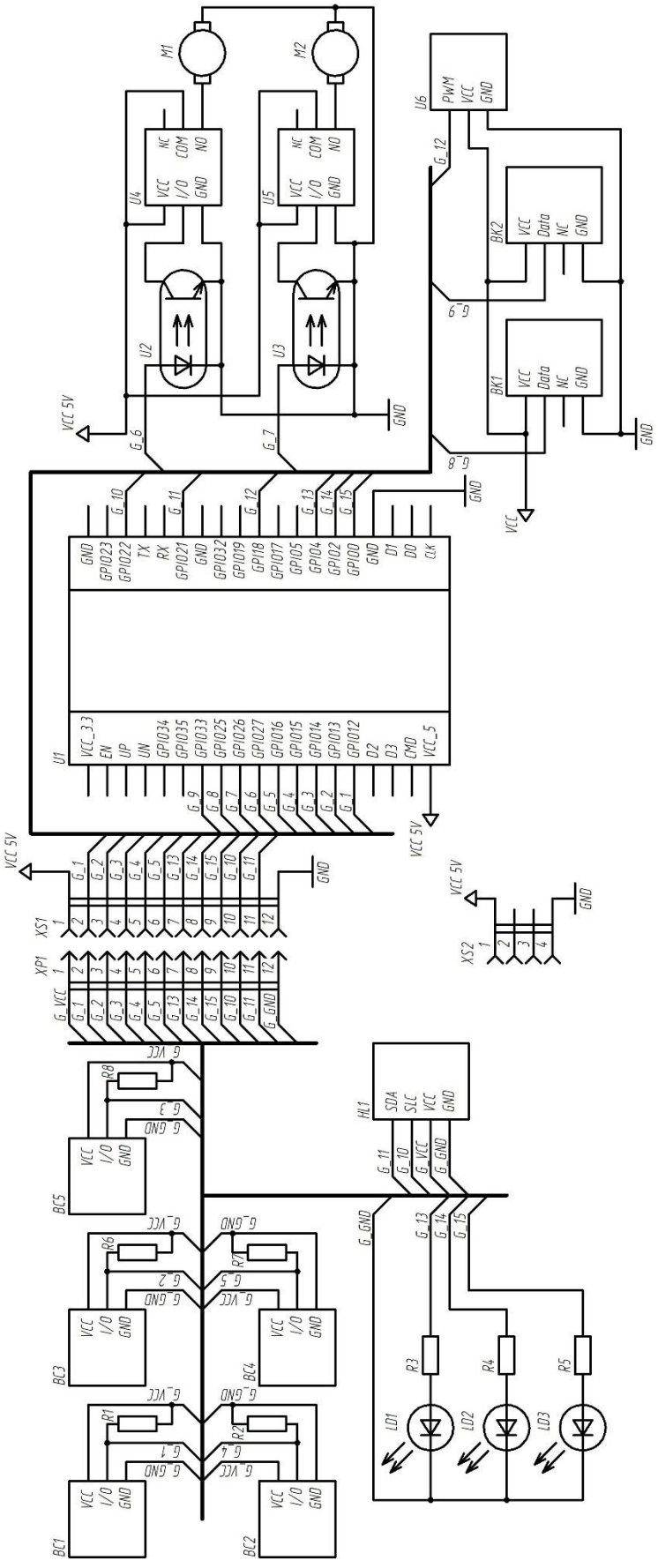
Перелік елементів

Поз. позначення	*Найменування*	*Кіл.*	*Примітка*
	З’єднувачі контактні		
XS1	EL4.5-12P F	1	
XP1	EL4.5-12P M	1	
XS2	USB_C	1	
	Індикатори		
HL1	SSD1306	1	
	Світлодіоди		
LED1..LED3	SMD LED Diode 5050	3	
	Резистори		
R1,R2,R6..R8	0805-0,125-10 кОм±5%	5	
R3..R5	0805-0,125-330 Ом±5%	2	
	Мікросхеми		
U1	ESP32 WROOM 32U	1	
U4, U5	AOC456	2	
U6	SG90	1	
	Оптрони		
U2, U3	PC817C	2	
	Давачі		
BC1...BC5	TTP223	5	
BK1, BK2	DHT11	2	
	Двигуни постійного струму		
M1,M2	WRF-520TB-PM7150T	2	

					КС КРБ 123.050.00.00 ПЕ	
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*		
Розробив		*Олійник М. Є*			*Комп’ютеризована система віддаленого керування альтернативним охолодженням приміщень для зберігання запасів продуктів*	*Літ.* / *Арк.* 1 / *Акркушів*
Перевірив		*Луцик Н.С.*				
Рецензент						*ТНТУ, каф. КС, гр. СІ-41*
Н. Контр.		*Тиш Є.В.*				
Затвердив		*Осухівська Г.М.*				

Додаток В

Схема електрична принципова



Додаток Д

Лістинг коду апаратної частини

```
//Diploma project for aircontrolling system autor: Mykhailo Oliynyk
CI-41
#include <DHT.h>
#include <ESP32Servo.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <WiFi.h>
#include <HTTPClient.h>
#include <Arduino_JSON.h>
const char* WIFI_SSID = "Tenda_300480";
const char* WIFI_PASSWORD = "222vova222";
String serverName = "http://192.168.0.121:5500/updating";
//choose dht type of sensors
#define DHTTYPE DHT11
//include DHT pins
#define DHT_INSIDE_PIN 25
#define DHT_OUTSIDE_PIN 33
//servo motors pin
#define SERVO 18
//controll diods` pins
#define BLUE_D 0
#define GREEN_D 2
#define YELLOW_D 4
//relay pins
#define RELAY_IN 27
#define RELAY_OUT 26
//sensor buttons` pins
#define SENSOR_BUTTONS 5
#define SENSOR_ONE 12
#define SENSOR_TWO 13
#define SENSOR_THREE 14
#define SENSOR_FOUR 15
#define SENSOR_FIVE 16
// Ours button routine:
#define BUTTON_RELEASED 0
#define BUTTON_PRESSED 1
#define BUTTON_SHORT_PRESS 2
#define BUTTON_LONG_PRESS 3
// only consider button input of at least 50ms as valid (debouncing)
#define WLED_DEBOUNCE_THRESHOLD 50
// long press if button is released after held for at least 600ms
#define WLED_LONG_PRESS 600
//OLED monitor data pin
#define SDA_PIN 21
#define SCL_PIN 22
// Oled monitor size
#define SCREEN_WIDTH 128  // OLED width,  in pixels
```

```
#define SCREEN_HEIGHT 64  // OLED height, in pixels 32 mine
// Air state routine:
const int G_IN_G_Out = 0;
const int G_IN_B_Out = 1;
const int B_IN_G_Out = 2;
const int B_IN_B_Out = 3;
const int LOWER_T_MIN = 4;
//initialisation of DHT pins
DHT dht_out(DHT_OUTSIDE_PIN, DHTTYPE);
DHT dht_in(DHT_INSIDE_PIN, DHTTYPE);
// create an OLED display object connected to I2C
Adafruit_SSD1306 oled(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
Servo servo;
// Own copies of these, separate from globals.
bool sensorPressedBefore[SENSOR_BUTTONS];
bool sensorLongPressed[SENSOR_BUTTONS];
unsigned long sensorPressedTime[SENSOR_BUTTONS];
int sensorStatus[SENSOR_BUTTONS];
byte sensorPin[SENSOR_BUTTONS] = {
  SENSOR_ONE,
  SENSOR_FOUR,
  SENSOR_FIVE,
  SENSOR_TWO,
  SENSOR_THREE
};
unsigned long shortPressTimer[SENSOR_BUTTONS] = { 0, 0, 0, 0, 0 };
//variable for functionality
float insideTemperature = 0;
int insideHumidity = 0;
float outsideTemperature = 0;
int outsideHumidity = 0;
bool insideDhtErr = false, outsideDhtErr = false;
bool isRuning = false;
//default data setup
bool isOn = false, canConnect = false;
int min_temperature = 7;
int max_temperture = 18;
int productivitySetCoutner = 0;
int previus_inside_temperature;
unsigned long start_cooling_time;
double productivity;
bool productivityIsSet = false;
bool isBreak = false, previosBreak = false;
int AIR_PROTOCOL;
unsigned long get_sensor_data_delay = 10000;     // milisecond
unsigned long check_productivity_delay = 15000;  // milisecond
unsigned long http_request_delay = 10000;        // milisecond
unsigned long max_working_time = 50000;       //30*60*1000;        //
milisecond
unsigned long break_delay_time = 10000;       //5*60*1000;         //
milisecond
unsigned long setResentleTimeAlowed = 100000;   //30*60*1000;     //
milisecond
bool settingState = false;
```

```
bool setSettings = false;
bool isSettingSet = false;
bool isDisplayingDigit = false;
bool isBlynk = false;
bool isYellowBlynk = false;
bool isGreenBlynk = false;
bool getDataErr = false;
unsigned long setLedBlynkTimeActive = 500;  //30*60*1000;
unsigned long ledBlynkDelay = 10000;        //30*60*1000;
unsigned long yellowLedBlynkDelay = 2000;   //30*60*1000;
unsigned long resentlyWorkedTime = 0;
// static unsigned long previousReconnect[4] = {0,0,0,0};
unsigned long previousReconnect[8] = { 0, 0, 0, 0, 5000, 0, 1000, 0
};
void setup() {
  Serial.begin(9600);
  Serial.print("Connected to WiFi network with IP Address: ");
  // pinMode(DS_PIN, OUTPUT);
  // pinMode(STCP_PIN, OUTPUT);
  // pinMode(SHCP_PIN, OUTPUT);
  pinMode(RELAY_OUT, OUTPUT);
  pinMode(RELAY_IN, OUTPUT);
  pinMode(BLUE_D, OUTPUT);
  pinMode(GREEN_D, OUTPUT);
  pinMode(YELLOW_D, OUTPUT);
  //calling initSensor function
  initSensors();
  // attaching servo
  servo.attach(SERVO, 500, 2400);
  servo.write(0);
  //conecting to wifi network
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to WiFi ");
  Serial.println(WIFI_SSID);
  //Wait for connection
  if (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
  Serial.print("Connected to WiFi network with IP Address: ");
  Serial.println(WiFi.localIP());
  //starting dht sensors data analyzing
  dht_in.begin();
  dht_out.begin();
  // initialize OLED display with I2C address 0x3C
  if (!oled.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("failed to start SSD1306 OLED"));
  }
  // wait two seconds for initializing
  delay(2000);
  oled.clearDisplay();            // clear display
  oled.setTextSize(2);            // set text size
  oled.setTextColor(WHITE);       // set text color
  oled.setCursor(0, 2);           // set position to display (x,y)
  oled.println("Ready to work");  // set text
```

```
  oled.display();
}
//function to opening or closing air valve
void setAirValve(bool state) {
  //for smooth opening or closing we can check state of valve current
position
  if (state) {
    for (int pos = 0; pos <= 180; pos += 1) {
      servo.write(pos);
      delay(10);
    }
  } else if (!state) {
    for (int pos = 180; pos >= 0; pos -= 1) {
      servo.write(pos);
      delay(10);
    }
  }
}
void setCoolerSystem(bool state) {
  if (state) {
    isRuning = true;
    setAirValve(true);
    digitalWrite(RELAY_OUT, HIGH);
    digitalWrite(RELAY_IN, HIGH);
    Serial.println("Cooling system is on");
    digitalWrite(GREEN_D, HIGH);
    isGreenBlynk = true;
  } else if (!state) {
    digitalWrite(RELAY_OUT, LOW);
    digitalWrite(RELAY_IN, LOW);
    isRuning = false;
    setAirValve(false);
    Serial.println("Cooling system is off");
    digitalWrite(GREEN_D, LOW);
    isGreenBlynk = false;
  }
}
//check curent sensor state
bool isSensorPressed(uint8_t b) {
  return digitalRead(sensorPin[b]);
}
//showing sensor status
void showSensorStatus(int sensorStatus) {
  switch (sensorStatus) {
    case BUTTON_RELEASED:
      Serial.print("RELEASED ");
      break;
    case BUTTON_PRESSED:
      Serial.print("PRESSED ");
      break;
    case BUTTON_SHORT_PRESS:
      Serial.print("SHORT_PRESS ");
      break;
    case BUTTON_LONG_PRESS:
```

```
      Serial.print("LONG_PRESS ");
      break;
    default:
      Serial.print("UNKNOWN(");
      Serial.print(sensorStatus);
      Serial.print(") ");
      break;
  }
}
//initialization of pins and data for sensor buttons
void initSensors() {
  // Configure pins and init variables.
