МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ

Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ «РОЗУМНОГО ДОМУ»**

Виконав: студент 2 курсу, групи 8.1218-з
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)

освітньої програми інженерія програмного забезпечення

В.В. Дніпровський
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, доцент, к.т.н. Мухін В.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент зав. кафедри фундаментальної математики, доцент, д.т.н. Гребенюк С.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний
Кафедра програмної інженерії
Рівень вищої освіти магістр
Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)
Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.

" ____ " _____ 2019 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Дніпровському Вадиму Володимировичу
(прізвище, ім'я та по-батькові)

1. Тема роботи Розробка програмного забезпечення для «Розумного дому»

керівник роботи Мухін Віталій Вікторович, к.т.н., доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від 29 травня 2019 р. № 812-С

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Розробка програмного коду для керування мікроконтролером

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) _____
презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.		
2.	Збір вихідних даних.		
3.	Обробка методичних та теоретичних		
	джерел.		
4.	Розробка першого і другого розділу.		
5.	Розробка третього розділу.		
6.	Оформлення і нормоконтроль		
	кваліфікаційної роботи.		
7.	Захист кваліфікаційної роботи		

Студент _____ В.В. Дніпровський
(підпис) (ініціали та прізвище)

Керівник роботи _____ В.В. Мухін
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.В. Кудін
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка програмного забезпечення для «Розумного дому»: 52 с., 27 рис., 17 джерел, 7 додатків.

ARDUINO, ESP, МІКРОКОНТРОЛЕР, РОЗУМНИЙ ДІМ.

Об’єкт дослідження – розробка програмного забезпечення для «Розумного дому».

Предмет дослідження – написання програмного коду для мікроконтролеру (МК) на базі мікросхеми ESP8266EX.

Мета роботи: розробка програмного забезпечення для керування мікроконтролером в системі «Розумного дому».

Методи дослідження – аналітичний, описовий.

У кваліфікаційній роботі розглянуті існуючі системи керування «Розумними домами», інтерфейси та протоколи взаємодії між окремими елементами «Розумного дому». Розроблено програмне забезпечення для керування мікроконтролером, що інтегрується в систему «Розумний дім». Результати роботи можуть бути використані для спрощення взаємодії людини та електричних приладів.

SUMMARY

Master's Qualification Paper «Development of Software for «Smart Home»: 52 pages, 27 images, 17sources, 7 attachments.

ARDUINO, ESP, MICROCONTROLLER, SMART HOME.

The object of the study is software development for «Smart home».

The aim of the study is software development for microcontroller operation in the «Smart home» system.

The methods of research are analytical, descriptive.

Current «Smart homes» management systems, interfaces and interaction protocols between separate elements of the «Smart home» are considered in the qualification work. The software for microcontroller operation, which is integrated into the «Smart home» system, is developed.

The work results can be used for simplifying the human and electrical appliances cooperation.

ЗМІСТ

ВСТУП

Розумний дім (РД) – це комплекс рішень для автоматизації повсякденних дій які ведуть до покращення якості життя, підвищення комфорту. Комфортне проживання складається з дрібниць, а РД візьме на себе усі ці дрібниці. Вам не знадобиться добиратись вночі по темряві, якщо захочеться попити води, ви не будете думати про те, що у ванній кімнаті необхідно ввімкнути вентиляцію щоб не мокли стіни під час прийому душу. Чи були такі моменти в житті, що ви не могли пригадати вимкнули ви праску, чи вона залишилась включеною? Були і не один раз… За допомогою сучасної автоматизації ви не тільки зможете, віддалено з телефону або планшету, вимкнути необхідну розетку, а й дізнатись скільки використано електричної енергії на одне прасування.

З розвитком систем «штучного інтелекту» поступово стали розвиватись системи «інтелектуального» управління і РД поступово набуватиме статусу «мисляча будівля».

Метою кваліфікаційної роботи є розробка системи управління елементами РД. Це дозволить реєструвати показники датчиків вологості, температури, а також виконувати керування мікрокліматом.

Для реалізації мети будуть розв’язані наступні задачі:

1) виконати аналіз існуючих систем та технологій які існують на сьогоднішній день, за допомогою яких є можливість побудувати системи домашньої автоматизації, для вирішення поставленої мети;

2) сформувати технічні вимоги до системи;

3) провести підбір апаратних засобів;

4) написання програмного коду до мікроконтролеру для реалізації мети роботи.

1 АНАЛІЗ РИНКУ ГОТОВИХ РІШЕНЬ

Сучасні системи «Розумний дім», котрі забезпечують комфортне проживання та безпеку життя, з кожним роком набувають популярності. Це стало можливим завдяки поступовому здешевленню обладнання, що застосовується в розробці систем «Розумного дому», а також підвищенню технічної грамотності людей.

З кожним днем ми все більше перекладаємо рішення побутових проблем на комп'ютерну техніку. Речі, які раніше описувались художніми фантастами, сьогодні стають реальними. Ми з легкістю передаємо підтримку комфорту в будівлі комп'ютерним технологіям. На сьогоднішній день домашня автоматизація або «Розумний дім» стає нормою життя сучасної людини. Під «Розумним домом» приймається система, котра по заздалегідь сформованим правилам приймає рішення для керування інженерними пристроями, відповідно до прийнятої з зовні інформації (датчики температури, газу, вологості, або камер відеоспостереження). Крім того невід'ємною рисою «Розумного дому» є можливість «спілкуватись» не тільки з компонентами системи, а й з інтернет-сервісами.

На даний момент ринок може запропонувати велику кількість готових продуктів для автоматизації житла тому по першу чергу необхідно провести порівняння систем «Розумного дому». Це дасть можливість максимально раціонально визначитись який функціонал ми захочемо мати в нашій системі. Слід звернути увагу на наступні моменти:

а) модульність системи (централізована, децентралізована схема роботи);

б) масштабованість системи (тобто додавання обладнання для розширення функціоналу);

в) спосіб комутації обладнання (дротовий, бездротовий);

г) засоби управління системою (панелі управління, комп’ютер, телефон);

д) інтерфейс зв’язку з користувачем системи (інтернет, GSM, радіоканал);

е) вартість побудови системи та подальше її утримання.

Принципи роботи будь-якої системи «Розумного дому» можуть бути закладені в одному пристрої, а можуть бути розподілені між окремими модулями системи. На рисунку 1.1 зображена схема децентралізованої роботи пристроїв в системі «Розумного дому».

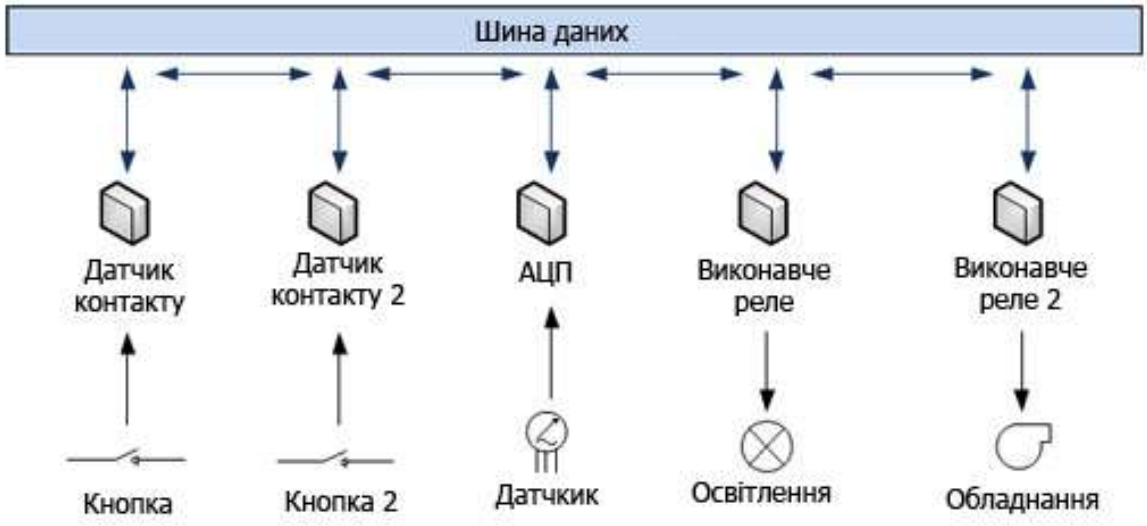

Рисунок 1.1 – Схема децентралізованої системи «Розумного дому» без управляючого контролеру

При такій будові всі елементи системи працюють незалежно один від одного. Вони мають можливість обмінюватись інформацією, посилати команди між окремими елементами, але жоден з них не є головним пристроєм. Зважаючи на ці обставини, а також на те, що обчислювальна спроможність окремих елементів дуже мала можемо зробити висновок, що в децентралізованій схемі неможливо реалізувати складні інтелектуальні алгоритми управління. Децентралізований (розподілений) функціонал між

різними модулями системи дає можливість підвищити відмовостійкість системи. В децентралізованій (модульній) системі вихід з ладу певних елементів призведе лише до часткового зменшення функціоналу «Розумного дому».

Для побудови «Розумного дому» спроможного не тільки керувати інженерними системами, безпекою, освітленням, а й узяти на себе ресурсоємні мультимедійні задачі, відеоспостереження, та інші задачі та функції необхідно використовувати централізовану систему «Розумного дому». Типовий приклад централізованої системи зображено на рисунку 1.2.

