

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Факультет комп'ютерних наук та технологій
Кафедра комп'ютерних систем та мереж

Пояснювальна записка

до дипломного проекту (роботи)

магістра

(ступінь вищої освіти (освітній ступінь))

на тему СИСТЕМА КЕРУВАННЯ ТЕПЛИЧНИМ ГОСПОДАРСТВОМ
НА БАЗІ МІКРОКОНТРОЛЕРІВ

Виконав: студент 2 курсу, групи КНТ-522м
спеціальності _____

123 Комп'ютерна інженерія

Освітня програма (спеціалізація)

Комп'ютерні системи та мережі

(код і назва спеціальності)

ЛИСКА Д. М.

(прізвище та ініціали)

Керівник ІЛ'ЯШЕНКО М.Б.

(прізвище та ініціали)

Рецензент ЄФИМЕНКО М.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування вищого навчального закладу)

Факультет Комп'ютерних наук і технологій
Кафедра «Комп'ютерні системи та мережі»
Ступінь вищої освіти (освітній ступінь) магістерський
Спеціальність 123 Комп'ютерна інженерія
(код і назва)
Освітня програма (спеціалізація) Комп'ютерні системи та мережі
(назва)

Зав. кафедри Кудерметов Р.К.

“ ” 2023 року

ЗАВДАННЯ НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

ЛИСКА Денис Миколайович

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Система керування тепличним господарством на базі мікроконтролерів

керівник проекту (роботи) ІЛ'ЯШЕНКО Матвій Борисович, к. т. н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від “24” жовтня 2023 року № 400

2. Строк подання студентом проекту (роботи) 01 грудня 2023 року

3. Вихідні дані до проекту (роботи) програмування Python, протокол MQTT, ОС Raspbian

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) _____

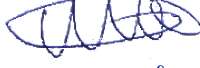
1) Сучасні автоматизовані системи

2) Розробка системи керування;

3) Практична реалізація.

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

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-3	ІЛЛЯШЕНКО М.Б., к. т. н., доцент		
Нормоконтроль	ПОЛЬСЬКА О.В., ст. викл.		

7. Дата видачі завдання 01.10.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Аналіз технічного завдання	10.10.2023 р.	
2	Вибір апаратного забезпечення	15.10.2023 р.	
3	Вибір програмного забезпечення	01.10.2023 р.	
4	Проектна реалізація	25.11.2023 р.	
5	Проектування програмних засобів	01.12.2023 р.	
6	Оформлення отриманих результатів у ПЗ	10.11.2023 р.	
7	Оформлення графічного матеріалу	15.11.2023 р.	
8	Захист роботи	16.12.2023 р.	

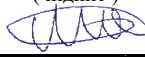
Студент


(підпис)

Денис ЛІСКА

(ініціали та прізвище)

Керівник проекту (роботи)


(підпис)

Матвій ІЛЛЯШЕНКО

(ініціали та прізвище)

РЕФЕРАТ

ПЗ: 73 с., 42 рис., 7 табл., 20 джерел.

РОЗУМНА ТЕПЛИЦЯ, ПРОГРАМНО-АПАРАТНИЙ КОМПЛЕКС, ВЕБ-САЙТ, LINUX.

У цій роботі описано процес і результати проєктування та реалізації програмно-апаратного комплексу управління тепличним господарством, а також вебсайту, що входить до його складу.

Комплекс реалізовано на базі клієнт-серверної архітектури, де сервер реалізовано на базі ОС Raspbian. У комплексі застосовано програмні продукти Arduino IDE, Apache 2.4, Mosquitto, СУБД MariaDB, протокол MQTT, а також мови програмування Python з набором бібліотек, C, мови розмітки HTML5 і CSS3, а також використовується мікрофреймворк flask [6].

Цей програмно-апаратний комплекс створювався для спрощення вирощування рослин і може автоматизувати процес підтримання температурного режиму (відчинення вікон на провітрювання, обігрів) і процес поливу. Також реалізовано ручне керування теплицями на сайті та виведення поточної температури і вологості та збереження їх у базі даних для кожної теплиці. Додатково налаштовано роботу освітлення для денного та нічного часу за рухом і вручну. Однак, ця конфігурація також підійде для акваріумів, тераріумів та інших місць, де необхідно підтримувати температурний режим і/або керувати будь-яким навантаженням.

Реалізований програмно-апаратний комплекс повністю відповідає технічному завданню на розробку.

РЕФЕРАТ

Explanatory note to the master's work: 73 p., 42 figures, 7 tables, 20 sources.

SMART GREENHOUSE, HARDWARE-SOFTWARE COMPLEX, WEBSITE,
LINUX.

This work describes the process and results of the design and implementation of the software and hardware complex for greenhouse management, as well as the website included in it.

The complex is implemented on the basis of the client-server architecture, where the server is implemented on the basis of the Raspbian OS. The complex uses Arduino IDE, Apache 2.4, Mosquitto, MariaDB DBMS, MQTT protocol, as well as Python programming languages with a set of libraries, C, HTML5 and CSS3 markup languages, as well as the flask microframework [6].

This hardware and software complex was created to simplify the cultivation of plants and can automate the process of maintaining the temperature regime (opening windows for ventilation, heating) and the watering process. Manual management of greenhouses on the site and output of current temperature and humidity and saving them in the database for each greenhouse are also implemented. In addition, the operation of lighting for day and night time according to movement and manually is configured. However, this configuration is also suitable for aquariums, terrariums and other places where it is necessary to maintain a temperature regime and/or manage any load.

The implemented software and hardware complex fully meets the technical development task.

ЗМІСТ

Вступ	7
1 Сучасні автоматизовані системи	8
1.1 Технічне та програмне забезпечення інтернету речей	8
1.2 Порівняння та вибір технічних характеристик складових	15
2 Розробка системи керування	33
2.1 Розробка та налаштування об'єкта розробки	33
2.2 Вимоги до моніторингу	54
2.3 Загальні відомості про побудову моніторингу	58
3 Практична реалізація	65
3.1 Опис продукту	65
3.2 Спеціалізовані показники. Підключення бібліотек	69
Висновки	71
Перелік джерел посилання	72

ВСТУП

Популярність автоматизованих систем та інших пристроїв, що полегшують життя людини, стає дедалі більшою. І часто виникає потреба автоматизувати різні процеси, такі як вирощування різних рослин у теплиці. Люди, які мають дачні ділянки, не завжди можуть бути в потрібний час, щоб відчинити вікно для провітрювання або ж полити рослини в теплиці, та й перспектива добиратися до місця, щоб тільки відчинити вікно, не завжди радує. Тому популярність різноманітних автоматичних систем у побуті та в промисловості стає дедалі більшою.