  for (int btn = 0; btn < SENSOR_BUTTONS; btn++) {
    pinMode(sensorPin[btn], INPUT);
    sensorPressedBefore[btn] = false;
    sensorLongPressed[btn] = false;
    sensorPressedTime[btn] = 0;
    sensorStatus[btn] = BUTTON_RELEASED;
  }
}
//Scan all buttons and store status.
void pollSensors(int n) {
  // Read them all at the same time.
  bool sensorPressedNow[SENSOR_BUTTONS];
  for (int b = n; b < SENSOR_BUTTONS; b++) {
    sensorPressedNow[b] = isSensorPressed(b);
  }
  // Some logic
  unsigned long now = millis();
  for (int b = n; b < SENSOR_BUTTONS; b++) {
    // momentary button logic
    if (sensorPressedNow[b])  // pressed
    {
      if (!sensorPressedBefore[b]) {
        sensorPressedBefore[b] = true;
        sensorPressedTime[b] = now;
      }
      long dur = now - sensorPressedTime[b];
      if (dur > WLED_LONG_PRESS)  // long press
      {
        if (!sensorLongPressed[b]) {
          sensorLongPressed[b] = true;
        }
        sensorStatus[b] = BUTTON_LONG_PRESS;
      } else if (dur > WLED_DEBOUNCE_THRESHOLD) {
        sensorStatus[b] = BUTTON_PRESSED;
      } else {
        sensorStatus[b] = BUTTON_RELEASED;
        shortPressTimer[b] = 0;
      }
    } else if (!sensorPressedNow[b] && sensorPressedBefore[b]) {
      // released
      long dur = now - sensorPressedTime[b];
      if (dur < WLED_DEBOUNCE_THRESHOLD) {
```

```
        sensorPressedBefore[b] = false;
        sensorStatus[b] = BUTTON_RELEASED;
        shortPressTimer[b] = 0;
      } else if (dur >= WLED_DEBOUNCE_THRESHOLD) {
        if (!sensorLongPressed[b])  // short press
        {
          if (sensorStatus[b] != BUTTON_SHORT_PRESS) {
            sensorStatus[b] = BUTTON_SHORT_PRESS;
            shortPressTimer[b] = millis();
          }
        } else {
          sensorStatus[b] = BUTTON_RELEASED;
          sensorLongPressed[b] = false;
          sensorPressedBefore[b] = false;
          shortPressTimer[b] = 0;
        }
        // Hold short press for some time...
        if    ((shortPressTimer[b]    !=    0)    &&    (millis()    -
shortPressTimer[b] > 50)) {
          shortPressTimer[b] = 0;
          sensorLongPressed[b] = false;
          sensorPressedBefore[b] = false;
          sensorStatus[b] = BUTTON_RELEASED;
        }
      }
    } else {
      sensorStatus[b] = BUTTON_RELEASED;
      shortPressTimer[b] = 0;
    }
  }  // end of for (int b=0; b<SENSOR_BUTTONS; b++)
}
// Retrieve last stored sensor status
int getLastPolledSensorStatus(uint8_t btn) {
  return sensorStatus[btn];
}
// Retrieve last stored status for two buttons.
int getLastPolledTwoSensorStatus(uint8_t btn1, uint8_t btn2) {
  static int lastTwoSensorStatus = BUTTON_RELEASED;
  if (sensorStatus[btn1] == sensorStatus[btn2]) {
    if (sensorStatus[btn1] != lastTwoSensorStatus) {
      lastTwoSensorStatus = sensorStatus[btn1];
    }
  }
  return lastTwoSensorStatus;
}
//set data on display
void displayData(String str) {
  oled.clearDisplay();   // clear display
  oled.setCursor(0, 2);  // set position to display (x,y)
  oled.println(str);     // set text
  oled.display();
  isDisplayingDigit = true;
}
//function to read data from dht sensors
```

```
int getSensorData() {
  insideTemperature = dht_in.readTemperature();
  insideHumidity = dht_in.readHumidity();
  outsideTemperature = dht_out.readTemperature();
  outsideHumidity = dht_out.readHumidity();
  if (isnan(outsideTemperature)) {
    Serial.println("Failed to read DHT outside data");
    //set error #protocol
    insideDhtErr = true;
    return 1;
  }
  if (isnan(insideTemperature)) {
    Serial.println("Failed to read DHT inside data");
    //set error #protocol
    outsideDhtErr = true;
    return 1;
  }
  insideDhtErr = false;
  outsideDhtErr = false;
  return 0;
}
//check all and return air state protocol
int setAirStateProtocol() {
  //G_IN_G_Out     0
  //G_IN_B_Out     1
  //B_IN_G_Out     2
  //B_IN_B_Out     3
  //ERROR        101
  if (insideTemperature > min_temperature && insideTemperature <
max_temperture) {
    //GOOD_INSIDE
    if (outsideTemperature < max_temperture) {
      //GOOD_OUTSIDE
      return G_IN_G_Out;
    } else if (outsideTemperature > max_temperture) {
      //BAD_OUTSIDE
      return G_IN_B_Out;
    }
  } else if (insideTemperature > max_temperture) {
    //BAD_INSIDE
    if (outsideTemperature < max_temperture) {
      //GOOD_OUTSIDE
      return B_IN_G_Out;
    } else if (outsideTemperature > max_temperture) {
      //BAD_OUTSIDE
      return B_IN_B_Out;
    }
  } else {
    return LOWER_T_MIN;
  }
  return 101;
}
//analyze AirStateProtocol
int analyzeAirStateProtocol(unsigned long current_time) {
```

```
  // G_IN_G_Out     0
  // G_IN_B_Out     1
  // B_IN_G_Out     2
  // B_IN_B_Out     3
  // ERROR        101
  AIR_PROTOCOL = setAirStateProtocol();
  if (AIR_PROTOCOL == 101) {
    Serial.println("analyzeAirStateProtocol error 101 ");
  }
  if (AIR_PROTOCOL == G_IN_G_Out || AIR_PROTOCOL == G_IN_B_Out) {
    //if all is good
    if (!isRuning) {
      //check system is it runing?