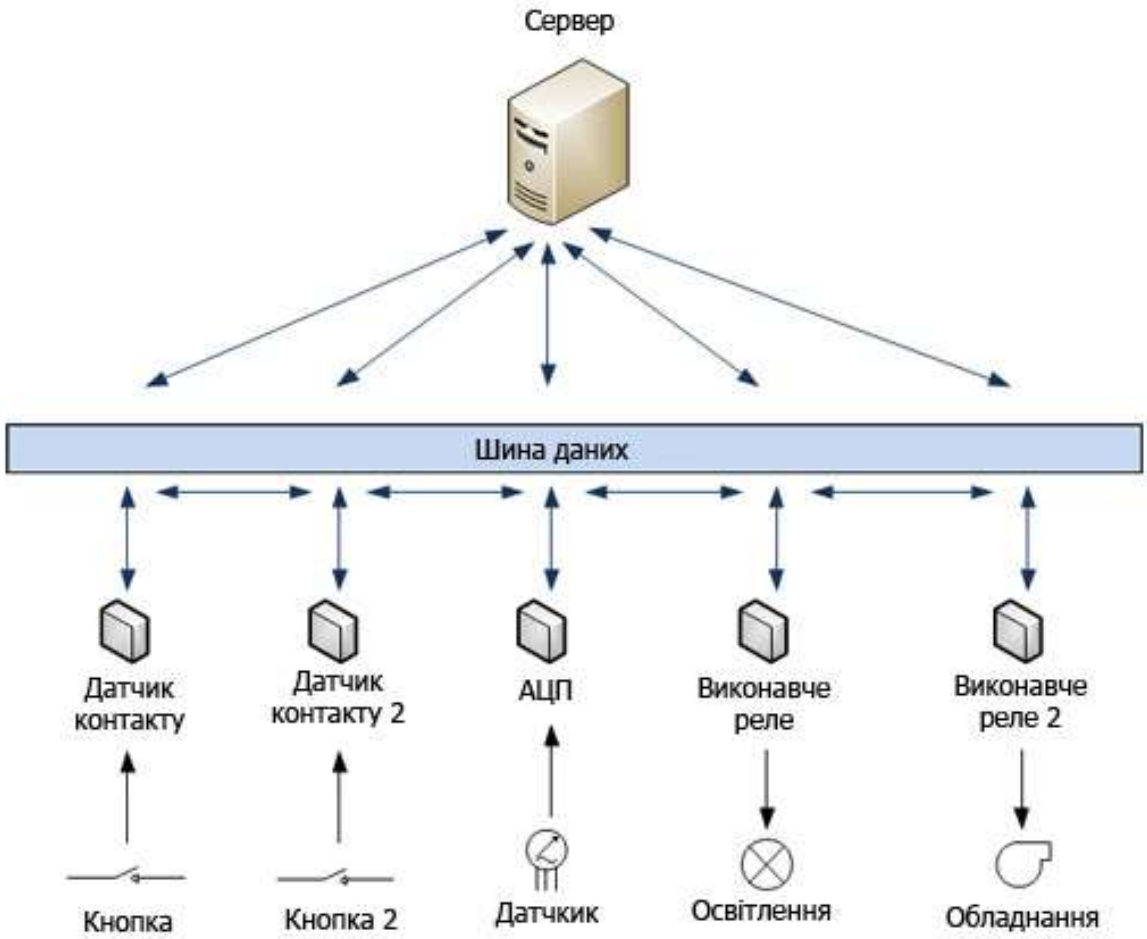

Рисунок 1.2 – Схема централізованої системи «Розумного дому» з управляючим контролером

У даному випадку все принципово інакше, всі елементи системи підкорюються головному управляючому елементу. Всі події в системі потрапляють на головний пристрій і лише він приймає рішення, що робити з цими даними. Такий підхід є оптимальним з точки зору реалізації інтелектуальних алгоритмів, а також надає змогу програмувати систему централізовано. Саме цей спосіб може надати можливість побудувати красивий та функціональний web-інтерфейс. Але в такій схемі реалізації «Розумного дому» є суттєвий недолік. Оскільки весь функціонал зав'язаний на одному пристрої, то його вихід з ладу означатиме лише те, що вся система перестане працювати.

Масштабованість системи також є одним з основних критеріїв на які необхідно звернути увагу ще на етапі проектування системи. У разі вибору системи «все в одному» потрібно ретельно планувати технічне завдання інтегратору системи «Розумного дому», оскільки після проектування та впровадження системи може з'ясуватися, що елементарно забули закласти проводку для вимикача, а підключення бездротових вимикачів не закладено в систему. Тому ще на етапі проектування системи слід закласти можливість масштабування, що полягає у можливості додавання елементів, які в майбутньому дадуть змогу працювати з бездротовими датчиками, надати можливість підключення елементів працюючих за різними протоколами, та на різних шинах даних.

Як показує практика дротове з'єднання є найбільш оптимальним з точки зору стабільності передавання даних та обміну інформації між модулями, та не завжди доцільним з економічної точки зору. Наприклад, для побудови інфраструктури передавання даних за допомогою Ethernet необхідно використовувати комутаційне обладнання яке може коштувати від кількох десятків до сотень доларів. Так само застосовується і спеціальний кабель який коштує близько десяти гривень за один метр. Не слід забувати, що для підключення кінцевого пристрою, будь то вимикач, термометр, або виконавчий механізм, що побудовано на мікроконтролері може виникнути

необхідність придбання Ethernet модуля, який може співпадати, а інколи навіть і перевищувати вартість мікроконтролеру, який працює за бездротовою технологією. Не зважаючи на це стандарт Ethernet за кілька десятиліть зарекомендував себе як дуже надійна середа передавання даних. Всі користувачі мережі Internet працюють за протоколом Ethernet. Маючи технологію з'єднання зірка при пошкодженні проводу перестане працювати лише той елемент, що знаходиться за пошкодженням. Ще однією перевагою Ethernet є робота на швидкості від 10 мБіт/с при дротовому з'єднанні. При використанні бездротових технологій, що працюють за протоколами IEEE 802.11, у залежності від версії протоколу канальна швидкість становитиме від 1 мБіт/с і може досягти кількох гБіт/с. Таким чином при розробці систем «Розумного дому» слід звернути увагу на бездротові технології управління обладнанням. Серед достоїнств технології є простота та гнучкість монтажу, відсутність електричних з'єднань. Мінусом може бути необхідність використання автономного джерела живлення.

Проте існують і більш дешеві рішення використання дротових з'єднань. Так наприклад розроблена корпорацією Dallas Semiconductor двонаправлена шина зв'язку 1-Wire для роботи потребує два дроти. Один з яких є «земля», а другий використовується для живлення та передавання даних. Ця шина не є швидкісною (до 125 кБіт/с, зазвичай 15.4 кБіт/с), тому використовується в більшості випадків для зв'язку з різноманітними датчиками. Пристрої підключаються паралельно до шини, довжина якої може досягати 300 метрів. Недоліком шини є робота по принципу «майстер – слейв». Тобто один ведучий і кілька відомих елементів. Самостійно, без запиту жоден з відомих елементів передавати в мережу не може. Ще одним прикладом використання двох проводів для передавання даних є шина CAN. Зустрічається інколи при побудові «Розумних будівель» але найбільшу популярність здобула в автобудівництві, оскільки відноситься до промислових шин. Промисловими шинами передавання даних також є: CEBus, BatiBus, EIB, LonTalk / LonWorks, LIN, Modbus та інші. Від'ємною

рисою цих протоколів від 1-Wire є те, що вони не працюють за принципом «ведучий – ведений», тобто всі елементи мережі можуть підтримувати зв'язок один з одним, не використовуючи центральний (майстер) вузол. Така реалізація дозволяє значно прискорити роботу системи. Перевагами використання промислових шин є:

– підтримка протоколів багатьма пристроями;
– добре задокументована специфікація;
– завадостійкість шини.

Недоліками використання промислових шин є:

– висока вартість пристрою;
– складність проектування;
– розробкою та впровадженням можуть займатись тільки вузькокваліфіковані спеціалісти.

Будь-якою системою автоматизації необхідно керувати, принаймні на етапі налагодження системи. В залежності від елемента системи можуть бути застосовані мобільний телефон, планшет, комп’ютер або спеціалізована панель управління. На панелі розташовуються кнопки, при натисканні на які виконується конкретна дія. Також на панелях можуть бути розміщені монітори, на які виводиться інформація о приладах, що є частиною системи «Розумного дому». Нажаль такі панелі більше ніде не можна застосувати тому як правило використовуються сучасні мобільні телефони або планшети з спеціальним програмним забезпеченням. Також можливе керування через WEB-браузер. До того моменту, поки інтернет не зайшов майже у кожну оселю для зв’язку з домашніми автоматизованими системами використовували GSM з’єднання та керували за допомогою SMS повідомлень. Сьогодні стали популярні PUSH повідомлення за допомогою онлайн сервісів таких як Telegram Bot. Під час розробки автоматизованих систем де є критичним втрата даних необхідно виконати резервування каналів передавання даних.

На сьогоднішній день існує достатньо велика кількість рішень систем домашньої автоматизації. Їх можна розділити на наступні типи:

– комерційні, з закритим кодом;
– комерційні, з відкритим кодом;
– відкриті системи домашньої автоматизації.

Розглянемо існуючий ринок систем домашньої автоматизації. Комерційні системи з відкритим та закритим кодом умовно віднесемо до одного типу. Оскільки достатньо часто буває, що після виходу на ринок системи з закритим кодом через певний час система ставала відкритою для сторонніх розробників.

Розглянемо найпопулярніші системи:

1) AJAX. Бездротова система українських розробників, яка на початку свого становлення позиціонувалась як охоронних комплекс. Але після розширення функціоналу за допомогою різноманітних бездротових модулів може керувати електричними приборами, працювати з відеокамерами, контролювати проток води та інше. Є комерційним продуктом з центральним модулем, базова ціна буде перевищувати 5000 гривень;

2) DOMOTICZ. Система домашньої автоматизації представляє собою безкоштовну, та відкриту програмну платформу для комплексного управління домашньою автоматикою. Дана система може бути встановлена практично на любий персональний комп’ютер з платформою Windows або Linux. Офіційний сайт розробника – https://www.domoticz.com/;

3) MAJORDOMO. Ще одна система домашньої автоматизації MajorDoMo (Major Domestic Module або Головний Домашній Модуль). Також є безкоштовною та відкритою програмною платформою для комплексного управління домашньою автоматикою, а також для інформаційної підтримки життєдіяльності. Як і Domoticz може бути встановлена як на Windows так і на Linux платформу. Навіть без

прив’язки до обладнання може бути використана для організації персонального інформаційного середовища. Проект представляє собою частину еко-системи SmartLiving. Офіційний сайт розробника - https://mjdm.ru/;

4) HOME ASSISTANT. Також безкоштовна платформа, що працює на Python3. Дозволяє відстежувати та контролювати всі пристрої в домі та автоматизувати дії. Ідеально працює на одноплатному комп’ютері Raspberry PI. Також може працювати на Windows або Linux платформах. Інтерфейс побудовано через браузер, робота можлива на пристроях Android, IOS. Офіційний сайт розробника - https://www.home-assistant.io/;

5) XIAOMI. Представник комерційного проекту. Базовий модуль коштуватиме від 2000 гривень. За допомогою додаткових шлюзів може працювати з різноманітними бездротовими модулями;

6) IOBROKER. WEB візуалізація для платформи ioBroker як додаток до мобільного пристрою. Цей додаток не може працювати самостійно і потребує попередньої установки ioBroker з налаштованими та активованими драйверами vis та web (або socket-io). Розробник: Bluefox. Офіційний сайт розробника - https://www.iobroker.net/.