Проект актуальний тим, що цей програмно-апаратний комплекс допоможе тим, у нашому випадку, хто хоче вирощувати гарний врожай у теплиці з мінімальними витратами сил не тільки вдома, а й на виробництві.

Предметом розробки є система управління тепличним господарством. Метою розробки є створення програмно-апаратного комплексу з автоматизованим і ручним керуванням теплицями через веб-інтерфейс зі збереженням даних про клімат і їх виведенням.

Завдання:

- провести аналіз стану проблеми та підходів до її вирішення;
- вивчити й обґрунтувати вибір програмних і апаратних технологій реалізації;
- вивчити принципи побудови автоматизованих систем;
- створити фізичну модель теплиці;
- створити програмно-апаратний комплекс на базі мікроконтролера та написати веб-інтерфейс;
- реалізувати mqtt-брокер і веб-сервер, на якому розміщено написаний веб-інтерфейс централізованого управління всіма теплицями і відображенням даних про них;
- реалізувати автоматичний і ручний режим роботи.

1 СУЧАСНІ АВТОМАТИЗОВАНІ СИСТЕМИ

1.1 Технічне та програмне забезпечення інтернету речей

У цьому світі цифрових технологій, який швидко розвивається, у великій кількості виробляють різні пристрої для автоматизації рутинних дій, щоб звільнити час людини, який можна витратити на цікавіші справи. Наприклад, посудомийки, роботи-пилососи, кава-машини та багато інших пристроїв були створені, щоб полегшити життя людей.

У багатьох у володінні є дачні ділянки, сади та інше, за якими необхідно доглядати і стежити, щоб отримати гідний урожай, на що йде дуже багато часу і сил, як і в промисловій сфері. Тому автоматизація процесів вирощування, з мінімальною участю людини, завжди актуальна. У цій ситуації необхідно автоматизувати цілу низку різних процесів, що зі свого боку утворює автоматизовану систему (АС), яка являє собою комплекс технічних і програмних засобів, що забезпечує централізоване управління об'єктом у виробничому або адміністративному середовищі. Прикладами застосування АС є виробництва автомобілів, побутової техніки, великої техніки тощо.

Розумні теплиці можна класифікувати таким чином:

- автономні - всі системи працюють виключно на тепловій або сонячній енергії;
- енергозалежні - живлення елементів здійснюється від підведеної електромережі.

Кожен тип має свої переваги і переваги, про важливість яких суперечки не вщухають і донині.

Автоматизована теплиця передбачає виконання низки операцій без участі людини, а саме:

- підтримка необхідної температури всередині, з урахуванням показань датчиків;
- провітрювання, шляхом відкривання вікон;

- автополив рослин, з урахуванням вологості ґрунту.

Багато автоматизованих систем мають локальне управління або ж не мають зовсім, покладаючись лише на автоматичні параметри. Окрім як виконання низки дій автоматично, АС може мати доступ до інтернету та можливість керування системою віддалено, без безпосередньої присутності людини поблизу системи. Такі системи називають "інтернет речей".

У сучасному світі в ужиток увійшла фраза "інтернет речей". Інтернет речей (Internet of Things, IoT) - це нова стадія розвитку інтернету, коли до нього підключено пристроїв більше, ніж людей. Перехід до неї стався у 2008-2009 роках.

роках, коли кількість пристроїв у мережі обігнала чисельність населення Землі. Інтернет речей дає змогу знижувати трудовитрати й автоматизувати процеси компаніям, в електроенергетиці покращує контрольованість підстанцій і ліній електропередачі завдяки дистанційному моніторингу, а в охороні здоров'я дає змогу перейти на новий рівень діагностики захворювань - "розумні" пристрої контролюють показники здоров'я пацієнта. У сільському господарстві "розумні" ферми і теплиці самі керують кількістю добрив і води. Розвиток інтернету речей у логістиці дає змогу скоротити витрати на вантажоперевезення і мінімізувати вплив людського фактора. Також IoT активно впроваджують нафтогазові та гірничодобувні галузі. Застосування поглибленої аналітики по бурових свердловинах допомагає нафтогазовій промисловості збільшити обсяги видобутку на вже відпрацьованих родовищах. Інтернет речей у транспорті складається з самого транспорту, навігаторів, електронних табло, камер спостереження, систем безпеки.

Розвиток інтернету речей став можливим завдяки широкому поширенню інтернету, смартфонів, бездротових мереж, здешевлення електронних компонентів і обробки даних. На практиці IoT-системи зазвичай складаються з мережі розумних пристроїв і хмарної платформи, до якої вони підключені. До них примикають системи зберігання, оброблення та захисту, зібраних датчиками даних.

IoT - складається зі стека технологій. Це і створення датчиків, і безліч протоколів їхньої взаємодії. Об'єкти можуть спілкуватися між собою через wi-fi, Bluetooth, LPWAN, BLE, Ethernet, RFID, ZigBee та інші види бездротового зв'язку.

Міжмашинна взаємодія (M2M) є окремим випадком інтернету речей.

Архітектура Інтернету речей передбачає наявність таких функціональних рівнів: мережа датчиків, шлюз, управління.

Оскільки нижній рівень складається з датчиків, сенсорів і реле, то одразу ж виникає потреба в "особливих" протоколах для забезпечення взаємодії пристроїв один з одним і з верхніми рівнями. Стандартні прикладні протоколи не підходять з огляду на їхню непристосованість до умов мережі інтернету речей. Датчик, зазвичай мініатюрний, з невеликою пам'яттю, вимірює фізичні параметри в режимі реального часу, найчастіше в умовах низького енергозабезпечення. Результати вимірювань обробляються сенсорним вузлом і передаються на сервер. Обсяг інформації, що формується одним сенсорним вузлом, порівняно невеликий, проте більшість сервісів "Інтернету речей" побудовано на принципі обробки інформації від безлічі вузлів, що принципово відрізняється від архітектур, прийнятих у класичних мережах, типу абонент - вузол зв'язку для телефонії, клієнт-сервер для передавання даних.