      if (isBreak) {
        //when system current have a break
        if (current_time - previousReconnect[3] >= break_delay_time)
{
          previousReconnect[3] = 0;
          //when breaking time is out
          isBreak = false;
          Serial.println("Set off break ");
          previosBreak = true;
        }
      } else if (previosBreak || resentlyWorkedTime == 0 ||
(current_time - resentlyWorkedTime) > setResentleTimeAlowed) {
        //if the system was working recently but if it was break
        if (previosBreak) {
          previosBreak = false;
          //end set 0 to resently worked time
          resentlyWorkedTime = 0;
        }
        if (AIR_PROTOCOL == G_IN_G_Out) {
          //Goood_IN_Good_Out protocol section
          if ((insideTemperature - outsideTemperature) > 2) {
            //if all good but we can have great temperature result we
turn on the system
            setCoolerSystem(true);
            start_cooling_time = current_time;
            previousReconnect[1] = current_time;
            previus_inside_temperature = insideTemperature;
          }
        }
      }
    } else {
      //if system is runing that we
      //calling the function to controll active cooling state
      //if   cooling   has   been   worked   more   than   10min   =
check_productivity_delay
      if (AIR_PROTOCOL == G_IN_G_Out) {
        Serial.println("AIR_PROTOCOL : G_IN_G_Out");
        if  ((insideTemperature  -  outsideTemperature)  >=  0  &&
(insideTemperature - outsideTemperature) <= 3) {
          setCoolerSystem(false);
          resentlyWorkedTime = current_time;
```

```
          productivityIsSet = false;
        } else if ((insideTemperature - outsideTemperature) < 0) {
          setCoolerSystem(false);
          resentlyWorkedTime = current_time;
          productivityIsSet = false;
        }   else   if   (current_time   -   previousReconnect[1]   >=
check_productivity_delay) {
          previousReconnect[1] = current_time;
          //check data delay for see productivity
          Serial.println("Cheking productivity");
          if (!productivityIsSet) {
            if (previus_inside_temperature > insideTemperature) {
              productivity    =    (previus_inside_temperature    -
insideTemperature);
              productivityIsSet = true;
              previus_inside_temperature = insideTemperature;
              productivitySetCoutner = 2;
            } else {
              productivity    =    (previus_inside_temperature    -
insideTemperature);
              productivityIsSet = true;
              previus_inside_temperature = insideTemperature;
              productivitySetCoutner = 1;
            }
            Serial.print("  Set productivity counter: ");
            Serial.println(productivitySetCoutner);
          } else {
            if (previus_inside_temperature > insideTemperature) {
              if ((previus_inside_temperature - insideTemperature) >
productivity) {
                productivity    =    (previus_inside_temperature    -
insideTemperature);
                previus_inside_temperature = insideTemperature;
                productivitySetCoutner += 1;
                Serial.print("Productivity counter grows +1: ");
                Serial.println(productivitySetCoutner);
              } else {
                productivity    =    (previus_inside_temperature    -
insideTemperature);
                previus_inside_temperature = insideTemperature;
                productivitySetCoutner -= 1;
                Serial.print("Productivity counter reduce -1: ");
                Serial.println(productivitySetCoutner);
              }
            } else {
              productivitySetCoutner -= 1;
              Serial.print("Productivity counter reduce -1: ");
              Serial.println(productivitySetCoutner);
            }
          }
        }
        if ((current_time - start_cooling_time) > max_working_time)
{
          Serial.println("Set break ---");
```

```
          isBreak = true;
          setCoolerSystem(false);
          previousReconnect[3] = current_time;
          resentlyWorkedTime = current_time;
        } else if (productivitySetCoutner <= 0 && productivityIsSet)
{
          //if cooling is not effective we turn off system
          Serial.println("Cooling is not productive");
          productivityIsSet = false;
          setCoolerSystem(false);
          resentlyWorkedTime = current_time;
        }
      } else {
        //Good_IN_Bad_Out
        setCoolerSystem(false);
        resentlyWorkedTime = current_time;
        productivityIsSet = false;
      }
    }
  } else if (AIR_PROTOCOL == B_IN_G_Out || AIR_PROTOCOL == B_IN_B_Out)
{
    //if all is not good
    if (!isRuning) {
      //check system is it runing?
      if (isBreak) {
        //when system current have a break
        if (current_time - previousReconnect[3] >= break_delay_time)
{
          previousReconnect[3] = 0;
          //when breaking time is out
          isBreak = false;
          Serial.println("Set off break ");
          previosBreak = true;
        }
      } else if ((insideTemperature - outsideTemperature) > 1 ||
previosBreak || resentlyWorkedTime == 0 || (current_time -
resentlyWorkedTime) > setResentleTimeAlowed) {
        //if the system was working recently but if it was break
        if (previosBreak) {
          previosBreak = false;
          //end set 0 to resently worked time
          resentlyWorkedTime = 0;
        }
        if (AIR_PROTOCOL == B_IN_G_Out) {
          //Bad_IN_Good_Out protocol section
          if ((insideTemperature - outsideTemperature) > 1) {
            //if all good but we can have great temperature result we
turn on the system
            setCoolerSystem(true);
            start_cooling_time = current_time;
            previus_inside_temperature = insideTemperature;
            previousReconnect[1] = current_time;
          }
        } else {
```

```
          //Bad_IN_Bad_Out protocol section
          if ((insideTemperature - outsideTemperature) > 2) {
            //if all good but we can have great temperature result we
turn on the system
            setCoolerSystem(true);
            start_cooling_time = current_time;
            previus_inside_temperature = insideTemperature;
            previousReconnect[1] = current_time;
          }
        }
      }
    } else {
      //if system is runing that we
      //calling the function to controll active cooling state
      if (AIR_PROTOCOL == B_IN_G_Out) {
        Serial.println("AIR_PROTOCOL : B_IN_G_Out");
        //Bad_INide_Good_OUTside
        if     (current_time     -     previousReconnect[1]     >=
check_productivity_delay) {
          previousReconnect[1] = current_time;
          //check data delay for see productivity
          Serial.println("Cheking productivity");
          if (!productivityIsSet) {
            if (previus_inside_temperature > insideTemperature) {
              productivity    =    (previus_inside_temperature    -
insideTemperature);
              productivityIsSet = true;
              previus_inside_temperature = insideTemperature;
              productivitySetCoutner = 5;
            } else {
              productivity    =    (previus_inside_temperature    -
insideTemperature);
              productivityIsSet = true;
              previus_inside_temperature = insideTemperature;
              productivitySetCoutner = 3;
            }
            Serial.print("Set productivity counter: ");
            Serial.println(productivitySetCoutner);
          } else {
            if (previus_inside_temperature > insideTemperature) {
              if ((previus_inside_temperature - insideTemperature) >
productivity) {
                productivity    =    (previus_inside_temperature    -
insideTemperature);
                previus_inside_temperature = insideTemperature;
                productivitySetCoutner += 1;
                Serial.print("Productivity counter grows +1: ");
                Serial.println(productivitySetCoutner);
              } else {
                productivity    =    (previus_inside_temperature    -
insideTemperature);
                previus_inside_temperature = insideTemperature;
                productivitySetCoutner -= 1;
                Serial.print("Productivity counter reduce -1: ");
```

```
              Serial.println(productivitySetCoutner);
            }
          } else {
            productivitySetCoutner -= 2;
            Serial.print("Productivity counter reduce -2: ");
            Serial.println(productivitySetCoutner);
          }
        }
      }
      if ((current_time - start_cooling_time) > max_working_time)
{
        Serial.println("Set break ---");
        isBreak = true;
        setCoolerSystem(false);
        previousReconnect[3] = current_time;
        resentlyWorkedTime = current_time;
      } else if (productivitySetCoutner <= 0 && productivityIsSet)
{
        //if cooling is not effective we grows up voltage in
cooler`s relay
        Serial.println("Cooling is not productive");
        productivityIsSet = false;
        resentlyWorkedTime = current_time;
        setCoolerSystem(false);
      }
    } else {
      Serial.println("AIR_PROTOCOL : B_IN_B_Out");
      //Bad_INide_Bad_OUTside
      if ((insideTemperature - outsideTemperature) > 1) {
        //if all good but we can have great temperature result we
turn on the system
        if     (current_time     -     previousReconnect[1]     >=
check_productivity_delay) {
          previousReconnect[1] = current_time;
          //check data delay for see productivity
          Serial.println("Cheking productivity");
          if (!productivityIsSet) {
            if (previus_inside_temperature > insideTemperature) {
              productivity    =    (previus_inside_temperature    -
insideTemperature);
              productivityIsSet = true;
              previus_inside_temperature = insideTemperature;
              productivitySetCoutner = 2;
            } else {
              productivity    =    (previus_inside_temperature    -
insideTemperature);
              productivityIsSet = true;
              previus_inside_temperature = insideTemperature;
              productivitySetCoutner = 1;
            }
            Serial.print("Set productivity counter: ");
            Serial.println(productivitySetCoutner);
          } else {
            if (previus_inside_temperature > insideTemperature) {
```

```
                if ((previus_inside_temperature - insideTemperature)
> productivity) {
                  productivity   =   (previus_inside_temperature   -
insideTemperature);
                  previus_inside_temperature = insideTemperature;
                  productivitySetCoutner += 1;
                  Serial.print("Productivity counter grows +1: ");
                  Serial.println(productivitySetCoutner);
                } else {
                  productivity   =   (previus_inside_temperature   -
insideTemperature);
                  previus_inside_temperature = insideTemperature;
                  productivitySetCoutner -= 1;
                  Serial.print("Productivity counter reduce -1: ");
                  Serial.println(productivitySetCoutner);
                }
              } else {
                productivitySetCoutner -= 2;
                Serial.print("Productivity counter reduce -2: ");
                Serial.println(productivitySetCoutner);
              }
            }
          }
          if ((current_time - start_cooling_time) > max_working_time)
{
            //set cooler in a pause
            Serial.println("Set break --- ");