Також існує велика кількість комерційних проектів, ціна яких сягає кількох тисяч доларів. Обмін інформацією ведеться через промислові протоколи передачі даних. Стороннє обладнання до них підключити неможливо, тому в даній роботі вони не розглядались.

2 ТЕХНІЧНІ ВИМОГИ ДО СИСТЕМИ

Під час проектування елементів системи РД були визначені наступні базові напрямки автоматизації:

– контроль вологості;

– управління освітленням;

– управління системою підігріву підлоги;

– відображення роботи системи через web-інтерфейс;

– можливість інтеграції системи до серверу розумного дому (MAJORDOMO, DOMOTICZ або інших).

Для реалізації кожного напрямку були сформовані наступні послідовності дій системи:

– у разі перевищення максимального значення вологості в приміщенні – включити витяжну вентиляцію;

– під час спрацьовування датчику руху (чи іншого ідентифікатора присутності людини) автоматично, на заданий час, включити освітлення;

– тримати температуру підлоги в заданому діапазоні протягом усього часу роботи системи;

– реалізувати на мікроконтролері web-інтерфейс для налаштування та ручного керування всіма елементами системи;

– реалізувати функцію передавання даних за протоколом MQTT до серверу розумного дому або Zabbix, для накопичення статистичних даних по температурі та вологості у приміщенні.

Відповідно до вищевикладеного можна сформулювати наступні вимоги котрим потрібна задовольняти наша система домашньої автоматизації:

– зчитування показників датчиків вологості та температури;

– управління системою вентиляції;

– управління системою освітлення;

– управління системою підігріву;

– розробити інтуїтивно-зрозумілий web-інтерфейс;

– реалізувати можливість ввімкнення/вимкнення елементів системи через web-інтерфейс;

– забезпечити оновлення даних на web-інтерфейсі без оновлення сторінки за допомогою JSON пакетів та javascript;

– закласти до програмного коду можливість роботи за протоколом MQTT.

Для виконання поставлених задач існують наступні рішення:

– Використання готової системи управління «Розумним домом».

– Розробка власної системи.

Проаналізувавши ринок готових рішень стало зрозуміло, що серед безкоштовних систем потрібного функціоналу нема, за комерційні доведеться заплатити велику кількість грошей.

Розробка системи «Розумного дому» з нуля досить непроста задача, з певною долею вірогідністю це буде рішення, що працює лише з конкретними компонентами системи. Під час побудови системи можна допустити недоліки, які в майбутньому буде неможливо виправити. Крім того розробка «закінченого» власного рішення потребує значних фінансових та часових витрат. Виходячи з вище написаного було прийнято рішення об’єднати обидва підходи. Суть цього підходу полягає у використані готової системи «Розумного дому» для реалізації красивих візуальних ефектів та можливості задіяти складні сценарії роботи системи. З другої сторони буде написано програмний код до мікроконтролеру який дасть змогу реалізувати технічні вимоги, що були пред’явлені до системи. Крім того написання власного коду дає можливість реалізувати гібридну схему управління системою. Тобто при наявності центрального серверу наша система буде підкорюватись командам від нього, як при централізованій системі керування. У разі відсутності або

виходу з ладу центрального серверу мікроконтролер буде самостійно приймати рішення по керуванню кінцевими елементами. Принцип роботи зображено на рисунку 2.1.

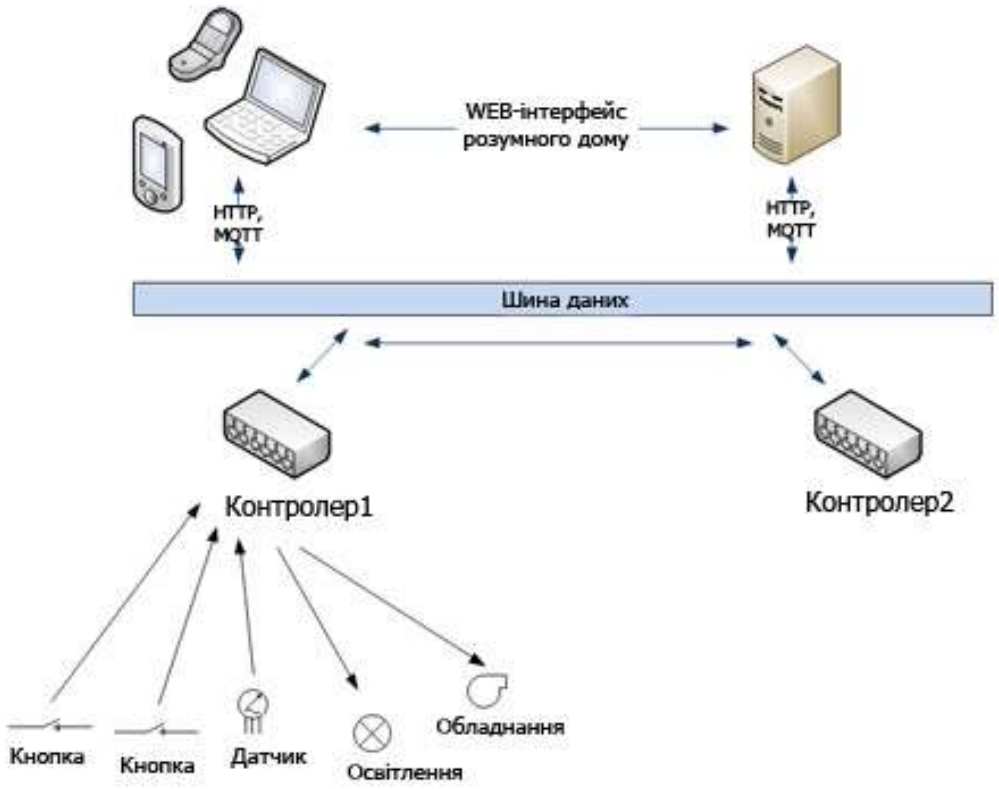

Рисунок 2.1 – Схема роботи серверу «Розумного дому» з мікроконтролером

3 ВИБІР АПАРАТНИХ ЗАСОБІВ

Для побудови серверу «Розумного дому» все частіше стали застосовувати одноплатні комп’ютери які мають певні переваги перед використанням стаціонарних систем. Серед переваг є:

– низька ціна;
– доступність;
– як правило повністю відсутній шум;
– низьке споживання електричної енергії;
– обчислювальної спроможності процесора достатньо для вирішення нескладних задач «Розумного дому»;
– наявність різноманітних інтерфейсів для безпосереднього підключення різноманітних датчиків.

Проаналізувавши переваги одноплатних комп’ютерів свій вибір можна зупинити на таких рішеннях як Raspberry PI, Orange PI, Banana PI, Cubieboard, Asus Tinker та інші. Всі вони наділені інструментарієм Linux, мають можливість підключення до мережі Ethernet дротовим або бездротовим способом. Деякі з них мають можливість працювати по бездротовому протоколу Bluetooth. Та оскільки за межами даної кваліфікаційної роботи планується подальший розвиток функціоналу «Розумного дому» було прийнято рішення використовувати стаціонарний сервер на базі процесору AMD з 16 ГБ оперативної пам’яті, програмним RAID1 та двома SSD дисками, для запуску системи та роботи «віртуальних машин». Для забезпечення безперебійної роботи, у випадку знеструмлення будинку, та для можливості безпечного вимкнення серверу використовується джерело безперебійного живлення на 1,5кВа. На рисунку 3.1 зображено сервер, що буде використовуватись в подальшій роботі з проектом.

Рисунок 3.1 – Сервер «Розумного дому»

Сервер працює під операційною системою Debian9, встановлені наступні основні пакети:

– Samba – для роботи домашнього мережевого сховища;
– proFTP – для передачі файлів за протоколом FTP;
– mosqitto – для обміну та збору даних з кінцевих пристроїв за протоколом MQTT;
– NGINX, Apache – для роботи за протоколом HTTP;
– PHP – для розширення можливостей WEB-сайтів;
– MariaDB – база даних;
– та інші пакети.

Серед різноманітних мікроконтролерів, що доступні на ринку України, було прийнято рішення використовувати мікроконтролер ESP8266, розробку фірми Espressif Systems с бездротовим інтерфейсом Wi-Fi. Технічні характеристики якого складають:

– 80MHz 32-bit процесор з можливістю двократного розгону до 160MHz;
– бездротовий інтерфейс IEEE 802.11 b/g/n з WEP, WPA/WPA2 шифруванням каналу;
– 14 портів вводу-виводу (з яких доступно для використання 11);
– Напруга живлення складатиме від 2,2 до 3,6 В.

Порти вводу-виводу дозволять запрограмувати їх на роботу як SPI, I2S, UART, I2C інтерфейси. До недоліків можна віднести те що мікроконтролер має лише один 10-bit аналогово-цифровий перетворювач. Під час роботи споживання становитиме близько 215мА, 70мА в режимі очікування. У разі використання мікроконтролеру з автономним джерелом живлення можна отримати три режиму зберігання енергії: без зберігання з'єднання з точкою доступу 15мА, Light Sleep 0,4мА, Deep Sleep 15мкА. Зовнішній вигляд мікроконтролеру зображено на рисунку 3.2.

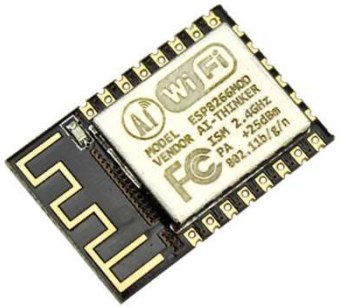

Рисунок 3.2 – Мікроконтролер ESP8266 фірми Espressif Systems

Недоліком такого рішення є неможливість роботи мікроконтролеру без зовнішніх елементів. Так для запуску мікроконтролеру при старті необхідна підтяжка виводу GPIO 15 до землі, а вивід GPIO 0 до шини живлення через резистори на 10 кОм. Мінімальна схема роботи зображена на рисунку 3.3.