В інтернеті речей джерелами даних є різноманітні датчики, які, своєю чергою, можуть підключатися до мікроконтролерів, для зручного збирання та надсилання даних для подальшого аналізу. Часто створюються проєкти автоматизованих систем керування на базі мікроконтролерів. Мікроконтролер - це спеціальна мікросхема, призначена для керування різними електронними пристроями.

Мікроконтролер являє собою об'єднаний процесор, пам'ять, ПЗП, ПЗП периферія всередині одного корпусу, який зовні схожий на звичайну мікросхему. Виробництво мікроконтролерів щорічно у багато разів перевищує виробництво процесорів, а потреба в них не знижується.

Відбувається через те, що мікроконтролери застосовуються переважно у вбудованих системах, в іграшках, у верстатах, у масовій домашній техніці, у домашній автоматичній, де більше потрібен баланс між ціною достатньою функціональністю, ніж потужність процесора.

Наявність контролера легко виявити в будь-якому більш-менш інтелектуальному пристрої: від дитячої іграшки до бездротової гарнітури стільникового телефону, оскільки спосіб використання і застосування мікроконтролерів у сучасному світі дуже різноманітний.

Мікроконтролер, у процесі роботи, зчитує команди з пам'яті або порту введення і виконує їх. Що означає кожна команда, визначається системою команд мікроконтролера. В архітектурі мікроконтролера закладена система команд, і виконання коду команди виражається в проведенні внутрішніми елементами мікросхеми певних мікрооперацій, здебільшого пам'ять у мікроконтролерах становить від 2 до 128 Кб, буває й більше, то використовують спеціальні версії мови Бейсіка, Паскаля, але здебільшого - Сі. Якщо менше, то писати доводиться асемблером або Форті. До того як остаточно запрограмувати мікроконтролер, його тестують в емуляторах - програмних або апаратних. Мікроконтролер - це вже не процесор, але ще й не комп'ютер [3].

Мікроконтролери для масового споживача відрізняються наявністю попередньо прошитого в них завантажувача. За допомогою завантажувача користувач, без використання окремих апаратних програматорів, завантажує свою програму в мікроконтролер. Завантажувач з'єднується з комп'ютером через інтерфейс USB (якщо він є на платі) або за допомогою окремого перехідника UART-USB. У безкоштовну програму Arduino IDE вбудована підтримка завантажувача і виконується в одне клацання миші [3].

Програмісту доступні деякі спеціальні можливості портів введення-виведення мікроконтролерів, наприклад, широтно-імпульсна модуляція (ШІМ), аналогово-цифровий перетворювач (АЦП), інтерфейси UART, SPI, I2C. Кількість і можливості портів введення-виведення визначаються конкретним варіантом плати.

Крім портів на платах мікроконтролерів іноді встановлюється периферія у вигляді інтерфейсів USB або Ethernet. Опціональний набір зовнішньої периферії на модулях розширення містить у собі:

- USB Device (найчастіше як віртуальний COM-порт через FTDI FT232, є також версії з емуляцією USB HID Class клавіатур і мишок);
- дротовий і бездротовий модулі зв'язку як на основній платі, так і на платах розширення;
- модуль GSM та інші бездротові інтерфейси;
- модуль для підключення SD карт;
- модулі керування моторами. Можуть підключатися також реле,
- електромагніти тощо;
- графічний ІК індикатор;
- модуль із макетним полем.

Сторонні виробники випускають широку гаму датчиків і виконавчих пристроїв, що підключаються до мікроконтролерів. Наприклад, гіроскопи, компаси, манометри, гігрометри, термометри, релейні модулі, індикатори, клавіатури тощо [20].

Найчастіше мікроконтролери, які керують різними пристроями, вводять у дротову або бездротову мережу.

Однією з основних частин в автоматизованій системі управління є програмне забезпечення для управління, збору, збереження, аналізу та виведення інформації. Для цього часто використовується веб-сервіс, який розташовується на веб-сервері.

Веб-сервер (web-server) - це сервер, який відповідає за приймання запитів від клієнтів до веб-сайту та їх обробку. Це може бути який-небудь комп'ютер, який спеціально виділено з групи персональних комп'ютерів, або робоча станція, на яких встановлено і працює сервісне програмне забезпечення. Клієнтами зазвичай виступають веб-браузери. Веб-сервер видає клієнтам HTTP-відповіді, здебільшого разом із HTML-сторінкою, що може містити: різноманітні файли, зображення, медіа-потік або будь-які інші дані. HTTP (англ. HyperText Transfer Protocol -

протокол передачі гіпертексту) - це мережевий протокол прикладного рівня передачі даних. Основним принципом протоколу HTTP є технологія "клієнт-сервер", що забезпечує взаємодію мережі та користувача.

Веб-сервер виконує різні функції виконання скриптів, написаних мовами програмування, як-от PHP, Python тощо, які відповідають за організацію запитів до мережевих служб, баз даних, доступу до файлів, пересилання електронної пошти та інших додатків електронної комерції.

Усі основні та додаткові функції веб-сервера:

- приймання запитів від веб-браузерів за протоколом стандарту HTTP і HTTPS з використанням мережевих протоколів TCP/IP;
- запуск прикладних програм на веб-сервері з подальшим передаванням і поверненням параметрів обробки;
- обслуговування та обробка запитів, типу: mailto, FTP, Telnet тощо;
- виконання пошуку і відсилання файлів з гіпертекстом або будь-яких документів у браузер за протоколом HTTP;
- завантаження Java-додатків;
- авторизація користувачів та їхня автентифікація;
- робота та обслуговування навігаційних карт зображень (Image map);
- адміністрація та оперативне управління сервером;
- підтримка сторінок, які генеруються динамічно;
- ведення реєстраційного журналу звернень користувачів до різних ресурсів;
- виконання скриптів, наприклад, написаних мовами програмування, таких як PHP, Python тощо.

На самому початку етапу розвитку веб-розробки, всі необхідні дані вказувалися безпосередньо в коді сторінки. Проте в сучасному середовищі, для отримання необхідної інформації можна використовувати скрипти для запитування даних із бази даних. Спеціальні алгоритми для зберігання та пошуку інформації в

базах даних дозволяють знаходити необхідні відомості буквально за декілька секунд.