            isBreak = true;
            setCoolerSystem(false);
            previousReconnect[3] = current_time;
            resentlyWorkedTime = current_time;
          }    else    if    (productivitySetCoutner    <=    0    &&
productivityIsSet) {
            //if cooling is not effective we grows up voltage in
cooler`s relay
            Serial.println("Cooling is not productive");
            productivityIsSet = false;
            resentlyWorkedTime = current_time;
            setCoolerSystem(false);
          }
        } else {
          //if enviroment is bad we turn off all and wait
          Serial.println("Enviroment is bad turn off cooling system
");
          setCoolerSystem(false);
          productivityIsSet = false;
          resentlyWorkedTime = current_time;
        }
      }
    }
  } else if (AIR_PROTOCOL == LOWER_T_MIN) {
    //if enviroment is bad we turn off all and wait
    if (isRuning) {
```

```
      Serial.println("Inside temperature lover than
MIN_TEMPERATURE");
      setCoolerSystem(false);
      productivityIsSet = false;
      resentlyWorkedTime = current_time;
    }
  } else {
    Serial.println("AnalyzeAirStateProtocol error");
    return 1;
  }
  return 0;
}
void settingsMenu() {
  String menuItems[9] = {
    "min T",
    "max T",
    "Connect",
    "max w t",
    "Break t",
    "Res. w t",
    "Data r t",
    "Check p t",
    "Connect t"
  };
  unsigned long int itemsValue[9] = {
    min_temperature,
    max_temperture,
    int(canConnect),
    max_working_time / 1000,
    break_delay_time / 1000,
    setResentleTimeAlowed / 1000,
    get_sensor_data_delay / 1000,
    check_productivity_delay / 1000,
    http_request_delay / 1000
  };
  setSettings = true;
  int currentOptin = 0;
  bool saveSetting = false;
  bool isGhanging = false;
  bool itemIsSelected = false;
  bool getChange = true;
  bool prevCoolingSysState = false;
  displayData("Settings");
  if (isOn) {
    if (isRuning) {
      setCoolerSystem(false);
      //resentlyWorkedTime = currentMillis;
    }
    if (isYellowBlynk) {
      digitalWrite(YELLOW_D, LOW);
      isYellowBlynk = false;
    }
    if (isGreenBlynk) {
      digitalWrite(GREEN_D, LOW);
```

```
    isGreenBlynk = false;
  }
  if (isBlynk) {
    digitalWrite(BLUE_D, LOW);
    isBlynk = false;
  }
}
delay(1000);
while (setSettings) {
  /*-----------------------------------------------------------------
    Reading all sensors inputs from "n"
*/
  pollSensors(0);
  //init new variable for analyze preveous state
  static int lastSensorStatus[SENSOR_BUTTONS];
  int sensorStatus[SENSOR_BUTTONS];
  //checking sensor state
  for (int btn = 0; btn < SENSOR_BUTTONS; btn++) {
    sensorStatus[btn] = getLastPolledSensorStatus(btn);
    if (sensorStatus[btn] != lastSensorStatus[btn]) {
      lastSensorStatus[btn] = sensorStatus[btn];
      //regisrtate button change
      getChange = true;
      //exit from settings
      if (sensorStatus[3] == BUTTON_LONG_PRESS) {
        setSettings = false;
        if (isGhanging) {
          displayData("Save changes?");
          while (1) {
            pollSensors(0);
            sensorStatus[2] = getLastPolledSensorStatus(2);
            if (sensorStatus[2] != lastSensorStatus[2]) {
              lastSensorStatus[2] = sensorStatus[2];
              if (sensorStatus[2] == BUTTON_SHORT_PRESS) {
                saveSetting = true;
                getChange = false;
                displayData("Ok");
                break;
              }
            }
            sensorStatus[3] = getLastPolledSensorStatus(3);
            if (sensorStatus[3] != lastSensorStatus[3]) {
              lastSensorStatus[3] = sensorStatus[3];
              if (sensorStatus[3] == BUTTON_SHORT_PRESS) {
                saveSetting = false;
                getChange = false;
                displayData("No");
                break;
              }
            }
            delay(10);
          }
          delay(600);
        }
```

```
          break;
        }
        //if pressed on up or down buttons
        if (!itemIsSelected && (sensorStatus[0] == BUTTON_SHORT_PRESS
|| sensorStatus[1] == BUTTON_SHORT_PRESS)) {
          if (sensorStatus[0] == BUTTON_SHORT_PRESS) {
            //if ressed   up ">" button
            currentOptin--;
            if (currentOptin < 0)
              currentOptin = 8;
          } else {
            //if ressed down "<" button
            currentOptin++;
            if (currentOptin > 8)
              currentOptin = 0;
          }
        }
        if (itemIsSelected && (sensorStatus[0] == BUTTON_SHORT_PRESS
|| sensorStatus[1] == BUTTON_SHORT_PRESS)) {
          if (sensorStatus[0] == BUTTON_SHORT_PRESS) {
            //if ressed   up ">" button
            if (!isGhanging)
              isGhanging = true;
            isSettingSet = true;
            if (currentOptin == 2) {
              itemsValue[currentOptin] = 1;
            } else {
              if (itemsValue[currentOptin] < 99) {
                itemsValue[currentOptin] += 1;
              }
            }
          } else {
            //if ressed down "<" button
            if (!isGhanging)
              isGhanging = true;
            if (currentOptin == 2) {
              itemsValue[currentOptin] = 0;
            } else {
              if (itemsValue[currentOptin] > 1) {
                itemsValue[currentOptin] -= 1;
              }
            }
          }
        }
        if (sensorStatus[2] == BUTTON_SHORT_PRESS) {
          itemIsSelected = !itemIsSelected;
        }
        Serial.print(btn);
        Serial.print("-");
        showSensorStatus(sensorStatus[btn]);
      }
    }
    //show changes when that is
    if (getChange) {
```

```
      if (itemIsSelected) {
        oled.clearDisplay();   // clear display
        oled.setCursor(0, 0);  // set position to display (x,y)
        oled.println("*" + menuItems[currentOptin]);
        oled.setCursor(0, 16);                                    //
set position to display (x,y)
        oled.println(">" + String(itemsValue[currentOptin]) + "<");
// set text
        oled.display();
      } else {
        oled.clearDisplay();   // clear display
        oled.setCursor(0, 0);  // set position to display (x,y)
        oled.println(menuItems[currentOptin]);
        oled.setCursor(0, 16);                                  // set
position to display (x,y)
        oled.println(String(itemsValue[currentOptin]));  // set text
        oled.display();
      }
      getChange = false;
    }
    /*-------------------------------------------------------------
      reading all sensors inputs from "n"
  */
    delay(10);
  }
  if (saveSetting) {
    min_temperature = itemsValue[0];
    max_temperture = itemsValue[1];
    canConnect = itemsValue[2] != 0;
    max_working_time = itemsValue[3] * 1000;
    break_delay_time = itemsValue[4] * 1000;
    setResentleTimeAlowed = itemsValue[5] * 1000;
    get_sensor_data_delay = itemsValue[6] * 1000;
    check_productivity_delay = itemsValue[7] * 1000;
    http_request_delay = itemsValue[8] * 1000;
    saveSetting = false;
  }
  //exit from menu
  displayData("Exit");
  delay(500);
  //Clearing oled display
  displayData("");
}
void displayAirData(float in_T, int in_H, float out_T, int out_H) {
  oled.clearDisplay();
  oled.setCursor(0, 0);  // set position to display (x,y)
  oled.setTextSize(5);
  oled.print(String(int(in_T)));
  oled.setTextSize(2);
  oled.print((char)247);
  oled.setTextSize(5);
  oled.print("C");
  oled.setCursor(103, 5);
  oled.setTextSize(2);
```

```
  oled.print(String(int(in_H)));
  oled.setCursor(108, 24);
  oled.print("%");
  oled.setCursor(4, 38);
  oled.setTextSize(1);
  oled.print("Outside temp");
  oled.setTextSize(2);
  oled.setCursor(4, 48);
  oled.print(String(int(out_T)));
  oled.setTextSize(1);
  oled.print((char)247);
  oled.setTextSize(2);
  oled.print("C ");
  oled.print(String(int(out_H)));
  oled.print("%");
  oled.display();
  isDisplayingDigit = true;
}
String httpGETRequest() {
  HTTPClient http;
  http.begin(serverName);
  http.addHeader("Content-Type", "application/json");
  String                        httpRequestData                              =
String("{\"secure\":\"41495100\",\"isWorking\":\"")  +  int(isRuning)
+    "\",\"isOn\":\""    +    int(isOn)    +    "\",\"minTemp\":\""    +
min_temperature    +    "\",\"maxTemp\":\""    +    max_temperture    +
"\",\"settingState\":\""           +           isSettingSet           +
"\",\"insideTemperature\":\""       +       insideTemperature       +
"\",\"insideHumidity\":\""         +         insideHumidity         +
"\",\"outsideTemperature\":\""      +      outsideTemperature      +
"\",\"outsideHumidity\":\"" + outsideHumidity + "\"}";
  // Send HTTP POST request
  int httpResponseCode = http.POST(httpRequestData);
  String payload = "{}";
  if (httpResponseCode > 0) {
    Serial.print("HTTP Response code: ");
    Serial.println(httpResponseCode);
    payload = http.getString();
    Serial.println(payload);
    if (isSettingSet) {
      isSettingSet = false;
    }
  } else {
    Serial.print("Error code: ");
    Serial.println(httpResponseCode);
  }
  http.end();
  return payload;
}
void jsonParsing(String str) {
  JSONVar myObject = JSON.parse(str);
  if (JSON.typeof(myObject) == "undefined") {
    Serial.println("Parsing input failed!");
    return;
```

```
  }
  if (JSON.typeof(myObject) == "") {
    return;
  }
  Serial.print("JSON object = ");
  Serial.println(myObject);
  JSONVar keys = myObject.keys();
  for (int i = 0; i < keys.length(); i++) {
    JSONVar value = myObject[keys[i]];
    Serial.print((const char*)keys[i]);
    Serial.print(" = ");
    Serial.println(value);
    if (strcmp((const char*)keys[i], "isOn") == 0) {
      isOn = int(value) == 1 ? true : false;
    } else if (strcmp((const char*)keys[i], "minTemp") == 0) {
      min_temperature = int(value);
    } else if (strcmp((const char*)keys[i], "maxTemp") == 0) {
      max_temperture = int(value);
    }
  }
}
int pos = 0;
void loop() {
  const unsigned long interval[3] = { get_sensor_data_delay,
check_productivity_delay, http_request_delay };
  //time point to reading and comparing
  unsigned long currentMillis = millis();
  /*-----------------------------------------------------------
    Checking sensors data section every 10 sec.
  */
  if (isOn && (currentMillis - previousReconnect[0] >= interval[0]))
{
    previousReconnect[0] = currentMillis;
    //if data reads correct
    if (getSensorData() == 0) {
      Serial.println("Check dht data ");
      if (getDataErr) {
        getDataErr = false;
        insideDhtErr = false;
        outsideDhtErr = false;
      }
      displayAirData(insideTemperature,              insideHumidity,
outsideTemperature, outsideHumidity);
      //analyzing air enviroment to decide when on/off the coling
system
      analyzeAirStateProtocol(currentMillis);
    } else {
      //DisplayDigit(22);
      String dhtSensor;
      if (outsideDhtErr) {
        dhtSensor = "OUT_DHT Error!";
      } else if (insideDhtErr) {
        dhtSensor = "IN_DHT Error!";
      }
```

```
    Serial.println(dhtSensor);
    displayData(dhtSensor);
    getDataErr = true;
    if (isRuning) {
      setCoolerSystem(false);
      //resentlyWorkedTime = currentMillis;
      productivityIsSet = false;
    }
  }
} else if (!isOn) {
  if (isRuning) {
    setCoolerSystem(false);
    productivityIsSet = false;
  }
  if (isDisplayingDigit) {
    oled.clearDisplay();
    oled.display();
  }
}
/*-----------------------------------------------------------------
  Checking sensors data end.