Ще однин момент на який необхідно звернути увагу, це необхідність програмувати мікроконтролер. Для цього необхідно до подачі живлення замкнути на землю GPIO 0, після чого подати живлення на мікроконтролер. Далі використовуючи виводи GPIO 1 та GPIO 3 (UART інтерфейс) провести програмування мікроконтролеру однією з програмам: XTCOM, NODEMCU, ESPTOOL. Тож для вирішення усіх можливих ускладнень з якими можемо зітнутися, а саме:

– організація живлення мікроконтролеру на 3,3в;

– програмування мікроконтролеру через USB інтерфейс;

– виконання резистивної обв’язки,

будемо використовувати модуль Wemos D1 mini, який представлено на рисунку 3.4. Він має всю необхідну резистивну та ємнісну обв’язку, перетворювач живлення з 5 на 3,3 В, USB/UART перетворювач інтерфейсів.

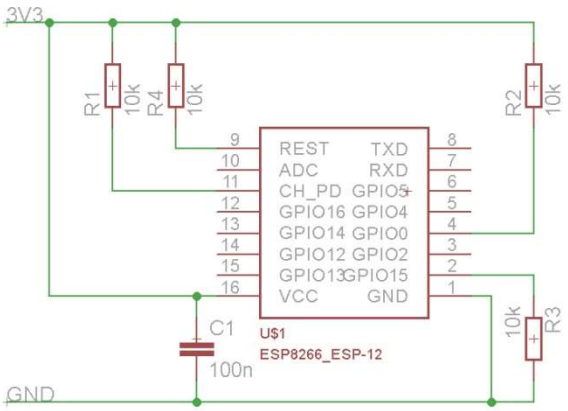

Рисунок 3.3 – Типова схема підключення мікроконтролеру ESP8266

Рисунок 3.4 – Wemos D1 mini з мікроконтролером ESP8266

Для керування електричними приладами будемо використовувати реле Omron g3mb-202p розрахований на струм комутації 2А. Не зважаючи на роботу від 5 вольт, реле добре працює від 2,7 вольт вхідної напруги. Що дає можливість прямого підключення реле до мікроконтролеру.

Рисунок 3.5 – Твердотільне реле OMRON G3MB-202P

Для вимірювання вологості в приміщенні будемо використовувати датчик SHT21 який працює по шині I2C. Датчик має можливість вимірювати вологість в діапазоні від 0 до 100% з похибкою вимірювання ±2 %, також закладена можливість вимірювати температуру в діапазоні від -40 °C до +125 °C з похибкою в ±0,3 °C. Напруга живлення становить від 2,1 до 3,6 В.

Рисунок 3.6 – Датчик вологості та температури SHT21

Для вимірювання температури теплої підлоги буде задіяний датчик DS18B20 від Dallas Semiconductor, який працює по шині 1-Wire. Датчик вимірює температуру в діапазоні від -55°C до +125°C. Точність вимірювання в діапазоні від -10°C до +85°C складатиме ±0,5°C.

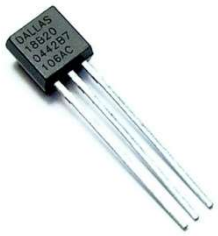

Рисунок 3.7 – Датчик температури DS18B20

Датчик детектування руху у приміщені зображено на рисунку 3.8.

Рисунок 3.8 – Піроелектричний датчик руху HC-SR501

Для обміну даними між сервером «Розумного дому» та мікроконтролером буде використовуватись обладнання фірми MikroTIK, а саме роутер RB800 з двома радіо модулями R52Hn. Один з яких працює в діапазоні 2,4ГГц для роботи з мікроконтролерами, другий в діапазоні 5ГГц для підключення абонентських терміналів. Роутер має три Ethernet інтерфейси. Це дозволило один інтерфейс задіяти для підключення

оптичного волокна, що йде від провайдера. Другий інтерфейс задіяно для підключення комп'ютерної периферії та камер відео спостереження, у якості комутатору використовується ProCurve Switch 2810-48G від компанії Hewlett-Packard.

Рисунок 3.9 – Роутер MikroTIK RB800

Рисунок 3.10 – Комутатор ProCurve Switch 2810-48G

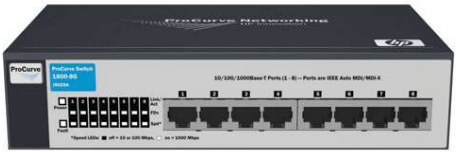

Рисунок 3.11 – Комутатор ProCurve Switch 1800-8G

Третій порт роутеру використовується для підключення серверу «Розумного дому» та інших пристроїв, доступ до яких обмежує Firewall роутеру, у якості комутатору використовується ProCurve Switch 1800-8G. Окрім функції Firewall, роутер виконує функції L2TP та PPPoE серверу для можливості віддаленого підключення.

4 ПРОГРАМНО-АПАРАТНА РЕАЛІЗАЦІЯ ПРОЕКТУ

Реалізація програмної частини потребує попередньої розробки електричної схеми з’єднань виводів мікроконтролеру та виводів сенсорів, реле та кнопок. На рисунку 4.1 представлено електричну схему.

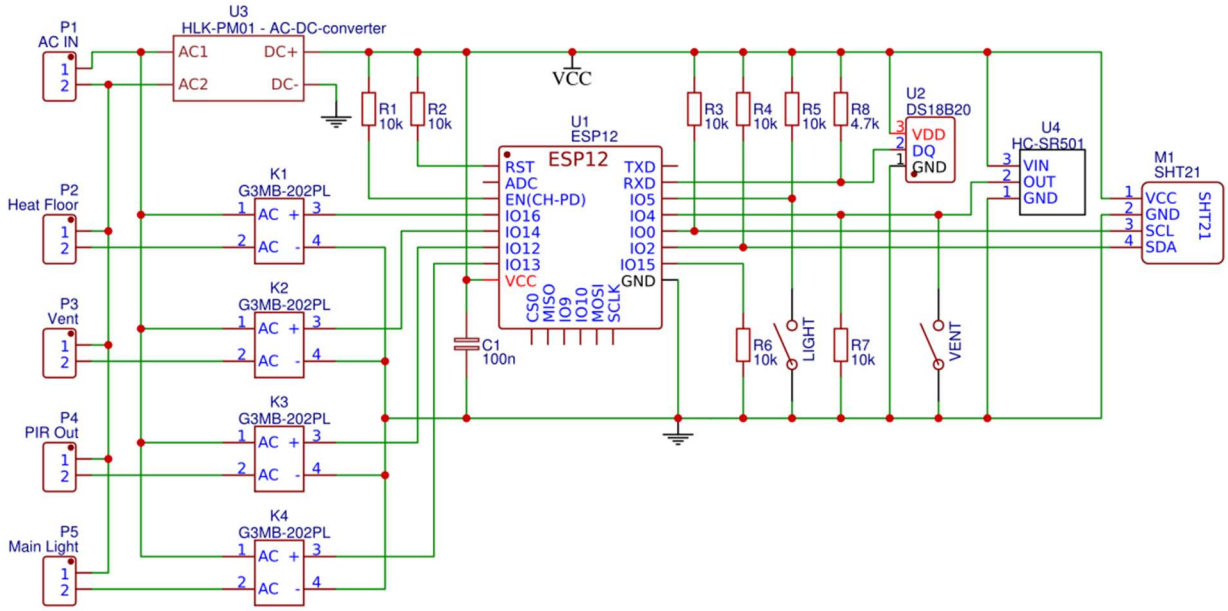

Рисунок 4.1 – Схема електрична

Необхідні 3,3 В для роботи мікроконтролеру та сенсорів використовується AC/DC перетворювач напруги HLK-PM01. Для ручного керування світлом та включення витяжної вентиляції необхідно використовувати кнопки без фіксації положення.

Для написання програмного коду та програмування мікроконтролеру будемо використовувати Arduino IDE. Для чого необхідно виконати підготовчі роботи ArduinoIDE. Відкрити налаштування і в полі Додаткові модулі для менеджера плат додати наступну строку:

http://arduino.esp8266.com/stable/package_esp8266com_index.json

Рисунок 4.2 – Фрагмент інтерфейсу Arduino IDE

Після чого відкривши менеджер плат виконати установку для плат **esp8266** by **ESP8266 Community.**

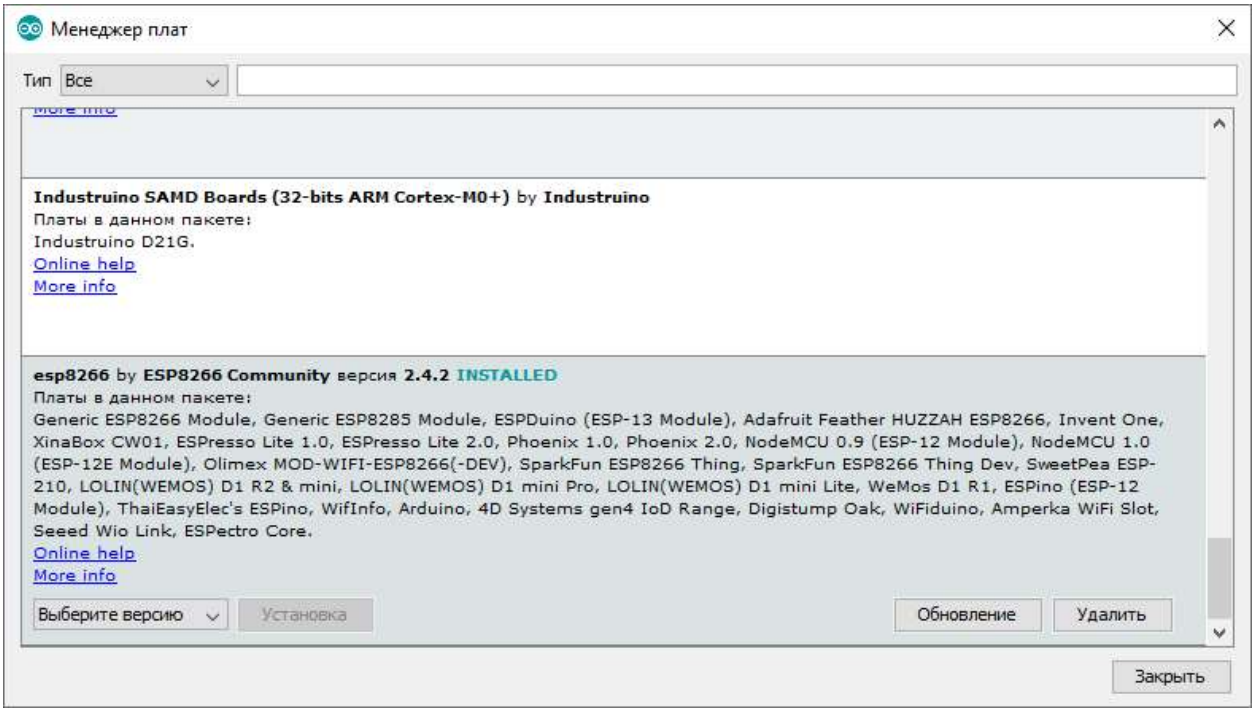

Рисунок 4.3 – Менеджер плат

Якщо все зроблено правильно то в менеджері плат будуть доступні для вибору різноманітні плати на базі мікроконтролеру ESP.