Одні з найпоширеніших веб-серверів: Apache (компанія Apache Software Foundation), IIS (компанія Microsoft) і iPlanet server (від компаній Sun Microsystems і Netscape Communications Corporation). Наразі на ринку існує величезний вибір веб-серверів, як комерційних, так і безкоштовних.

Для повного функціонування сайту потрібні не тільки файли з кодом сторінок, а й бази даних. База даних являє собою певний набір даних, які, як правило, пов'язані об'єднувальною ознакою або властивостями.

Велика кількість різних даних, які можуть бути поміщені в єдину базу, веде до безлічі варіацій того, що може бути записано: особисті дані користувачів, записи, дати, замовлення тощо. Це зручно тим, що інформацію можна швидко заносити в базу даних і так само швидко її витягувати за необхідності.

На зорі розвитку web-розробки всі необхідні дані потрібно було прописувати в коді сторінки, зараз же потрібну інформацію можна запросити з бази даних за допомогою скриптів. Спеціальні алгоритми зберігання і пошуку інформації, які використовуються в базах даних, дають змогу знаходити потрібні відомості буквально за частки секунд.

Для взаємодії з базами даних використовуються системи керування базами даних (СКБД), що є програмним забезпеченням, яке використовується для створення і роботи з базами даних. Головною функцією СКБД є управління даними, обов'язково підтримує мови баз даних, а також відповідає за копіювання та відновлення даних після будь-яких збоїв. Додатково СКБД дає змогу швидко обробляти клієнтські запити та видавати актуальну інформацію.

Багато систем "розумних" теплиць налаштовані на певний функціонал з мінімальною можливістю його розширення або ж спрямовані на управління тільки однією теплицею.

Для автоматизації процесу вирощування рослин буде реалізовано проєкт "розумної" теплиці, в основі якого послужить мікроконтролер, а також централізована система управління.

1.2 Порівняння та вибір технічних характеристик складових

У світі мікроконтролерів існує величезна кількість моделей, які підходять під будь-які завдання і вимоги користувачів, від різних виробників і з різними характеристиками. Тому необхідно вибрати мікроконтролер, що підходить під наш проєкт.

Вибір відбувався з моделей різних виробників, що перебувають в одному сегменті, для виявлення необхідного мікроконтролера для реалізації проєкту. Порівняльну характеристику мікроконтролерів наведено в таблиці 1.1

Таблиця 1. Характеристики NodeMCU (ESP8266)

Характеристика	Значення	Значення	Значення
1	2	3	4
Процесор	Tensilica Xtensa L106, частота 80 МГц (до 160 МГц) 32 розряди з низьким енергоспоживанням	ATmega328 16 МГц	ARM Cortex-M3 72 МГц 32 розряди
ОЗП	128 КБ	2 КБ	20 КБ
Сховище	4 мегабайти Flash-пам'яті	32 КБ	64 КБ
Wi-Fi	802.11 b/g/n	Ні (доп. модуль)	Ні (доп. модуль)
Мережеві протоколи	IPv4, TCP/UDP/HTTP/FTP	-	-
USB-порти	1 порт MicroUSB	1 x USB	1 x MicroUSB
Периферія	14 портів введення-виведення	14 (6 з яких можуть використовуватися як виходи ШІМ+аналогові входи 6)	14 (6 з яких можуть використовуватися як виходи ШІМ+аналогові входи 6)
Живлення	5 В, <0,5-2.5 А через порт micro-USB або GPIO	7-12 В	7-12 В
Розміри	60 мм x 30 мм x 0,8 мм	69 x 53 мм	69 x 53 мм
Підтримувані	C, Lua, MicroPython, JavaScript, Basic, Lisp	C	C

Для створення проєкту обрано модель NodeMCU (ESP8266). На рисунку 1.1 зображено NodeMCU (ESP8266).

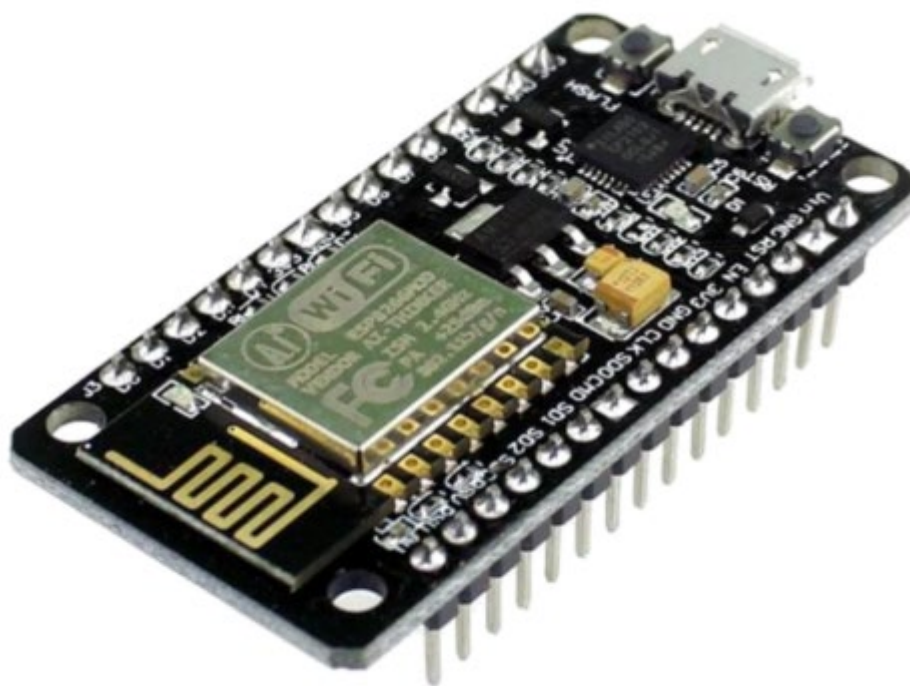


Рисунок 1.1 - NodeMCU (ESP8266)

Ця плата має хороші технічні характеристики, функціонал, придатний для майбутнього поліпшення проєкту і під'єднання додаткових модулів бездротового зв'язку.