*/
/*-----------------------------------------------------------------
  Reading all sensors inputs from "n"
*/
pollSensors(3);
static int lastSensorStatus[SENSOR_BUTTONS];
int sensorStatus[SENSOR_BUTTONS];
//checking sensor state
for (int btn = 3; btn < SENSOR_BUTTONS; btn++) {
  sensorStatus[btn] = getLastPolledSensorStatus(btn);
  if (sensorStatus[btn] != lastSensorStatus[btn]) {
    lastSensorStatus[btn] = sensorStatus[btn];
    //turn on/off touch button
    if (sensorStatus[4] == BUTTON_LONG_PRESS) {
      isOn = !isOn;
      isSettingSet = true;
      if (isOn == true) {
        displayData("System On");
        delay(500);
      } else {
        displayData("System off");
        delay(500);
      }
      Serial.print("isOn ");
      Serial.println(isOn);
    }
    //turn settings touch button
    if (sensorStatus[3] == BUTTON_LONG_PRESS) {
      settingsMenu();
    }
    Serial.print(btn);
    Serial.print("-");
    showSensorStatus(sensorStatus[btn]);
```

```
    }
  }
  /*-----------------------------------------------------------------
    reading all sensors inputs from "n"
  */
  if (isOn) {
    if (getDataErr && !isYellowBlynk && (currentMillis -
previousReconnect[6] >= yellowLedBlynkDelay)) {
      previousReconnect[6] = currentMillis;
      digitalWrite(YELLOW_D, HIGH);
      isYellowBlynk = true;
      previousReconnect[7] = currentMillis;
    } else if (isYellowBlynk && getDataErr && (currentMillis -
previousReconnect[7] >= setLedBlynkTimeActive)) {
      digitalWrite(YELLOW_D, LOW);
      isYellowBlynk = false;
    }
    if (!isBlynk && (currentMillis - previousReconnect[4] >=
ledBlynkDelay)) {
      previousReconnect[4] = currentMillis;
      digitalWrite(BLUE_D, HIGH);
      isBlynk = true;
      previousReconnect[5] = currentMillis;
    } else if (isBlynk && (currentMillis - previousReconnect[5] >=
setLedBlynkTimeActive)) {
      digitalWrite(BLUE_D, LOW);
      isBlynk = false;
    }
  } else {
    if (isYellowBlynk) {
      digitalWrite(YELLOW_D, LOW);
      isYellowBlynk = false;
    }
    if (isGreenBlynk) {
      digitalWrite(GREEN_D, LOW);
      isGreenBlynk = false;
    }
    if (isBlynk) {
      digitalWrite(BLUE_D, LOW);
      isBlynk = false;
    }
  }
  /*-----------------------------------------------------------------
    http.get section with timer delay send requets every 10 seconds
  */
  if (canConnect && !getDataErr && (currentMillis -
previousReconnect[2] >= interval[2])) {
    previousReconnect[2] = currentMillis;
    if (WiFi.status() != WL_CONNECTED) {
      // simple reconnect doesn't ever succeed on Wokwi
      //displaying data on oled
      displayData("Reconnecting WiFi...");
      Serial.println("Reconnecting WiFi...");
      WiFi.disconnect();
```

```
      WiFi.reconnect();
    }
    if (WiFi.status() == WL_CONNECTED) {
      //http request sending section
      String responseData = httpGETRequest();
      jsonParsing(responseData);
      Serial.println("Sending get response request to server ...");
    }
  }
  /*-----------------------------------------------------------------
    http.get section end
  */
  delay(10);
}
```

Додаток Е

Лістинг коду клієнтської частини

```
import axios from "axios";
const BASE_URL = "http://localhost:5500";
export default axios.create({
  baseURL: BASE_URL,
});
export const axiosPrivate = axios.create({
  baseURL: BASE_URL,
  headers: { ContentType: "application/json" },
  withCredentials: true,
});
import React from "react";
import "./model.css";
const Model = ({ children, onSubmit }) => {

  return (
      <form className="model-container" onSubmit={onSubmit}>
        {children}
      </form>
  );
};
import React from "react";
import { useRef, useState, useEffect, useContext } from "react";
import {
  Link,
  useLocation,
  useNavigate,
  Navigate,
  Outlet,
} from "react-router-dom";
import useAuth from "../../hooks/useAuth";
import axios from "../../api/axios";

import {
  showErrMsg,
  showWarningMsg,
  showSucsessMsg,
} from "../../msgpopup/ShowPopup";
import Model from "../customModel/Model";
import ModelHeader from "../customModel/ModelHeader";
import ModelContent from "../customModel/ModelContent";
import ModelFooter from "../customModel/ModelFooter";
import VerticalLine from "../customModel/VerticalLine";
const LOGIN_URL = "/login";
const LoginForm = () => {
  const { setAuth } = useAuth();
  const location = useLocation();
  const navigate = useNavigate();
  const userRef = useRef();
```

```
const [username, setUserName] = useState("");
const [pwd, setPwd] = useState("");
const [errMsg, setErrMsg] = useState("");
const [success, setSuccess] = useState(false);
useEffect(() => {
  userRef.current.focus();
}, []);
const handleSubmit = async (e) => {
  e.preventDefault();
  try {
    const response = await axios.post(
      LOGIN_URL,
      JSON.stringify({ username: username, password: pwd }),
      {
        headers: {
          "Content-Type": "application/json",
          "Access-Control-Allow-Origin": "*",
        },
        withCredentials: true,
      }
    );
    console.log(JSON.stringify(response));
    const accessToken = response?.data?.accessToken;
    const roles = response?.data?.roles;
    setAuth({ username, roles, accessToken });
    setSuccess(true);
    setUserName("");
    navigate("/main", {
      state: { from: location.pathname },
      replace: true,
    });
    setPwd("");
    //navigate(from, {replace: true});
  } catch (err) {
    if (!err?.response) {
      showErrMsg("Немає відповіді від серверу");
    } else if (err.response?.status == 400) {
      showWarningMsg("Неправильний e-mail, або пароль");
    } else if (err.response?.status == 401) {
      showWarningMsg("Схоже, що ви ще не зареєстровані");
    } else {
      showErrMsg("Виникла невідома помилка");
    }
  }
};
const handleToRegister = () => {
  navigate("/register", {
    state: { from: location.pathname },
    replace: true,
  });
};
return (
  <Model onSubmit={handleSubmit}>
    <ModelHeader>Вхід в акаунт</ModelHeader>
```

```
      <ModelContent>
        <input
          type="text"
          placeholder="Ваше ім'я"
          ref={userRef}
          onChange={(e) => setUserName(e.target.value)}
          value={username}
          required
        />
        <input
          type="password"
          placeholder="Ваш ароль"
          onChange={(e) => setPwd(e.target.value)}
          value={pwd}
          required
        />
      </ModelContent>
      <ModelFooter>
        <input
          type="button"
          value="Зареєструватись"
          onClick={handleToRegister}
        />
        <VerticalLine />
        <input type="submit" value="Увійти" />
      </ModelFooter>
    </Model>
  );
};
export default LoginForm;
export default Model;
import { useEffect, useState, useRef } from "react";
import "./nav-mobile.css";
const NavMobile = ({addOption, autoOption, settingOption,
onChangeAdd, onChangeAuto, onChangeSetting }) => {
  return (
    <nav>
      <div className="nav-content">
        <label>
          <input
            type="checkbox"
            onChange={onChangeAdd}
            checked={addOption}
          />
          <svg
            version="1.1"
            id="Layer_1"
            xmlns="http://www.w3.org/2000/svg"
            viewBox="0 0 50 50"
            height="30px"
          >
            <path
              class="st0"
```

```
              d="M35.5,0h-
21C5.4,0,0,5.4,0,14.5v20.9C0,44.6,5.4,50,14.5,50h20.9C44.6,50,50,44.