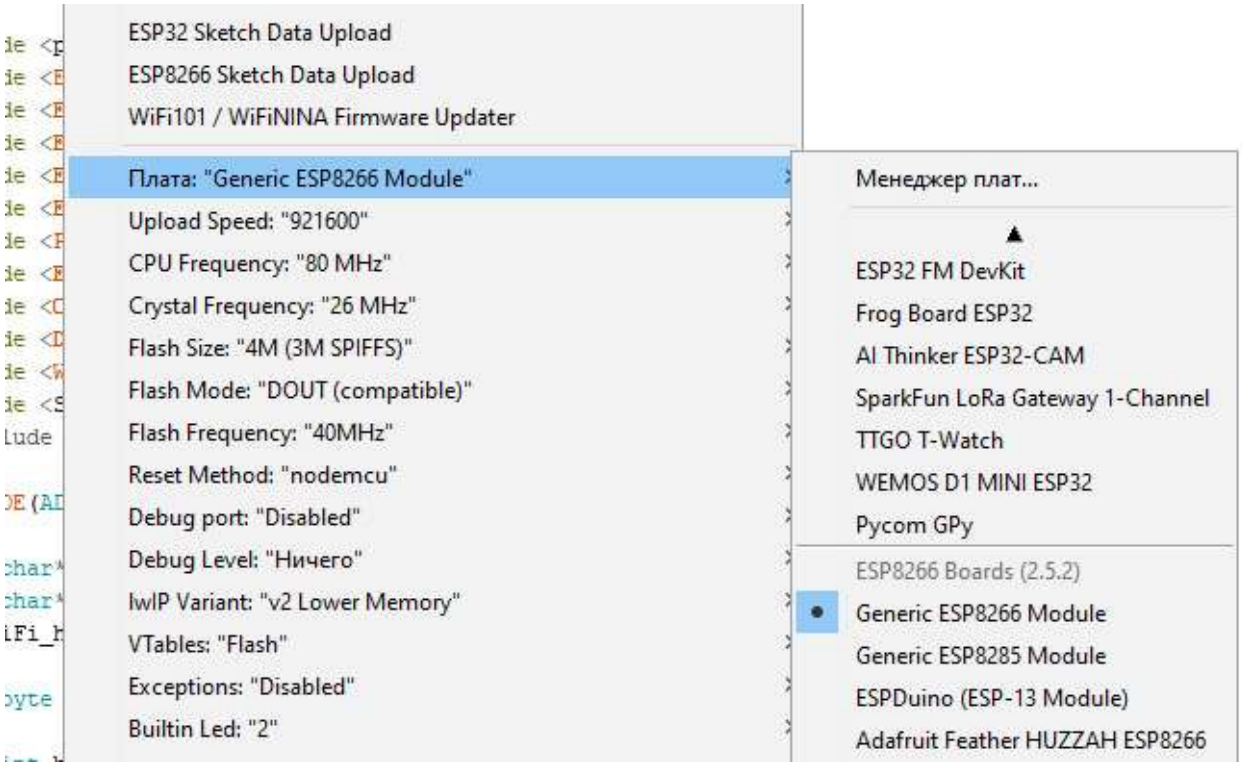

Рисунок 4.4 – Доступність плат в середовищі ArduinoIDE

Для роботи з датчиком вологості та температури SHT21 будемо використовувати бібліотеку **Sodaq_SHT2x** Built-In by **keestux,SODAQ**. Для роботи з датчиком температури ds18b20 будемо використовувати бібліотеку **DS18B20Events** by **Ihar Yakimush**. Для роботи з брокером MQTT будемо використовувати бібліотеку **PubSubClient** by **Nick O`Leary**. Бібліотеки необхідно додати через менеджер бібліотек, або безпосередньо скачати з GitHub. Інші, використані бібліотеки під час написання коду входять у стандартний пакет.

Для економії пам’яті SRAM будемо використовувати модифікатор змінних PROGMEM, це дозволить записувати дані у flash-пам'ять (тобто програмна пам'ять). Він вказує компілятору про необхідність розміщувати інформацію у flash-пам'ять, а не SRAM як це відбувається при звичайній компіляції коду. PROGMEM є частиною бібліотеки pgmspace.h, котра доступна лише для архітектури ARM. Для її підключення необхідно в самому початку коду додати наступний запис:

```
#include <pgmspace.h>
```

після чого додавати інші необхідні бібліотеки.

https://www.arduino.cc/reference/en/language/variables/utilities/progmem/

Під час написання коду був розроблений веб-інтерфейс, який дозволяє відстежувати роботу мікроконтролеру, а також при необхідності змінювати налаштунки. Головна сторінка зображена на рисунку 4.5.

На головну сторінку винесені наступні данні:

– VCC – поточні значення напруги на мікроконтролері;

– WiFi mode – дає можливість визначити режим роботи мікроконтролеру (режим точки доступу чи режим клієнта);

– MQTT broker – показує, чи відбулося з’єднання мікроконтролеру та брокеру MQTT;

– Heap free size – відображає наявну вільну оперативну пам'ять;

– Uptime – показую скільки минуло часу з моменту включення мікроконтролера;

– ліміт, та поточні показники датчику температури ds18b20;

– ліміт, та поточні показники датчику вологості та температури SHT21;

– елементи ручного керування виходами мікроконтролеру;

– кнопки налаштування мікроконтролеру.

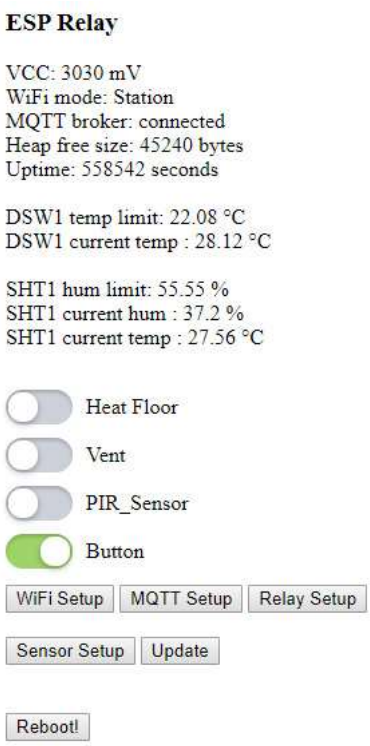

Рисунок 4.5 – Головна сторінка web-інтерфейсу

Якщо перейти на сторінку WiFi Setup то нам будуть доступні наступні налаштування мікроконтролеру:

– SSID – ім’я точки доступу, до якої буде підключатись мікроконтролер;

– Password – ключ шифрування точки доступу;

– mDNS domain – доменне ім’я мікроконтролеру в локальній мережі.

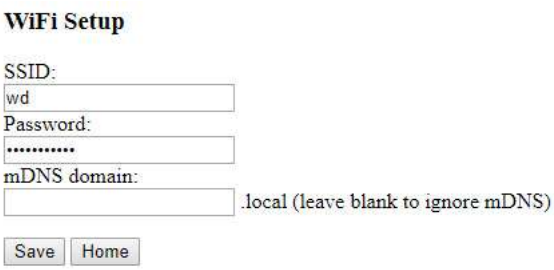

Рисунок 4.6 – Сторінка налаштування бездротового з’єднання

В програмному коді реалізовано наступний принцип роботи бездротового модулю: при першому підключенні, або неможливості підключитися до точки доступу (інтервал спроб підключення складає тридцять секунд), що зазначено в налаштуваннях WiFi Setup пристрій переходе в режим точки доступу і доступний за адресою http://192.168.4.1/ з ім’ям точки доступу espAP.

Через п’ять хвилин роботи в режимі точки доступу буде виконана повторна спроба підключитися до точки доступу зазначеною в налаштуваннях бездротового модулю. Ця функція реалізована наступним кодом:

```
if (millis() - WiFiModeTime > 300000) {
   if (WiFi.status() != WL_CONNECTED) {
     Serial.print("Reconnect to Wi-Fi: ");
     setupWiFi();
   }
   WiFiModeTime = millis();
 }
```

Рисунок 4.7 - Фрагмент коду повторного підключення до базової станції

Необхідність такого функціоналу полягає в тому, що у разі повного знеструмлення будівлі та відключення джерела безперебійного живлення після подачі живлення першим завантажиться мікроконтролер, а вже потім точка доступу, до якої він підключався. У зв’язку з цим контролер перейде в режим точки доступу і в такому режимі залишався би до перезавантаження.

Однією з можливостей пошуку та ідентифікації пристрою в локальній мережі є використання протоколу Multicast DNS або скорочено mDNS. Цей протокол дає можливість обрати пристрою ім’я в зоні .local. Працює він майже як звичайний DNS. Кожен пристрій зберігає записи своєї зони сам і також самостійно обробляє запити до них. Коли сторонній пристрій хоче

визначити запис зони (узнати IP-адресу по імені) він звертається до широкомовної-адреси 224.0.0.251, відповідно цей запит отримують всі пристрої в локальній мережі, а відповість лише той, хто зберігає зону для нашого імені. Принцип роботи протоколу описаний в RFC 3927.

http://tools.ietf.org/html/rfc3927

Наступним елементом налаштувань є вкладка MQTT Setup.