Мікроконтролер ESP8266 створений для використання в розумних розетках, mesh-мережах (Mesh мережа - це розподілена, однорангова, пориста мережа), IP-камерах, бездротових сенсорах, переносній електроніці тощо. Одним словом, ESP8266 з'явився на світ, щоб стати мозком прийдешнього "Інтернету речей".

Чіп ESP8266 є одним із найбільш високоінтегрованих рішень для роботи з Wi-Fi [20]. Так і в нашому проєкті мікроконтролери та сервер з'єднуюватимуться бездротовою мережею Wi-Fi.

Wi-Fi - це абревіатура, яка походить від англійського словосполучення Wireless Fidelity, що означає "бездротова передача даних". Це протокол і стандарт на обладнання для широкосмугового радіозв'язку, призначеного для організації

локальних бездротових мереж. Цей вид зв'язку добре підходить для нашого проекту, коли теплиці розташовані на деякій відстані від вузла управління. Wi-Fi підтримується величезною кількістю пристроїв, що відрізняються простим налаштуванням і досить невеликою ціною.

Усередині чіпа ESP8266 вмістилася купа всього того, що в конкуруючих рішеннях часто є частиною зовнішньої обв'язки.

Передбачено два варіанти використання чіпа. Перший варіант у вигляді моста UART-WIFI, коли модуль на базі ESP8266 під'єднується до наявного рішення на базі будь-якого іншого мікроконтролера і керується AT-командами, забезпечуючи зв'язок рішення з інфраструктурою Wi-Fi. Цей варіант добре тим, у кого вже є в руках готова схематика і налагоджена прошивка на базі чогось свого. Другий, реалізуючи нове рішення, що використовує сам чіп ESP8266 як керуючий мікроконтролер.

У проєкті буде використовуватися другий варіант у вигляді плати проєкту NodeMCU, де встановлений модуль ESP-12E, в якому розміщено ESP8266 вбудований у 32-бітний мікроконтролер (MCU) Tensilica L106, що вирізняється ультранизьким енергоспоживанням. Тактова частота становить 80 МГц, також вона може досягти максимального значення 160 МГц, і зовнішня flash-пам'ять на 4 МБайта. Модуль ESP-12E показаний на рисунку 2 [20].

Далі на платі NodeMCU реалізовано CP2102 - перетворювач інтерфейсу USB в UART, можливість живлення плати від 5 вольт і роз'єм MicroUSB для підключення до комп'ютера.

Модуль підтримує стандарт IEEE802.11 b/g/n, повний стек TCP/IP протоколів. Різні WiFi режими: станція, програмна точка доступу, програмна точка доступу і станція. Безпека WPA, WPA2 і шифрування WEP, TKIP, AES. Також має можливість оновлення прошивки через UART, OTA (через мережу) або завантаження і запис прошивки через хост. Додатково підтримуються мережеві протоколи: IPv4, TCP, UDP, HTTP, FTP, що буде дуже зручно для реалізації простого веб-сервера [22].

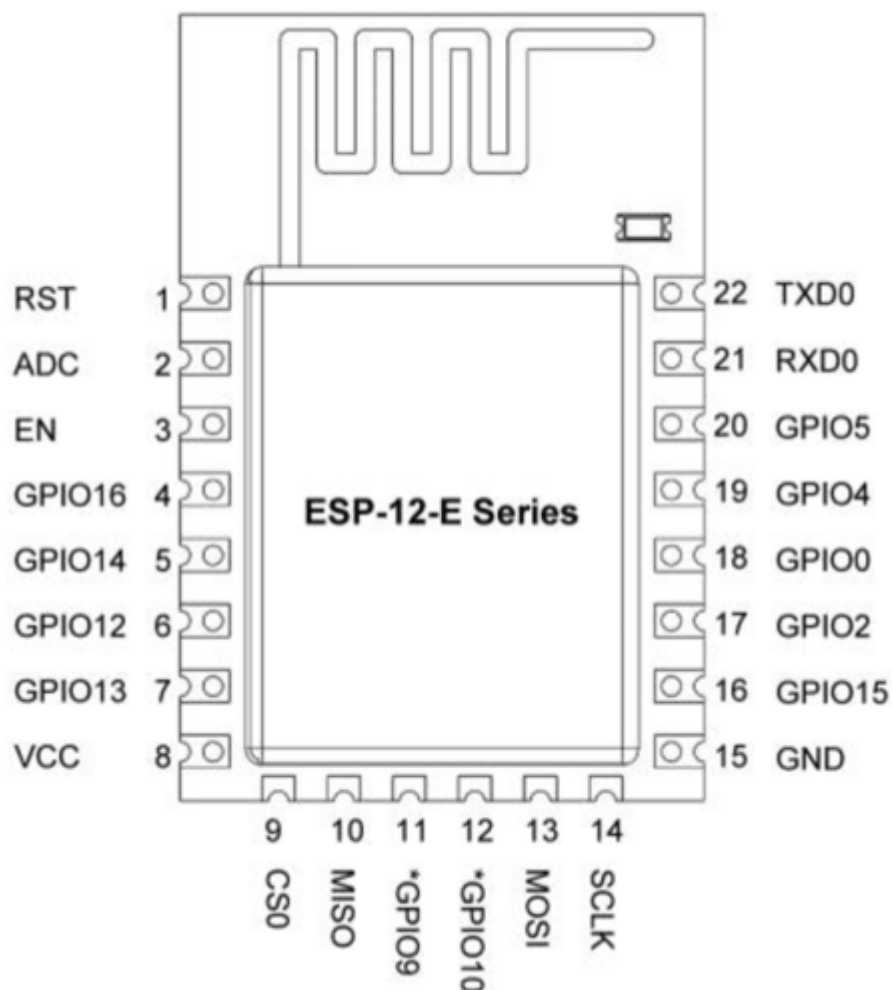


Рисунок 1.2 - ESP-12E

Розмір плати NodeMCU буде в рази меншим, лише 6 сантиметрів завдовжки і 3 завширшки, ніж у конкурентів із модулями, що виконують такий самий функціонал. Живлення плати NodeMCU походить від 5-20 вольт з подальшим зниженням до 3.3 вольт для живлення модуля ESP-12E.

У ніші мінікомп'ютерів існує величезна кількість моделей, що підходять під будь-які вимоги користувачів, від різних виробників і з різними характеристиками. Вибір відбувався з моделей різних виробників, що перебувають в одному сегменті, для виявлення необхідного міні-комп'ютера для реалізації проєкту. Порівняльну характеристику мінікомп'ютерів наведено в таблиці 1.2.