6,50,35.5v-21
     C50,5.4,44.6,0,35.5,0z M35,26.9h-8.1V35c0,1-0.9,1.9-1.9,1.9c-
1,0-1.9-0.8-1.9-1.9v-8.1H15c-1,0-1.9-0.9-1.9-1.9s0.9-1.9,1.9-1.9
     h8.1V15c0-1,0.9-1.9,1.9-
1.9c1,0,1.9,0.9,1.9,1.9v8.1H35c1,0,1.9,0.9,1.9,1.9S36,26.9,35,26.9z"
            />
          </svg>
          <p className={`${addOption ? "active" : ""}`}>Додати</p>
        </label>
        <label>
          <input
            type="checkbox"
            onChange={onChangeAuto}
            checked={autoOption}
          />
          <svg
            version="1.1"
            id="Layer_1"
            xmlns="http://www.w3.org/2000/svg"
            viewBox="0 0 50 50"
            height="30px"
          >
            <path
              class="st0"
              d="M50,21.8c0-2.7-1.4-5.1-3.5-6.5L29.8,1.6c-2.7-2.2-7-
2.2-9.7,0L3.6,15.2C1.4,16.6,0,19,0,21.8
     C0,22,0,42.2,0,42.2C0,46.5,3.4,50,7.7,50h34.7c4.2,0,7.7-
3.5,7.7-7.8 M18.5,38.2c-1.4,0-2.5-1.1-2.5-2.5c0-1.4,1.1-2.5,2.5-2.5
     c1.4,0,2.5,1.1,2.5,2.5C21,37.1,19.8,38.2,18.5,38.2z
M26,37.6C26,37.6,26,37.6,26,37.6c-1.1,0-1.9-0.8-1.9-1.9
     c-0.1-3.4-2.3-5.7-5.7-5.8c-1,0-1.9-0.9-1.8-2c0-1,0.9-1.9,1.9-
1.9c0,0,0,0,0.1,0c5.4,0.1,9.2,4,9.3,9.5C27.8,36.7,27,37.6,26,37.6z
      M33.5,37.6c-1.1,0-1.9-0.8-1.9-1.9c0-1.7-0.3-3.3-0.8-4.8c-1.3-
3.7-4-6.5-7.7-7.8c-1.5-0.5-3.1-0.8-4.7-0.8c-1,0-1.9-0.9-1.9-1.9
     c0-1,0.9-1.9,1.9-
1.9h0c2.1,0,4.1,0.4,5.9,1c4.7,1.7,8.3,5.3,9.9,10.1c0.7,1.9,1,3.9,1,6
C35.3,36.7,34.5,37.6,33.5,37.6z"
            />
          </svg>
          <p className={`${autoOption ? "active" :
""}`}>Автоматизація</p>
        </label>
        <label>
          <input
            type="checkbox"
            onChange={onChangeSetting}
            checked={settingOption}
          />
          <svg
            version="1.1"
            id="Layer_1"
            xmlns="http://www.w3.org/2000/svg"
```

```
            viewBox="0 0 50 50"
            height="30px"
          >
            <path
              class="st0"
              d="M3.9,29.4l-2.6,1.4c-1.3,0.7-1.7,2.2-
1,3.4l3.9,6.4c0.7,1.2,2.3,1.6,3.6,0.9l2.7-1.5c1.6-0.9,3.6-0.9,5.2,0
    l0.1,0c1.6,0.9,2.6,2.5,2.6,4.3v3c0,1.4,1.2,2.5,2.6,2.5H29c1.5,0
,2.6-1.1,2.6-2.5v-3c0-1.8,1-3.4,2.6-4.3l0.1,0
    c1.6-0.9,3.6-0.9,5.2,0l2.7,1.5c1.3,0.7,2.9,0.3,3.6-0.9l3.9-
6.4c0.7-1.2,0.3-2.7-1-3.4c0,0,0,0,0,0L46,29.4
    c-1.6-0.9-2.6-2.5-2.6-4.3V25c0-1.8,1-3.4,2.6-4.3l2.6-1.4c1.3-
0.7,1.7-2.2,1-3.4l-3.9-6.4c-0.7-1.2-2.3-1.6-3.6-0.9l-2.7,1.5
    c-1.6,0.9-3.6,0.9-5.2,0l0,0c-1.6-0.9-2.6-2.5-2.6-4.3v-3c0-1.4-
1.2-2.5-2.6-2.5H21c-1.5,0-2.6,1.1-2.6,2.5v3.2c0,1.7-0.9,3.2-2.4,4
    L15.6,10c-1.5,0.8-3.4,0.8-4.9,0L7.9,8.4C6.6,7.7,5,8.1,4.3,9.3l-
3.9,6.4c-0.7,1.2-0.3,2.7,1,3.4l2.6,1.4c1.6,0.9,2.6,2.5,2.6,4.3
    V25C6.6,26.8,5.6,28.5,3.9,29.4z
M25,17.5c4.4,0,7.9,3.4,7.9,7.5s-3.5,7.5-7.9,7.5s-7.9-3.4-7.9-
7.5S20.6,17.5,25,17.5z"
            />
          </svg>
          <p className={`${settingOption ? "active" :
""}`}>Параметри</p>
        </label>
      </div>
    </nav>
  );
};
export default NavMobile;
import { Outlet } from "react-router-dom";
import { useState, useEffect } from "react";
import useRefreshToken from "../../hooks/useRefreshToken";
import useAuth from "../../hooks/useAuth";
const PersistLogin = () => {
    const [isLoading, setIsLoading] = useState(true);
    const refresh = useRefreshToken();
    const {auth} = useAuth();
    useEffect(() => {
        let isMounted = true;
        const verifyRefreshToken = async () => {
            try {
                await refresh();
            } catch (error) {
                console.log(error);
            }
            finally {
                isMounted && setIsLoading(false);
            }
        }
        !auth.accessToken ? verifyRefreshToken() :
setIsLoading(false);
        return () => isMounted = false;
    }, []);
```

```
    useEffect(() => {
        console.log(`isLoading: ${isLoading}`);
        console.log(`aT: ${JSON.stringify(auth?.accessToken)}`);
    }, [isLoading]);
    return (
        <>
            {isLoading
                ? <p> Loading ...</p>
                : <Outlet />
            }
        </>
    )
}
export default PersistLogin;
import React from "react";
import { useRef, useState, useEffect, useContext } from "react";
import {Link,useLocation,useNavigate,Navigate,Outlet,
} from "react-router-dom";
import useAuth from "../../hooks/useAuth";
import axios from "../../api/axios";
import {
  showErrMsg,showWarningMsg,showSucsessMsg,
} from "../../msgpopup/ShowPopup";
import Model from "../customModel/Model";
import ModelHeader from "../customModel/ModelHeader";
import ModelContent from "../customModel/ModelContent";
import ModelFooter from "../customModel/ModelFooter";
import VerticalLine from "../customModel/VerticalLine";
const REGISTER_URL = "/register";
const RegisterForm = () => {
  const { setAuth } = useAuth();
  const location = useLocation();
  const navigate = useNavigate();
  const userRef = useRef();
  const [username, setUserName] = useState("");
  const [password, setPassword] = useState("");
  const [success, setSuccess] = useState(false);
  useEffect(() => {
    userRef.current.focus();
  }, []);
  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      const response = await axios.post(
        REGISTER_URL,
        JSON.stringify({ username: username, password: password }),
        {
          headers: {
            "Content-Type": "application/json",
          },
          withCredentials: true,
        }
      );
      console.log(JSON.stringify(response));
```

```
      const accessToken = response?.data?.accessToken;
      const status = response?.status;
      setAuth({ username, status, accessToken });
      setSuccess(true);
      showSucsessMsg("Ви успішно зареєструвались");
      setTimeout(() => {
        navigate("/login", {
          state: { from: location.pathname },
          replace: true,
        });
      }, 1000);
      setUserName("");
      setPassword("");
      //navigate(from, {replace: true});
    } catch (err) {
       showErrMsg("Виникла невідома помилка");
      }
    }
  };
  const handleToLogin = () => {
    navigate("/login", { state: { from: location.pathname },
replace: true });
  };
  return (
    <Model onSubmit={handleSubmit}>
      <ModelHeader>Зареєструватися</ModelHeader>
      <ModelContent>
        <input
          type="text"
          placeholder="Ваше ім'я"
          ref={userRef}
          onChange={(e) => setUserName(e.target.value)}
          value={username}
          required
        />
        <input
          type="password"
          placeholder="Ваш пароль"
          onChange={(e) => setPassword(e.target.value)}
          value={password}
          required
        />
      </ModelContent>
      <ModelFooter>
        <input type="button" value="Увійти" onClick={handleToLogin}
/>
        <VerticalLine />
        <input type="submit" value="Зареєструватись" />
      </ModelFooter>
    </Model>
  );
};
export default RegisterForm;
import React from "react";
```

```
import { Outlet } from "react-router-dom";
import { useEffect, useState, useRef } from "react";
import { useNavigate } from "react-router-dom";
import Sensor from "../sensorSection/Sensor";
import useAxiosPrivate from "../../hooks/useAxiosPrivate";
import useAuth from "../../hooks/useAuth";
import "./room.css";
const SENSORS_URL="/sensors";
function Room(props) {
  const { setSelectedSensors } = useAuth();
  const axiosPrivate = useAxiosPrivate();
  const [controllers, setControllers] = useState([]);
  const sensorData = {
    id: "6f_7f-e7",
    type: "active",
    name: "Еко охолодження",
    tag: "cooling",
  };
  useEffect(() => {
    let isMounted = true;
    const interval = setInterval(() => {
      axiosPrivate
      .get(SENSORS_URL+`/${props.id}`)
      .then((response) => {
        console.log(response.data);
        if(isMounted)
        setControllers(response.data);
        setSelectedSensors(response.data);
      })
      .catch((err) => {
        console.log(err.response.data.message);
      })
    }, 15000);
    return () => {
      clearInterval(interval);
      isMounted = false;
    }
  },[]);
  useEffect(()=>{
    axiosPrivate
    .get(SENSORS_URL+`/${props.id}`)
    .then((response) => {
      console.log(response.data);
      setControllers(response.data);
    })
    .catch((err) => {
      console.log(err.response.data.message);
    })
  },[])
  return (
    <section className="room-section">
      <article id={props.id} onClick={props.onClick}>
        {props.roomName}
        <p>&gt;</p>
```

```
      </article>
      <section className="sensors-section">
      {controllers?.length != 0 ? (
        controllers.map((controller) => (
          <Sensor key={controller._id} id={controller._id}
{...controller} onClick={props.onSensorClick}/>
        ))
      ) : (
        <></>
      )}
      </section>
    </section>
  );
}
export default Room;
import React from "react";
import cooling_active from "../../assets/cooling_active.svg";
import cooling_off from "../../assets/cooling_off.svg";
import cooling_no_conect from "../../assets/cooling_no_conect.svg";
import { Outlet } from "react-router-dom";
import { useEffect, useState, useRef } from "react";
import { useNavigate } from "react-router-dom";
import useAxiosPrivate from "../../hooks/useAxiosPrivate";
import useAuth from "../../hooks/useAuth";
import {
  showErrMsg,showWarningMsg,showSucsessMsg,
} from "../../msgpopup/ShowPopup";
import "./sensor.css";
const Sensor = (props) => {
  const axiosPrivate = useAxiosPrivate();
  console.log("props", props);
  function diff_minutes(dt1) {
    let splitData = dt1.split(" ")
    console.log(splitData);
    let curentDate  =  new Date();
    // var diff = (curentDate.getTime() - dt1.getTime()) / 1000;
    // diff /= 60;
    // return Math.abs(Math.round(diff));
  }
  return (
    <>
      <section
        className="sensor-data-section "
        id={props._id}
        onClick={props.onClick}
      >
        <div className="header-sensor-section-container">
          <div className="animated-icon">
            <img
              className={
                props.state.isOn
                  ? props.state.isWorking
                    ? "rotation-animation animated-icon-img"
                    : "animated-icon-img"
```

```
                    : "animated-icon-img"
                }
                src={
                  // sensorData.conection ?