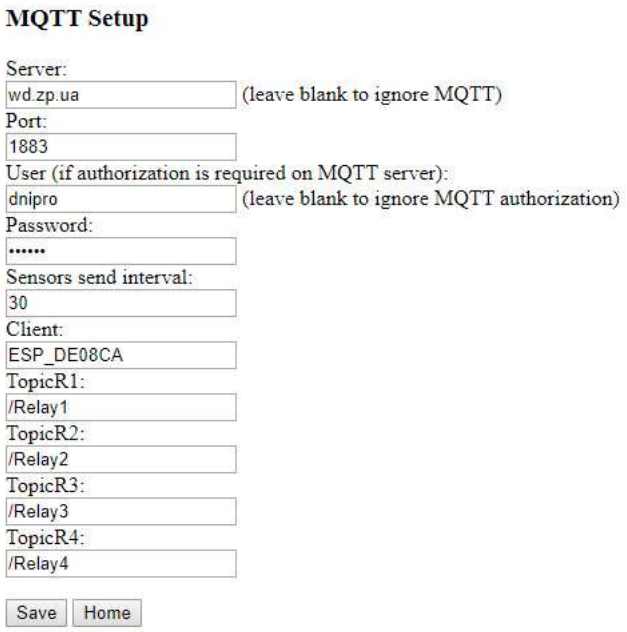

Рисунок 4.8 – Сторінка налаштування MQTT Setup

Ця вкладка дозволяє налаштувати роботу мікроконтролеру з брокером MQTT. Саме за допомогою протоколу MQTT контролер РД буде керувати нашим мікроконтролером та отримувати від нього дані.

Вкладка Relay Setup дозволяє призначити кожному реле номер виходу на мікроконтролері, а для реле 1 та реле 2 задати граничні показники температури та вологості. Крім того в даному меню є можливість вказати логічний рівень виходів мікроконтролеру під час завантаження, а також виконати їх інверсію.

В коді реалізовано наступну логіку роботи виходів:

– для реле 1 – статус виводу зміниться якщо буде досягнуто граничне значення датчику температури ds18b20. У разі ручного включення через веб-інтерфейс, або віддалено через брокер MQTT відбудеться зміна статусу виводу на час зазначений на вкладці Sensor Setup у полі «Floor Heater Work Limit»;

– для реле 2 – статус виводу зміниться якщо буде досягнуто граничне значення датчику вологості SHT21. Якщо включити реле вручну через веб-інтерфейс, або через брокер MQTT відбудеться зміна статусу виводу на постійній основі і подальші значення датчику вологості будуть ігноруватись поки вручну, або через брокер не прийде команда на зміну статусу реле;

– для реле 3 – статус виводу зміниться якщо прийде високий рівень сигналу від датчику руху, або після натискання фізичної кнопки. Зміна статусу пройде на час, що зазначено на вкладці Sensor Setup у полі «PIR-Sensor Work Limit». У разі, якщо час не було вичерпано, а був отриман повторно високий рівень сигналу від датчику руху лічильник почне відлік з початку. Якщо включити реле вручну через веб-інтерфейс, або через брокер MQTT відбудеться зміна статусу виводу на постійній основі принцип роботи буде аналогічний до реле 2;

– для реле 4 – не залежно від того, де було виконано натискання (веб-інтерфейс, брокер MQTT, фізична кнопка) кнопки статус виводу буде змінюватись на протилежний.

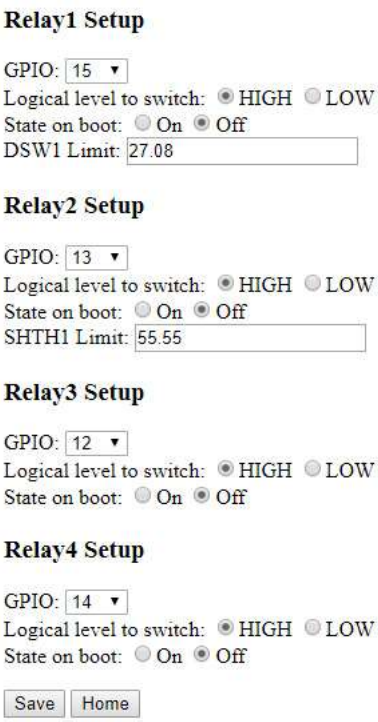

Рисунок 4.9 – Сторінка налаштування Relay Setup

Вкладка Sensor Setup призначена для налаштування I2C, 1-Wire інтерфейсів, а також вказується час опиту датчиків вологості та температури, час роботи реле теплої підлоги та датчику руху.

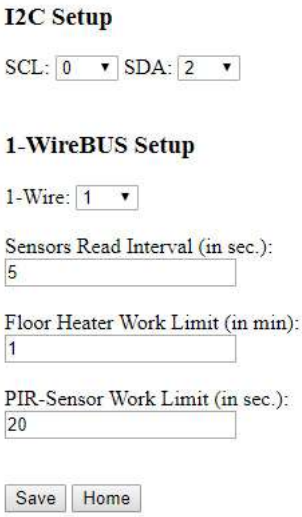

Рисунок 4.10 – Сторінка налаштування Sensor Setup

Оскільки робота мікроконтролеру не передбачає постійного доступу до UART-інтерфейсу мікроконтролеру, то була закладена можливість виконувати оновлення прошивки через веб-інтерфей. Для цього необхідно в середовищі ArduinoIDE виконати експорт бінарного файлу. Після цього відкрити вкладку Update і вказати місце зберігання нашого файлу прошивки. Він буде розташований у папці зі скетчем. Після чого натиснути кнопку «Update»

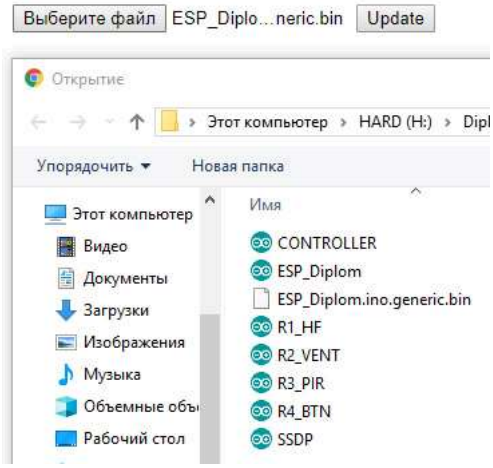

Рисунок 4.11 – Сторінка Update

На головній сторінці імітацію кнопок виконують чекбокси. Зазвичай для їх оформлення, як того вимагає дизайн сайту, раніше використовували JavaScript та після розширення можливостей CSS до JavaScript необхідно звертатись для оптимізації роботи сайту лише зі старими браузерами (наприклад Internet Explorer 8).

Для оформлення крім самого тегу елементу <input type="checkbox"> ще нам знадобиться тег <label>. Завдяки цьому тегу буде можливість перемикати стан реле не тільки натискаючи на елемент, а й на текст. Обов’язковою умовою є використання тегу < input > до тегу <label>. Суть роботи полягає у використанні псевдо селекторів :checked та :not. При цьому сам чекбокс стає невидимим, а його емуляція виконується за допомогою

псевдо елементів :before та :after для тегу <label>. За допомогою властивостей position, z-index та opacity для класу .checkbox ми візуально прячемо оригінальні елементи, при цьому вони залишаються на тому ж самому місці, де будуть стилізовані елементи.

Ще одна функція для пошуку мікроконтролеру в локальній мережі реалізована за допомогою протоколу SSDP. Відкривши мережеве середовище можна знайти наш пристрій, він буде знаходитись серед «Інші пристрої». Подвійним кліком лівої кнопки миші відкриється веб-інтерфей мікроконтролеру у браузері за замовченням. Натиснувши правою кнопкою миші по іконці пристрою можна подивитись властивості пристрою. Інформацію, що виводиться можна змінити у скетчі для цього необхідно змінити наступні строчки коду:

```
SSDP.setHTTPPort(80);
SSDP.setName(WiFi_hostname);
SSDP.setSerialNumber(ESP.getChipId());
SSDP.setURL("/");
SSDP.setModelName("ESP-HUM/PIR_Controller");
SSDP.setModelNumber("000000000001");
SSDP.setModelURL("http://wd.zp.ua/");
SSDP.setManufacturer("Dneprovskiy Vadim");
SSDP.setManufacturerURL("http://wd.zp.ua");
```

Рисунок 4.12 - Фрагмент коду SSDP ідентифікації в мережевому середовищі

Повний код надано в додатку Ж.

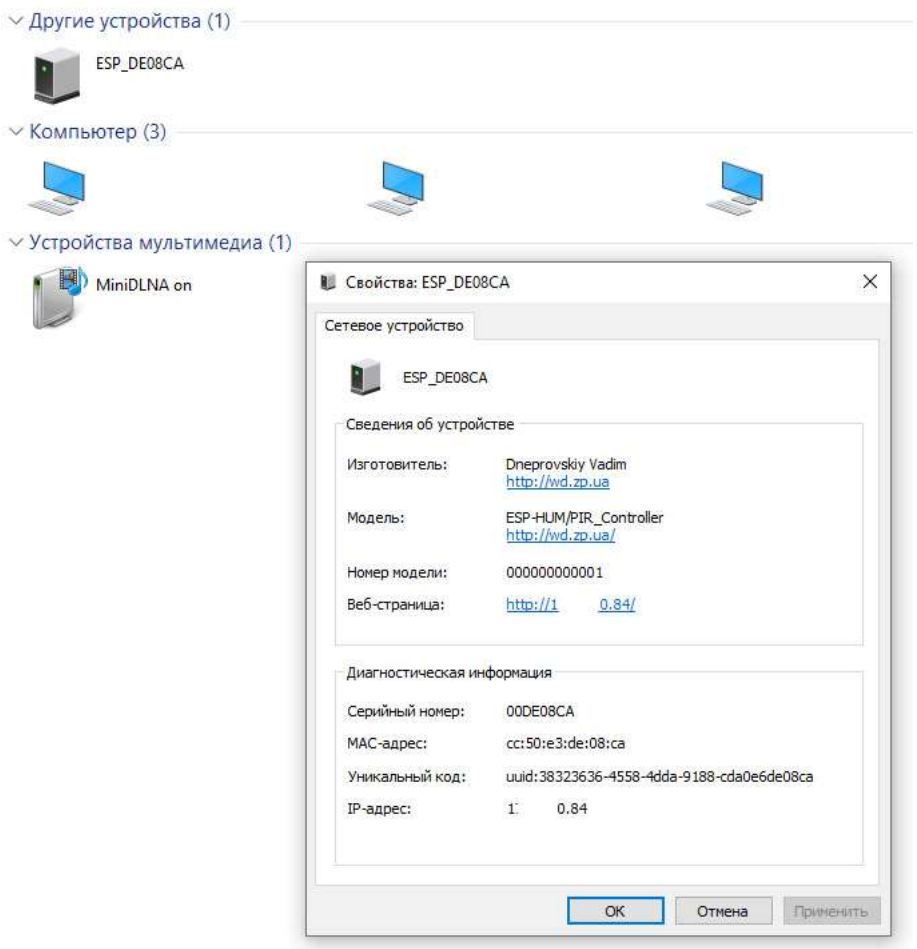

Рисунок 4.13 – Відображення мікроконтролеру в мережевому середовищі

ВИСНОВКИ

Під час написання кваліфікаційної роботи було проведено аналіз існуючих систем та технологій, розроблено програмний код до мікроконтролеру на базі мікросхеми ESP8266. У можливості коду увійшли наступні функції: можливість керування мікрокліматом в приміщені за показниками вологості та температури, робота освітлювальними приладами в ручному та автоматичному режимі. Можливість керування мікроконтролером реалізована через веб-інтерфейс, та за протоколом MQTT з серверу РД. Крім того реалізована можливість накопичення статистичних даних на сервері РД.