Таблиця 1.2 - Характеристики мінікомп'ютерів

Параметр	Значення	Значення	Значення
Ім'я мініПК	Raspberry Pi 3 модель B	Orange Pi PC	Banana Pi M2
Система на кристалі (SoC)	Broadcom BCM2837 (CPU + GPU)	Allwinner H3 (CPU + GPU)	Allwinner A20
Процесор	64-бітний чотирьохядерний ARMv8 Cortex-A53, 1.2 ГГц	Чотирьохядерний 32-бітний процесор ARM Cortex-A7 1.2-1.6 ГГц	ARM Cortex-A7 подвійний core 1.0 ГГц
Графічний процесор	Двоядерний процесор (GPU) 400МГц здатний кодувати, декодувати і виводити FULL-HD відео	Mali400 MP2 600 МГц	ARM Mali400 MP2
ОЗП	1 ГБайт	1 ГБайт	1 ГБайт
Сховище	слот для карти пам'яті MicroSDHC, USB Boot Mode	microSD до 64 ГБ (у як постійна пам'яті за завантажувального диска)	слот для карти пам'яті MicroSDHC
Ethernet	10/100 Мбіт Ethernet RJ45	10/100 Мбіт/с Ethernet RJ45	2× USB 2.0, 1x USB OTG
Wi-Fi/Bluetooth	Wi-Fi 802.11n і Bluetooth 4.1	ні	Wi-Fi 802.11n і
Відео вхід	1 x HDMI вихід, 1 x відеовихід 3.5 мм роз'єм 1 x DSI (Display Serial Interface) для підключення штатного дисплея	1 x HDMI вихід, 1 x відеовихід 3.5 мм роз'єм	1 x HDMI вихід, 1 x відеовихід 3.5 мм роз'єм
USB-порти	4 порти USB 2.0	2 порти USB 2.0	4 порти USB 2.0
Периферія	40 портів введення-виведення загального загального призначення (GPIO)	40 портів введення-виведення загального загального призначення (GPIO)	40 портів введення-виведення загального загального призначення (GPIO)
Живлення	5 В, 2-2.5 А через порт micro-USB або GPIO	5 В 2 А	5 В 2 А
Розміри	85.6 мм x 56.5 мм x 17 мм	85 x 55 мм	92 × 60 мм
Підтримувані ОС	Ubuntu, Debian, Fedora, Arch Linux, Gentoo, RISC OS, Android, Firefox OS, NetBSD, FreeBSD, Slackware, Tiny Core Linux, Windows 10 IOT	Android, Ubuntu,	Android, Ubuntu

Вузол, що відповідає за збір і передачу даних, а також виступає в ролі веб-сервера, вирішено вибрати серед одноплатних мінікомп'ютерів, чийх потужностей у цьому проєкті достатньо, щоб реалізувати задумане.

Для створення проєкту обрано модель Raspberry Pi 3. На рисунку 1.3 зображено Raspberry Pi 3 Model B з підписаними елементами, що знаходяться на платі.

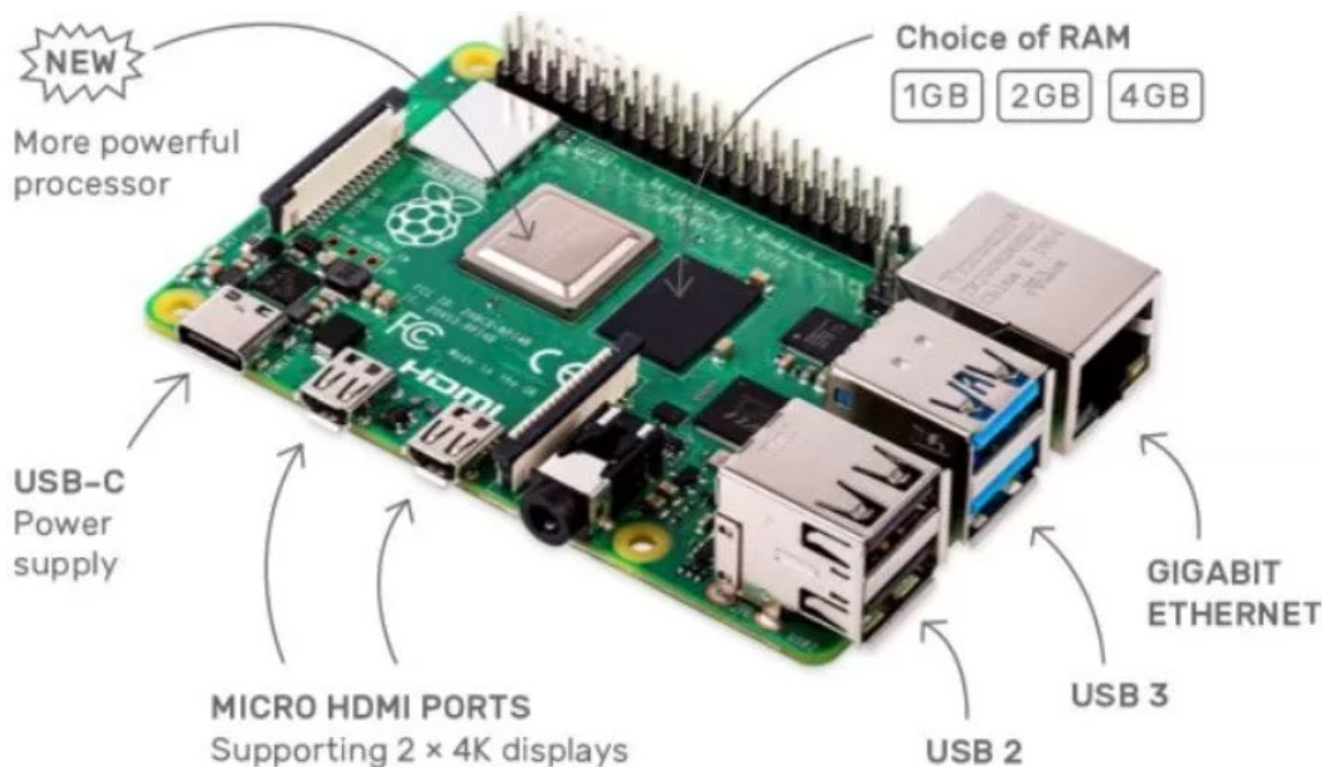


Рисунок 1.3 - Raspberry Pi 3 Model B

Дана модель має хороші технічні характеристики, функціонал і відповідні обчислювальні потужності для майбутнього поліпшення проєкту і під'єднання додаткових модулів, а також задіяння в інших проєктах одночасно [24].