                  props.state.isOn ? cooling_active : cooling_off
                  //: cooling_no_conect
                }
                alt=""
                height="25px"
              />
            </div>
            <div className="main-sensor-data">
              {props.sensors.inside.temperature} &#xb0;C
            </div>
            <div className="aditional-sensor-data">
              <p className="aditional-
data">{props.sensors.inside.humidity} %</p>
              <p className="aditional-data-descryption">в середині</p>
            </div>
          </div>
          <div className="params-sensor-section-container">
            <p className="sensor-name">{props?.name}</p>
            <p className="active-param-rule">
              Зберігати темперетуру в межах {props.state.minTemp}-
              {props.state.maxTemp} &#xb0;C
            </p>
            {
              diff_minutes(props.state.lastConnection)
            }
          </div>
        </section>
      </>
    );
};
export default Sensor;
import MainSetting from "./MainSetting";
import RoomSettings from "./RoomSettings";
import SensorSettings from "./SensorSettings";
const SettingComponent = (props) => {
  return (
    <section className="setting">
      {props?.param ? (
        props.param == "roomParam" ? (
          <RoomSettings
            onChange={props.editedName}
            onClick={props.deleteSubmit}
            roomName={props.selectedRoomName}
            originRoomName={props.selectedOriginName}
          />
        ) : props.param == "sensorParam" ? (
          <SensorSettings
            onChangeMinTemp={props.onChangeMinTemp}
            onChangeMaxTemp={props.onChangeMaxTemp}
            currentMinTemp={props.currentMinTemp}
```

```
            currentMaxTemp={props.currentMaxTemp}
            currentIsOn={props.currentIsOn}
            onClick={props.onClickOnOff}
          />
        ) : (
          <MainSetting />
        )
      ) : (
        <MainSetting />
      )}
    </section>
  );
};
export default SettingComponent;
import { createContext, useState } from "react";
const AuthContext = createContext({});
export default AuthContext;
export const AuthProvider = ({ children }) => {
  const [auth, setAuth] = useState({});
  const [selectedSensors, setSelectedSensors] = useState([]);
  return (
    <AuthContext.Provider
      value={{
        auth,
        setAuth,
        selectedSensors,
        setSelectedSensors
      }}
    >
      {children}
    </AuthContext.Provider>
  );
};
import { Outlet } from "react-router-dom";
import { useState, useEffect } from "react";
import useRefreshToken from "../hooks/useRefreshToken";
import useAuth from "../hooks/useAuth";
const PersistLogin = () => {
    const [isLoading, setIsLoading] = useState(true);
    const refresh = useRefreshToken();
    const { auth } = useAuth();
    useEffect(() => {
        let isMounted = true;
        const verifyRefreshToken = async () => {
            try {
                await refresh();
            } catch (error) {
                console.log(error);
            }
            finally {
                isMounted && setIsLoading(false);
            }
        }
```

```
        !auth.accessToken ? verifyRefreshToken() :
setIsLoading(false);
        return () => isMounted = false;
    }, []);
    useEffect(() => {
        console.log(`isLoading: ${isLoading}`);
        console.log(`aT: ${JSON.stringify(auth?.accessToken)}`);
    }, [isLoading]);
    return (
        <>
            {isLoading
                ? <p> Loading ...</p>
                : <Outlet />
            }
        </>
    )
}
export default PersistLogin
import React, { useEffect} from "react";
import { Navigate, useNavigate,  Routes, Route } from "react-router-
dom";
import Layout from "./layouts/Layout";
import GuessLayout from "./layouts/GuessLayout";
import RequireAuth from "./provider/RequireAuth";
import PersistLogin from "./components/persistLogin/PersistLogin";
import { Main, NotFound, Login, Register } from "./pages";
import "./App.css";
import "./msgpopup/msgpopup.css";
function App() {
  const navigate = useNavigate();
  const ROLES_LIST = {
    Admin: 5001,
    Editor: 4001,
    User: 3001,
  };
  return (
    <Routes>
      <Route path="/" element={<Layout />}>
        <Route element={<GuessLayout />}>
          <Route path="login" element={<Login />} />
          <Route path="register" element={<Register />} />
        </Route>
        <Route element={<PersistLogin/>}>
          <Route element={<RequireAuth
allowedRoles={[5001,4001,3001]}/>}>
            <Route path="main" element={<Main />} />
          </Route>
        </Route>
        <Route path="*" element={<NotFound />} />
      </Route>
    </Routes>
  );
}
export default App;
```

Додаток Ж

Лістинг коду серверної частини

```
const User = require('../model/User');
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');
//require('dotenv').config();
const handleLogin = async (req, res) => {
    const {username, password} = req.body;
    if(!username || !password)
    return res.status(400).json({"message": "Username and password
are reqiured."});
    // check for duplicate usernames in the db
    const foundUser = await User.findOne({username:
username}).exec();
    if(!foundUser)
    return res.sendStatus(401); //Unauthorized
    //evaluate passwors
    const match = await bcrypt.compare(password, foundUser.password);
    if(match){
        const roles = Object.values(foundUser.roles);
        //create JWTs
        const accessToken = jwt.sign(
            {
                "UserInfo":{
                    "username": foundUser.username,
                    "roles": roles
                }
            },
            process.env.ACCESS_TOKEN_SECRET,
            {expiresIn: '60s'}
        );
        const refreshToken = jwt.sign(
            {"username": foundUser.username},
            process.env.REFRESH_TOKEN_SECRET,
            {expiresIn: '1d'}
        );
        //saving refresh token with current user
        foundUser.refreshToken = refreshToken;
        const result = await foundUser.save();
        console.log(result);

        res.cookie("jwt", refreshToken, {httpOnly: true, sameSite:
'None', secure: true,  maxAge: 24*60*60*1000}); //secure: true,
        res.json({ roles, accessToken });
    } else{
        res.sendStatus(401);
    }
}
module.exports = {handleLogin};
const User = require('../model/User');
```

```
const handleLogout = async (req, res) => {
    //On client, also delete the accessToken
    const cookies = req.cookies;
    if(!cookies?.jwt) return res.sendStatus(401); //no content
    const refreshToken = cookies.jwt;
    if(!refreshToken    ||    refreshToken    ===    '')    return
res.sendStatus(401);
    // is refresh token in db?
    const foundUser = await User.findOne({refreshToken}).exec();
    if(!foundUser){
        res.clearCookie("jwt", {httpOnly: true, sameSite: 'None'});
//secure: true - only an https
        return res.sendStatus(403);
    }

    //Delete refreshToken in the db
    foundUser.refreshToken = '';
    const result = await foundUser.save();
    console.log(result);
    res.clearCookie("jwt",  {httpOnly:  true,  sameSite:  'None'});
//secure: true - only an https
    res.sendStatus(204);
}
module.exports = {handleLogout}
const User = require('../model/User');
const jwt = require('jsonwebtoken');
//require('dotenv').config();
const handleRefreshToken = async (req, res) => {
    const cookies = req.cookies;
    if(!cookies?.jwt)
        return res.status(401);
    console.log(cookies?.jwt);
    const refreshToken = cookies.jwt;
    console.log(refreshToken );
    // check for duplicate usernames in the db
    const foundUser = await User.findOne({refreshToken}).exec();
    if(!foundUser)
        return res.sendStatus(403); //Forbidden
    //evaluate jwt
    jwt.verify(
        refreshToken,
        process.env.REFRESH_TOKEN_SECRET,
        (err, decoded) => {
            if(err || foundUser.username !== decoded.username)
                return res.sendStatus(403);
            const roles = Object.values(foundUser.roles);
            const username = decoded.username;
            const accessToken = jwt.sign(
                {
                    "UserInfo":{
                        "username": decoded.username,
                        "roles": roles
                    }
```

```
                },
                process.env.ACCESS_TOKEN_SECRET,
                {expiresIn: '600s'}
            );
            res.json({roles, username, accessToken});
        }
    );
}
module.exports = {handleRefreshToken};
const User = require('../model/User');
const bcrypt = require('bcrypt');
const handleNewUser = async (req, res) => {
    const {username, password} = req.body;
    if(!username || !password)
    return res.status(400).json({"message": "Username and password
are reqiured."});

    // check for duplicate usernames in the db
    const duplicate = await User.findOne({username:
username}).exec();
    if(duplicate)
    return res.sendStatus(409);
    try {
        //encrypt the password
        const hashPwd = await bcrypt.hash(password, 10);
       //create and store the new user
        const result = await User.create({
            "username": username,
            "password": hashPwd
        });
        console.log(result);
        res.status(201).json({'success':`New user ${username}
created`});
    } catch (err) {
        res.status(500).json({"message": err.message});
    }
}
module.exports = {handleNewUser};
const Room = require("../model/Room");
const User = require("../model/User");
const Controller = require("../model/Controller");
const addNewRoom = async (req, res) => {
  const { roomName } = req.body;
  if (!roomName)
    return res.status(400).json({ message: "roomName are reqiured."