СПИСОК ЛІТЕРАТУРИ

1. Методичні вказівки до написання курсових і кваліфікаційних робіт для здобувачів ступеня вищої освіти бакалавра та магістра математичного факультету / [С.І. Гоменюк, С.М. Гребенюк, І.В. Зіновєєв та ін.]. – Запоріжжя: ЗНУ, 2017. – 67 с.

2. Меласк Розумний Дім - приватне дослідно-виробниче підприємство. URL: http://melask.com.ua/

3. Умный дом. URL: https://ru.wikipedia.org/wiki/Умный_дом

4. Олексій Волочай. «Розумний будинок» - Огляди ринків. URL: http://www.jeybud.com.ua/index.php?item=articles&sub=3437&d_id=4

5. Рейтинг систем «Умный дом по производителям». URL: https://vencon.ua/articles/reyting-sistem-umnyy-dom-po-proizvoditelyam

6. DOMOTICZ. URL: https://www.domoticz.com/

7. MAJORDOMO. URL: https://mjdm.ru/

8. HOME ASSISTANT. URL: https://www.home-assistant.io/

9. XIAOMI. URL: https://xiaomi-smarthome.ru/

10. IOBROKER. URL: https://www.iobroker.net/

11. MegaD-328 – Готовый многофункциональный контроллер Умного дома. URL: https://ab-log.ru/smart-house/ethernet/megad-328

12. Digital Humidity Sensor SHT2x (RH/T). URL: https://www.sensirion.com/en/environmental-sensors/humidity-sensors/humidity-temperature-sensor-sht2x-digital-i2c-accurate/

13. Programmable Resolution 1-Wire Digital Thermometer. URL: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

14. Стилизация чекбоксов и радиокнопок на чистом CSS с совместимостью для старых браузеров. URL: http://dimox.name/custom-checkboxes-and-radio-buttons-using-css-only/

15. Arduino library for SHT2x temperature and humidity sensors. URL: https://github.com/SodaqMoja/Sodaq_SHT2x

16. Arduino thermometer with onChange event based on DS18B20 sensor. URL: https://github.com/IharYakimush/arduino-temperature-control-events

17. PROGMEM. URL: https://www.arduino.cc/reference/en/language/variables/utilities/progmem/

ДОДАТОК А

Фрагмент основного коду мікроконтролеру, що зображує перелік підключених бібліотек, змінних, цикл void setup() та void loop():

```
#include <pgmspace.h>
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include <ESP8266SSDP.h>
#include <ESP8266mDNS.h>
#include <ESP8266HTTPUpdateServer.h>
#include <PubSubClient.h>
#include <EEPROM.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <Wire.h>
#include <Sodaq_SHT2x.h>

ADC_MODE(ADC_VCC);

const char* const ssidAP PROGMEM = "espAP";
const char* const passwordAP PROGMEM = "";
char WiFi_hostname[20];

const byte pinBuiltinLed = 255;

const int button1Pin = 4;
const int button2Pin = 5;
int button1State;
int button2State;
int lastButton1State = HIGH;
int lastButton2State = LOW;
bool BTN2State = LOW;
unsigned long lastDebounce1Time = 0;
unsigned long lastDebounce2Time = 0;
unsigned long debounce1Delay = 50;
unsigned long debounceDelay = 50;
unsigned long countHF = 0;

const char configSign[4] PROGMEM = { '#', 'R', 'E', 'L' };
const byte maxStrParamLength = 32;

const char* const ssidArg PROGMEM = "ssid";
const char* const passwordArg PROGMEM = "password";
const char* const domainArg PROGMEM = "domain";
const char* const serverArg PROGMEM = "server";
const char* const portArg PROGMEM = "port";
const char* const userArg PROGMEM = "user";
const char* const mqttpswdArg PROGMEM = "mqttpswd";
const char* const sendintervalArg PROGMEM = "sendinterval";
const char* const sensreadmaxArg PROGMEM = "sensreadmax";
const char* const floormaxArg PROGMEM = "floormax";
const char* const PIRmaxArg PROGMEM = "PIRmax";
const char* const clientArg PROGMEM = "client";
const char* const topicR1Arg PROGMEM = "topicR1";
const char* const topicR2Arg PROGMEM = "topicR2";
const char* const topicR3Arg PROGMEM = "topicR3";
const char* const topicR4Arg PROGMEM = "topicR4";
```

```
const char* const gpio1Arg PROGMEM = "gpio1";
const char* const gpio2Arg PROGMEM = "gpio2";
const char* const gpio3Arg PROGMEM = "gpio3";
const char* const gpio4Arg PROGMEM = "gpio4";
const char* const gpioSCLArg PROGMEM = "gpioSCL";
const char* const gpioSDAArg PROGMEM = "gpioSDA";
const char* const gpio1WRArg PROGMEM = "gpio1WR";
const char* const level1Arg PROGMEM = "level1";
const char* const level2Arg PROGMEM = "level2";
const char* const level3Arg PROGMEM = "level3";
const char* const level4Arg PROGMEM = "level4";
const char* const onboot1Arg PROGMEM = "onboot1";
const char* const onboot2Arg PROGMEM = "onboot2";
const char* const onboot3Arg PROGMEM = "onboot3";
const char* const onboot4Arg PROGMEM = "onboot4";
const char* const rebootArg PROGMEM = "reboot";
const char* const vccArg PROGMEM = "vcc";
const char* const wifimodeArg PROGMEM = "wifimode";
const char* const mqttconnectedArg PROGMEM = "mqttconnected";
const char* const freeheapArg PROGMEM = "freeheap";
const char* const uptimeArg PROGMEM = "uptime";
const char* const dsw1maxArg PROGMEM = "dsw1max";
const char* const shth1maxArg PROGMEM = "shth1max";
const char* const relay1Arg PROGMEM = "relay1";
const char* const relay2Arg PROGMEM = "relay2";
const char* const relay3Arg PROGMEM = "relay3";
const char* const relay4Arg PROGMEM = "relay4";
const char* const dsw1Arg = "dsw1";
const char* const shtt1Arg = "shtt1";
const char* const shth1Arg = "shth1";
bool relay1, relay2, relay3, relay4;
bool r1sw, r2sw, r3sw, r4sw, hf, r1swState, vent, pirSt, lastrelay1,
lastrelay2, lastrelay3;
unsigned long PirTime;
String ssid, password, domain;
String mqttServer, mqttUser, mqttPassword, mqttSendinterval, mqttClient =
"ESP_Relay";
String mqttTopicR1 = "/Relay1", mqttTopicR2 = "/Relay2", mqttTopicR3 =
"/Relay3", mqttTopicR4 = "/Relay4";
String sensreadmax, floormax, PIRmax;
uint16_t mqttPort = 1883;
float dsw1, shtt1, shth1;
String dsw1max = "22.05";
String shth1max = "55.55";
byte relay1Pin = 15;
byte relay2Pin = 13;
byte relay3Pin = 12;
byte relay4Pin = 14;
byte gpioSCLPin = 2;
byte gpioSDAPin = 0;
byte gpio1WRPin = 1;
bool relay1Level = LOW;
bool relay2Level = LOW;
bool relay3Level = LOW;
bool relay4Level = LOW;
bool relay1OnBoot = false;
bool relay2OnBoot = false;
bool relay3OnBoot = false;
bool relay4OnBoot = false;
unsigned long sensorsTime = 0;
unsigned long mqttsendTime = 0;
unsigned long WiFiModeTime = 0;
```

```
ESP8266WebServer httpServer(80);
ESP8266HTTPUpdateServer httpUpdater;
WiFiClient espClient;
PubSubClient pubsubClient(espClient);
OneWire oneWire(gpio1WRPin);
DallasTemperature sensors(&oneWire);

void setup() {
  Wire.begin(gpioSDAPin, gpioSCLPin);
  //Serial.begin(115200);
  Serial.println();
  sensors.begin();
  Serial.println();
  Serial.print("ESP chip ID: ");
  Serial.println(ESP.getChipId());
  Serial.print("ESP flash chip size/real size: ");
  Serial.print(ESP.getFlashChipSize());
  Serial.print(" / ");
  Serial.println(ESP.getFlashChipRealSize());
  sprintf(WiFi_hostname, "ESP_%06X", ESP.getChipId());
  Serial.print("WiFi_hostname: ");
  Serial.println(WiFi_hostname);
  SSDP_init();

  EEPROM.begin(1024);
  if (! readConfig())
  Serial.println(F("EEPROM is empty!"));

  pinMode(pinBuiltinLed, OUTPUT);
  pinMode(relay1Pin, OUTPUT);
  pinMode(relay2Pin, OUTPUT);
  pinMode(relay3Pin, OUTPUT);
  pinMode(relay4Pin, OUTPUT);
  pinMode(button1Pin, INPUT);
  pinMode(button2Pin, INPUT);

  digitalWrite(relay1Pin, relay1Level == relay1OnBoot);
  digitalWrite(relay2Pin, relay2Level == relay2OnBoot);
  digitalWrite(relay3Pin, relay3Level == relay3OnBoot);
  digitalWrite(relay4Pin, relay4Level == relay4OnBoot);

  setupWiFi();

  httpUpdater.setup(&httpServer);
  httpServer.onNotFound([]() {
    httpServer.send(404, F("text/plain"), F("FileNotFound"));
  });
  httpServer.on("/", handleRoot);
  httpServer.on("/index.html", handleRoot);
  httpServer.on("/wifi", handleWiFiConfig);
  httpServer.on("/mqtt", handleMQTTConfig);
  httpServer.on("/relay", handleRelayConfig);
  httpServer.on("/sensor", handleSensorConfig);
  httpServer.on("/store", handleStoreConfig);
  httpServer.on("/switch1", handleRelay1Switch);
  httpServer.on("/switch2", handleRelay2Switch);
  httpServer.on("/switch3", handleRelay3Switch);
  httpServer.on("/switch4", handleRelay4Switch);
  httpServer.on("/reboot", handleReboot);
  httpServer.on("/data", handleData);

  if (mqttServer.length()) {
    pubsubClient.setServer(mqttServer.c_str(), mqttPort);
```

```
    pubsubClient.setCallback(mqttCallback);
  }
}

void loop() {
  if ((WiFi.getMode() == WIFI_STA) && (WiFi.status() != WL_CONNECTED)) {
    setupWiFi();
  }
  httpServer.handleClient();
  if (mqttServer.length() && (WiFi.getMode() == WIFI_STA)) {
    if (! pubsubClient.connected())
      mqttReconnect();
    if (pubsubClient.connected())
      pubsubClient.loop();
  }
  if (millis() - WiFiModeTime > 300000) {
    if (WiFi.status() != WL_CONNECTED) {
      Serial.print("Starting reconnect to Wi-Fi: ");
      setupWiFi();
    }
    WiFiModeTime = millis();
  }
  CONTROLLER();
  R1_HF();
  R2_VENT();
  R3_PIR();
  R4_BTN();
  delay(1);
}
```