Живлення Raspberry Pi 3 Model B здійснюється від 5-вольтового адаптера через роз'єм micro-USB або контакти живлення на GPIO.

У теплиці необхідно вимірювати щонайменше температуру, вологість повітря та ґрунту. Для цього було вирішено використовувати електронні датчики, здатні на вимірювання необхідних параметрів [3].

Для вимірювання температури і вологості повітря необхідно підібрати датчик, що підходить за характеристиками, який буде використовуватися в цій роботі. Порівняння характеристик датчиків, з яких відбувався вибір, представлено в таблиці 3.

Таблиця 1.3 - Порівняння датчиків

Найменування характеристики	DHT11	DHT22	DHT21
Напруга живлення, В	3-5	3-5	3-5
Температура, хв °C	0 ± 2	$-40 \pm 0,5$	$-40 \pm 0,5$
Температура, макс °C	50 ± 2	$125 \pm 0,5$	$80 \pm 0,5$
Вологість, хв %	20	0	0
Вологість, макс %	80	100	100

Для вимірювання температури та вологості повітря обрано датчик DHT22 з напругою живлення від 3 до 5 вольт постійного струму, який має більші діапазони вимірювання температури та вологості з меншими похибками.



Рисунок 1.4 - Датчик DHT22

Також цей датчик має менші розміри, ніж DHT21. Обраний датчик зображено на рисунку 1.4.

Також одним із головних параметрів, який необхідно відстежувати, це вологість ґрунту, для визначення моменту поливу. Для цього необхідно вибрати відповідний датчик - гігрометр, характеристика якого подана в таблиці 1.4.

Таблиця 1.4 - Порівняльна характеристика датчиків

Найменування характеристики	FC-28	YL-69	ICB510-01
Напруга живлення, В	3.3, 5	3.3 - 5	5 В, 12-24 В
Вихід	цифровий і аналоговий	цифровий і аналоговий	аналоговий
Споживаний струм, мА	35	35	50
Розмір, мм	30×60	20×60	50×208

Обрано датчик YL-69 через свої розміри, можливість використовувати з аналоговим і цифровим входами контролера і вартість. Цей датчик зображено на рисунку 1.5.

Цей датчик під'єднують до плати керування з потенціометром (компаратор), який слугує для завдання порогового значення в разі під'єднання датчика до цифрового контакту.

Далі необхідно визначати час доби, для цього використовується фоторезистор, виготовляють його різні компанії, але він має однакові характеристики плюс мінус на похибки, і порівнювати їх не має сенсу. Тому беремо фоторезистор із компаратором, для зручності задавання порогового значення, оскільки нам необхідно визначати лише день і ніч, то й під'єднуватиметься він до цифрового контакту контролера.

Фоторезистор - це датчик, електричний опір якого змінюється залежно від інтенсивності падаючого на нього світла. Що інтенсивніше світло, то менший опір елемента.



Рисунок 1.5 - Датчик вологості ґрунту YL-69

На рисунку 1.6 показано датчик освітленості з компаратором [3].

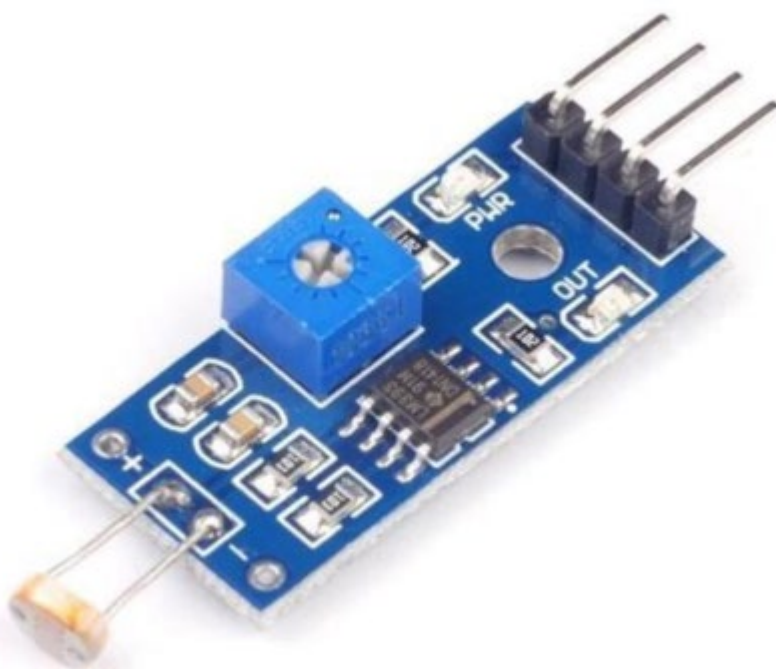


Рисунок 1.6 - Фоторезистор із компаратором

Для того, щоб знизити температуру всередині теплиці, необхідно випускати гаряче повітря назовні, для цього буде використовуватися сервопривід. Оскільки в цьому проєкті представлено приблизну модель теплиці, то для відчинення вікна використовуватиметься невеликий сервопривід. У таблиці 1.5 наведено порівняльну характеристику трьох моделей сервопривода [3].

Таблиця 1.5. Порівняльні характеристики сервоприводів

Найменування характеристики	SG-90	Turnigy TSS-10MG	ASMC-04B
Напруга живлення, В	5В	5В	12 В
Кут повороту ротора, градуси	360	360	300
Крутний момент	1.6 кг/см	2.2 кг/см	90 кг/см
Внутрішній інтерфейс	аналоговий	цифровий	аналоговий
Напрямок	проти годинникової стрілки	проти годинникової стрілки	проти годинникової стрілки
Матеріал шестерні	нейлон	метал	метал
Матеріал корпусу	пластик	пластик	метал
Довжина проводів	20 см	20 см	30 см
Габарити	23.2 × 12.5 × 22 мм	23.1 x12.0 x24.9 мм	60*95*102 мм
Вага	9 г	15 г	530 г

Для використання в проєкті було обрано сервопривід SG-90. Він має невелику вагу і робочу напругу живлення 5 Вольт, також, як і більшість компонентів проєкту. Має достатній крутний момент і невеликий розмір. Його буде достатньо, щоб реалізувати необхідний функціонал. Обраний сервопривід представлений на рисунку 1.7.