});
  // check for duplicate usernames in the db
  console.log(`username: ${req.username}`);
  const username = req.username;
  const foundUser = await User.findOne({ username: username
}).exec();
  if (!foundUser) return res.sendStatus(401); //Unauthorized
  try {
    //create and store the new user
```

```
    const result = await Room.create({
      userID: foundUser._id,
      roomName: roomName,
    });
    console.log(result);

    res.status(201).json({ success: `New room ${roomName} created`
});
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
};
const getAllUserRooms = async (req, res) => {
  const username = req.username;
  const foundUser = await User.findOne({ username: username
}).exec();
  if (!foundUser) return res.sendStatus(401); //Unauthorized
  const rooms = await Room.find({ userID: foundUser._id });
  if (!rooms) return res.status(204).json({ message: "No rooms" });
  res.json(rooms);
};
const updateRoom = async (req, res) => {
  if (!req?.body?.id)
    return res.status(400).json({ message: "ID room is requred." });
  const roomID = req.body.id;
  const username = req.username;
  const foundUser = await User.findOne({ username: username
}).exec();
  if (!foundUser) return res.sendStatus(401); //Unauthorized
  const room = await Room.findOne({ _id: roomID });
  if (!room)
    return res.status(204).json({ message: `No any rooms witn ID:
${roomID}` });

  if (room.userID != foundUser._id)
    return res.status(403).json({ message: `No permition` });
  if (req?.body?.roomName) room.roomName = req.body.roomName;
  const result = await room.save();
  res.status(201).json(result);
};
const deleteRoom = async (req, res) => {
  if (!req?.body?.id)
    return res.status(400).json({ message: "ID room is requred." });
  const roomID = req.body.id;
  const username = req.username;
  const foundUser = await User.findOne({ username: username
}).exec();
  if (!foundUser) return res.sendStatus(401); //Unauthorized
  const room = await Room.findOne({ _id: roomID });
  if (!room) {
    return res.status(204).json({ message: `No any rooms witn ID:
${roomID}` });
  }
  if (room.userID != foundUser._id)
```

```
    return res.status(403).json({ message: `No permition` });
  const controller = await Controller.findOne({ roomID: roomID });
  if (controller) {
    controller.deleteOne({ _id: controller._id });
  }
  const result = await room.deleteOne({ _id: roomID });
  res.json(result);
};
module.exports = { addNewRoom, getAllUserRooms, updateRoom,
deleteRoom };
const Room = require("../model/Room");
const User = require("../model/User");
const Controller = require("../model/Controller");
// const data = {
//     sensors: require('../model/sensors.json'),
//     setSensors: function (data) {this.sensors = data}
// };
// const getAllSensors  = (req, res) => {
//     res.json(data.sensors);
// }
const getAllUserControllers = async (req, res) => {
    const username = req.username;
    const foundUser = await User.findOne({ username: username
}).exec();
    if (!foundUser) return res.sendStatus(401); //Unauthorized

    const controllers = await Controller.find({ userID: foundUser._id
});
    if (!controllers) return res.status(204).json({ message: "No
controllers" });
    res.json(controllers);
};
const getRoomControllers = async (req, res) => {
    const roomId = req.params.id;
    if (!roomId )
      return res.status(400).json({ message: "Some data are
reqiured." });
       const username = req.username;
    const foundUser = await User.findOne({ username: username
}).exec();
    if (!foundUser) return res.sendStatus(401); //Unauthorized

    const controllers = await Controller.find({ roomID: roomId });
    if (!controllers) return res.status(204).json({ message: "No
controllers" });
    res.json(controllers);
};
const addNewSensor = async (req, res) => {
    const { roomID, name, secure } = req.body;
    if (!roomID || !name || !secure)
      return res.status(400).json({ message: "Some data are
reqiured." });
    // check for duplicate usernames in the db
    console.log(`username: ${req.username}`);
```

```
    const username = req.username;
    const foundUser = await User.findOne({ username: username
}).exec();
    if (!foundUser) return res.sendStatus(401); //Unauthorized
    try {
      //create and store the new user
      const result = await Controller.create({
        userID: foundUser._id,
        roomID: roomID,
        name: name,
        secure: secure,
      });
      console.log(result);

      res.status(201).json({ success: `New controller ${name}
created` });
    } catch (err) {
      res.status(500).json({ message: err.message });
    }
};
// const addNewSensor = (req, res) => {
//     const newSensor = {
//         id: data.sensors[data.sensors.length - 1].id + 1 || 1,
//         name: req.body.name,
//         data: req.body.data,
//         location: {
//             place: req.body.place,
//             description: req.body.description
//         },
//         type: req.body.type,
//         range: req.body.range,
//         connecting: req.body.connecting
//     }
//     if(!newSensor.name || !newSensor.data){
//         return res.status(400).json({"message": "Inputs data are
required."});
//     }
//     data.setSensors([...data.sensors, newSensor]);
//     res.status(201).json(data.sensors);
// }
// const updateSensor = (req, res) => {
//         const sensor = data.sensors.find(sen => sen.id ===
parseInt(req.body.id));
//     if(!sensor){
//             return res.status(400).json({"message": `Sensors ID
${req.body.id} not found`});
//     }
//     if(req.body.name)sensor.name = req.body.name;
//     if(req.body.data)sensor.data = req.body.data;
//      const fileredArray = data.sensors.filter(sen => sen.id !==
parseInt(req.body.id));
//     const unsortedArray = [...fileredArray, sensor];
//     data.setSensors(unsortedArray.sort((a,b) => a.id > b.id ? 1 :
a.id < b.id ? -1 : 0));
```

```
//     res.json(data.sensors);
// }
const updateSensor = async (req, res) => {
    if (!req?.body?.id)
      return res.status(400).json({ message: "Controller ID is
requred." });

    const controllerID = req.body.id;
    const username = req.username;

    const foundUser = await User.findOne({ username: username
}).exec();
    if (!foundUser) return res.sendStatus(401); //Unauthorized

    const controller = await Controller.findOne({ _id: controllerID
});
    if (!controller)
      return res.status(204).json({ message: `No any controller witn
ID: ${controllerID}` });

    if (controller.userID != foundUser._id)
      return res.status(403).json({ message: `No permition` });

    if (req?.body?.controllerName) controller.name =
req.body.controllerName;
    if (req?.body?.minTemp) controller.state.minTemp =
req.body.minTemp;
    if (req?.body?.maxTemp) controller.state.maxTemp =
req.body.maxTemp;
    if (req?.body?.isOn != null) controller.state.isOn =
req.body.isOn;
    if (req?.body?.settingState) controller.state.setings =
req.body.settingState;
    const result = await controller.save();
    res.status(201).json(result);
  };
// const deleteSensor = (req, res) => {
//         const sensor = data.sensors.find(sen => sen.id ===
parseInt(req.body.id));
//     if(!sensor){
//             return res.status(400).json({"message": `Sensors ID
${req.body.id} not found`});
//     }
//      const fileredArray = data.sensors.filter(sen => sen.id !==
parseInt(req.body.id));
//     data.setSensors([...fileredArray]);
//     res.json(data.sensors);
// }
const deleteSensor = async (req, res) => {
    if (!req?.body?.id)
      return res.status(400).json({ message: "Controller ID is
requred." });

    const controllerID = req.body.id;
```

```
    const username = req.username;

    const foundUser = await User.findOne({ username: username
}).exec();
    if (!foundUser) return res.sendStatus(401); //Unauthorized

    const controller = await Controller.findOne({ _id: controllerID
});
    if (!controller){
      return res.status(204).json({ message: `No any controller witn
ID: ${controllerID}` });
    }

    if (controller.userID != foundUser._id)
      return res.status(403).json({ message: `No permition` });

    const result = await controller.deleteOne({ _id: controllerID });
    res.json(result);
  };
// const getSensor = (req, res) => {
//          const sensor = data.sensors.find(sen => sen.id ===
parseInt(req.params.id));
//     if(!sensor){
//             return res.status(400).json({"message": `Sensors ID
${req.body.id} not found`});
//     }
//     res.json(sensor);
// }
module.exports = {
    getAllUserControllers,
    addNewSensor,
    updateSensor,
    deleteSensor,
    getRoomControllers
}
const Room = require("../model/Room");
const User = require("../model/User");
const Controller = require("../model/Controller");
const updateController = async (req, res) => {
  if (!req?.body?.secure)
    return res.status(400).json({ message: "Controller secure is
requred." });
  const controllerSecure = req.body.secure;
  const controller = await Controller.findOne({
    secure: controllerSecure,
  }).exec();
  if (!controller)
    return res.status(204).json({ message: `No any registered
controller` });
  if (req?.body?.settingState == 1) {
    if (req?.body?.minTemp) controller.state.minTemp =
req.body.minTemp;
    if (req?.body?.maxTemp) controller.state.maxTemp =
req.body.maxTemp;
```

```
    if (req?.body?.isOn != null) controller.state.isOn =
req.body.isOn == 1 ? true : false;
  }

  if (req?.body?.isWorking != null) controller.state.isWorking =
req.body.isWorking == 1 ? true : false;
  if (req?.body?.insideTemperature)
    controller.sensors.inside.temperature =
req.body.insideTemperature;
  if (req?.body?.insideHumidity)
    controller.sensors.inside.humidity = req.body.insideHumidity;
  if (req?.body?.outsideTemperature)
    controller.sensors.outside.temperature =
req.body.outsideTemperature;
  if (req?.body?.outsideHumidity)
    controller.sensors.outside.humidity = req.body.outsideHumidity;
    controller.lastConnection =  new Date();
  if (req?.body?.settingState == 0) {
    if (controller.state.setings == true) {
      controller.state.setings = false;
      const result = await controller.save();
      res.status(201).json({
        "isOn": controller.state.isOn == true ? 1 : 0,
        "minTemp": controller.state.minTemp,
        "maxTemp": controller.state.maxTemp,
      });

    }else{
        const result = await controller.save();
        res.status(201).json('');
    }
  } else {
    const result = await controller.save();
    res.status(201).json('');
  }
};
module.exports = {
  updateController,
};
require('dotenv').config();
const express = require('express');
const app = express();
const path = require('path');
const cors = require('cors');
const corsOptions = require('./config/corsOptions');
const {logger} = require('./middleware/logEvents');
const errorHandler = require('./middleware/errorHandler');
const verifyJWT = require('./middleware/verifyJWT');
const cookieParser = require('cookie-parser');
const credentials = require('./middleware/credentials');
const mongoose = require('mongoose');
const connectDB = require('./config/dbConnection');
const PORT = process.env.PORT || 5500;
```

```
//connect to MongoDB
connectDB();

//custom maiddleware logger
app.use(logger);

// Handle options credentials check before CORS!
// and fetch cookies credentials requirement
app.use(credentials);

//Cross Origin Resource Sharing
app.use(cors(corsOptions));

// built-in middleware to handle urlencoded from data
app.use(express.urlencoded({extended: false}));

//built-in middleware for json
app.use(express.json());

//middleware for cookies
app.use(cookieParser());

app.get('/', (req, res) => {
    res.send("Server is running now");
});
//serve static file
app.use('/', express.static(path.join(__dirname, '/public')));
//routes
app.use('/register',require('./routes/register'));
app.use('/login',require('./routes/auth'));
app.use('/refresh',require('./routes/refresh'));
app.use('/logout',require('./routes/logout'));
app.use('/updating',require('./routes/updating'));
app.use(verifyJWT);
app.use('/sensors',require('./routes/api/sensors'));
app.use('/rooms',require('./routes/api/rooms'));
app.all('*',(req, res) => {
    res.status(404);
    if(req.accepts('html')){
        res.send("anything in there");
    }else if(req.accepts('json')){
        res.json({error: "404 Not found"});
    } else {
        res.type('txt').send("404 Not found");
    }
});
app.use(errorHandler);
mongoose.connection.once('open', () => {
    console.log('Connected to MongoDB');
    app.listen(PORT, () => console.log(`Server running on port
${PORT}`));
});
```