ДОДАТОК Б

Код CONTLOLLER:

```
void CONTROLLER(void) {
  if (millis() - sensorsTime > (sensreadmax.toInt() * 1000)) {
    sensors.requestTemperatures();
    sensorsTime = millis();
    dsw1 = sensors.getTempCByIndex(0);
    shtt1 = SHT2x.GetTemperature();
    shth1 = SHT2x.GetHumidity();
  }

  if (millis() - mqttsendTime > (mqttSendinterval.toInt() * 1000)) {
    mqttsendTime = millis();

    String topic;

    if (mqttServer.length() && pubsubClient.connected()) {
      topic = '/';
      topic += mqttClient;
      topic += "/dsw1";
      mqtt_publish(pubsubClient, String(topic), String(dsw1));
    }

    if (mqttServer.length() && pubsubClient.connected()) {
      topic = '/';
      topic += mqttClient;
      topic += "/shtt1";
      mqtt_publish(pubsubClient, String(topic), String(shtt1));
    }

    if (mqttServer.length() && pubsubClient.connected()) {
      topic = '/';
      topic += mqttClient;
      topic += "/shth1";
      mqtt_publish(pubsubClient, String(topic), String(shth1));
    }
    if (mqttServer.length() && pubsubClient.connected()) {
      topic = '/';
      topic += mqttClient;
      topic += mqttTopicR1;
      mqtt_publish(pubsubClient, topic, String(relay1));
    }

    if (mqttServer.length() && pubsubClient.connected()) {
      topic = '/';
      topic += mqttClient;
      topic += mqttTopicR2;
      mqtt_publish(pubsubClient, topic, String(relay2));
    }

    if (mqttServer.length() && pubsubClient.connected()) {
      topic = '/';
      topic += mqttClient;
      topic += mqttTopicR3;
      mqtt_publish(pubsubClient, topic, String(relay3));
    }
```

```
    if (mqttServer.length() && pubsubClient.connected()) {
      topic = '/';
      topic += mqttClient;
      topic += mqttTopicR4;
      mqtt_publish(pubsubClient, topic, String(relay4));
    }
  }
}
```

ДОДАТОК В

Код R1_HF:

```
void R1_HF (void) {
  if (dsw1 < dsw1max.toFloat()) {
    hf = true;
  }
  else {
    hf = false;
  }
  relay1 = digitalRead(relay1Pin);
  if (! relay1Level) {
    relay1 = ! relay1;
  }

 if (r1sw == true) {
    r1swState = true;
    countHF = millis();
  }
  if (millis() - countHF > (floormax.toInt() * 60000)) {
    r1swState = false;
    r1sw = false;
  }
  if (relay1Level == HIGH) {
    if (hf == true || r1swState == true) {
      digitalWrite(relay1Pin, HIGH);
      r1sw = false;
    }
    if (hf == false && r1swState == false) {
      digitalWrite(relay1Pin, LOW);
      //r1sw = false;
    }
  }
  if (relay1Level == LOW) {
    if (hf == true || r1swState == true) {
      digitalWrite(relay1Pin, LOW);
      r1sw = false;
    }
    if (hf == false && r1swState == false) {
      digitalWrite(relay1Pin, HIGH);
   }
  }
  if (mqttServer.length() && pubsubClient.connected()) {
    if (relay1 != lastrelay1) {
      String topic('/');
      topic += mqttClient;
      topic += mqttTopicR1;
      mqtt_publish(pubsubClient, topic, String(relay1));
      lastrelay1 = relay1;
    }
  }
}
```

ДОДАТОК Г

Код R2_VENT:

```
void R2_VENT (void) {
  if (shth1 > shth1max.toFloat()) {
    vent = true;
  }
  else {
    vent = false;
  }
  relay2 = digitalRead(relay2Pin);
  if (! relay2Level) {
    relay2 = ! relay2;
  }

  if (relay2Level == HIGH) {
    if (vent == true && r2sw == false && relay2 == LOW) {
      digitalWrite(relay2Pin, HIGH);
    }
    if (vent == false && r2sw == false && relay2 == HIGH) {
      digitalWrite(relay2Pin, LOW);
    }
    if (vent == false && r2sw == true && relay2 == HIGH) {
      digitalWrite(relay2Pin, HIGH);
    }
  }

  if (relay2Level == LOW) {
    if (vent == false && r2sw == true && relay2 == HIGH) {
      digitalWrite(relay2Pin, LOW);
    }
    if (vent == true && r2sw == false && relay2 == LOW) {
      digitalWrite(relay2Pin, LOW);
    }
    if (vent == false && r2sw == false && relay2 == HIGH) {
      digitalWrite(relay2Pin, HIGH);
    }
  }

  if (mqttServer.length() && pubsubClient.connected()) {
    if (relay2 != lastrelay2) {
      String topic('/');
      topic += mqttClient;
      topic += mqttTopicR2;
      mqtt_publish(pubsubClient, topic, String(relay2));
      lastrelay2 = relay2;
    }
  }
}
```

ДОДАТОК Д

Код R3_PIR:

```
void R3_PIR (void) {
  if (digitalRead(button1Pin) == HIGH) {
    pirSt = true;
    PirTime = millis();
  }
  if (millis() - PirTime > (PIRmax.toInt() * 1000)) {
    pirSt = false;
  }
  relay3 = digitalRead(relay3Pin);
  if (! relay3Level) {
    relay3 = ! relay3;
  }

 if (relay3Level == HIGH) {
    if (pirSt == true && r3sw == false && relay3 == LOW) {
      digitalWrite(relay3Pin, HIGH);
    }
    if (pirSt == false && r3sw == true && relay3 == LOW) {
      digitalWrite(relay3Pin, HIGH);
    }
    if (pirSt == false && r3sw == false && relay3 == LOW) {
      digitalWrite(relay3Pin, LOW);
    }
    if (pirSt == true && r3sw == false && relay3 == HIGH) {
      digitalWrite(relay3Pin, HIGH);
    }
    if (pirSt == false && r3sw == false && relay3 == HIGH) {
      digitalWrite(relay3Pin, LOW);
    }
    if (pirSt == false && r3sw == true && relay3 == HIGH) {
      digitalWrite(relay3Pin, HIGH);
    }
  }

  if (relay3Level == LOW) {
    if (pirSt == false && r3sw == true && relay3 == HIGH) {
      digitalWrite(relay3Pin, LOW);
    }
    if (pirSt == true && r3sw == false && relay3 == LOW) {
      digitalWrite(relay3Pin, LOW);
    }
    if (pirSt == false && r3sw == false && relay3 == HIGH) {
      digitalWrite(relay3Pin, HIGH);
    }
  }
  if (mqttServer.length() && pubsubClient.connected()) {
    if (relay3 != lastrelay3) {
      String topic('/');
      topic += mqttClient;
      topic += mqttTopicR3;
      mqtt_publish(pubsubClient, topic, String(relay3));
      lastrelay3 = relay3;
    }
  }
}
```

ДОДАТОК Е

Код R4_BTN:

```
void R4_BTN (void) {
  int reading2 = digitalRead(button2Pin);
  relay4 = digitalRead(relay4Pin);
  if (! relay4Level)
    relay4 = ! relay4;
  if (reading2 != lastButton2State) {
    lastDebounce2Time = millis();
  }
  if ((millis() - lastDebounce2Time) > debounceDelay) {
    if (reading2 != button2State) {
      button2State = reading2;
      if (button2State == LOW) {
        if (BTN2State == LOW && relay4 == LOW) {
         switchRelay4(true);
        }
        if (BTN2State == HIGH && relay4 == LOW) {
         switchRelay4(true);
        }
        if (BTN2State == HIGH && relay4 == HIGH) {
         switchRelay4(false);
        }
        if (BTN2State == LOW && relay4 == HIGH) {
         switchRelay4(false);
        }
      }
      BTN2State = !BTN2State;
    }
  }
  lastButton2State = reading2;
}
```

ДОДАТОК Ж

Код SSDP:

```
void SSDP_init(void) {

  httpServer.on("/description.xml", HTTP_GET, []() {
    SSDP.schema(httpServer.client());
  });
  SSDP.setDeviceType("upnp:rootdevice");
  SSDP.setSchemaURL("description.xml");
  SSDP.setHTTPPort(80);
  SSDP.setName(WiFi_hostname);
  SSDP.setSerialNumber(ESP.getChipId());
  SSDP.setURL("/");
  SSDP.setModelName("ESP-HUM/PIR_Controller");
  SSDP.setModelNumber("000000000001");
  SSDP.setModelURL("http://wd.zp.ua/");
  SSDP.setManufacturer("Dneprovskiy Vadim");
  SSDP.setManufacturerURL("http://wd.zp.ua");
  SSDP.begin();
}
```