Рисунок 1.7 - Сервопривід SG90

Під час реалізації проєкту в польових умовах використовуються лінійні приводи кінцевими вимикачами, які повідомляють, що більше не потрібно подавати живлення на привід. Лінійний привід (актуатор) - це пристрій, який перетворює обертальний рух двигуна постійного струму на лінійне переміщення.

У теплиці вирішено реалізувати освітлення входу теплиці та її внутрішнє приміщення в темний час доби за рухом, для чого необхідно вибрати датчик руху. Вибір відбувався з трьох моделей модулів, характеристика яких наведена в таблиці 1.6.

Таблиця 1.6. Порівняльні характеристики датчиків руху

Найменування характеристики	E18-D80NK	HC-SR501	HC-SR505
Напруга живлення, В	5	4.5 - 20	4.5 - 20
Кут зони виявлення, градуси	< 15	< 140	< 100
Робоча температура, С	-25...+55	-20...+80	-20...+80
Дальність виявлення, мм	0.03 - 0.8	3-7м (підлаштовується)	3
Розміри плати, мм	45x17	32x24x18	10x23

Для використання в проєкті обрано датчик руху HC-SR501 - піроелектричний інфрачервоний датчик руху, що дає змогу виявляти рух людей у

контрольованій зоні. Являє собою модуль, що складається з ІЧ сенсора 500BP, лінзи Френеля, і керуючої модулем мікросхеми BISS0001. На рисунку 1.8 зображено модуль HC-SR501 [3].

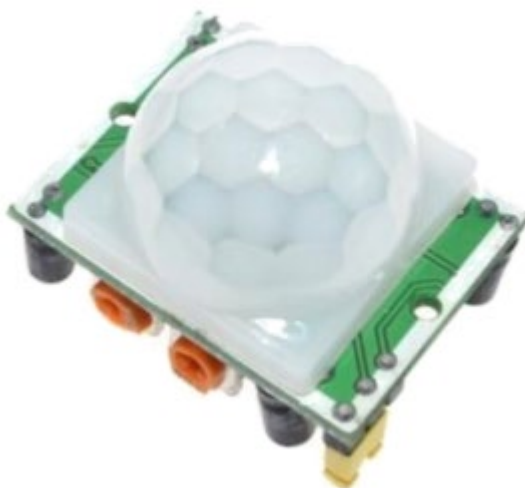


Рисунок 1.8 - Модуль датчика HC-SR501

Для організації поливу також необхідний насос, але оскільки створюється макет теплиці, то і використовувати будемо невеликий за розміром і потужністю AD20P.

Для управління силовим навантаженням необхідний такий пристрій як реле. Реле - комутаційний пристрій, що з'єднує або роз'єднує ланцюг електричної або електронної схеми під час зміни входних величин струму. У таблиці 1.7 наведено порівняльну характеристику реле.

Для комутації силового навантаження обрано електромагнітне реле SRD-05VDC-SL-C.. Реле буде у вигляді модуля для зручності і простоти монтажу та легкої заміни навантаження. Крім самого реле, модуль містить ще й оптоелектронну розв'язку з транзистором, яка захищає виводи мікроконтролера від стрибків напруги на котушці.

Таблиця 1.7 - Порівняльні характеристики реле

Найменування характеристики	NCR HHG1- 1/032F-38-100A	NCR HHG1-1/032F- 38-30A	SRD-05VDC-SL-C
Тип	твердотільний	твердотільний	електромагнітне
Керуюча напруга, В	3...32	3...32	5
Комутоване змінна напруга, В	40...440	40...440	125...250
Максимальний струм навантаження, А	100	30	10

На рисунку 1.9 зображено 4-канальний релейний модуль на основі SRD-05VDC-SL-C.



Рисунок 1.9 - Релейний модуль

До цього модуля зі свого боку можна буде підключати необхідне навантаження: освітлення, обігрівач, насос для поливу та інше [3].

Становлення Інтернету речей значно розширює можливості збору, аналізу та розподілу даних, які для людини перетворюються на інформацію, знання та використовуються для розв'язання специфічних завдань. Для зв'язку між пристроями необхідно обрати за допомогою якого протоколу вони будуть

спілкуватися. Існує безліч протоколів для використання в різних завданнях, наприклад CoAP, MQTT, Modbus.

MQTT (Message Queuing Telemetry Transport) - це асинхронний протокол передавання повідомлень за принципом видавець/передплатник із використанням брокера в мережах із низькою пропускнуою спроможністю. Особливістю протоколу є його компактність, тому що величина заголовка пакета даних становить усього лише кілька байт. Працює поверх TCP/IP, який забезпечує простий і надійний потік даних. Основною метою MQTT є віддалений моніторинг даних, що збираються з великої кількості пристроїв, і їхня телеметрія в IT інфраструктуру. Протокол націлений на велику мережу невеликих пристроїв, які необхідно контролювати або керувати з хмари. Також він призначений для "багатоадресної передачі" даних для багатьох приймачів. MQTT використовується в системах M2M (Машинно-Машинна взаємодія) і IIoT (Промисловий Інтернет речей).

Протокол CoAP часто порівнюють з MQTT для розробки систем IoT. Вони схожі, але є помітні відмінності. MQTT - це протокол "багато до багатьох", тоді як CoAP - це переважно протокол "один до одного" для зв'язку між сервером і клієнтом. Водночас CoAP надає функції метаданих, виявлення та узгодження вмісту, яких немає у MQTT.

Modbus - це відкритий комунікаційний протокол для обміну даними між мережевими пристроями, що ґрунтується на архітектурі ведучий-ведений (master-slave) Перевагами протоколу Modbus є відкритість і масовість використання (безліч виробників випускають різні типи пристроїв, датчиків, що підтримують цей протокол). Завданнями протоколу Modbus є контроль і керування мережами, читання і запис даних у реєстри зберігання, доступ до файлів, діагностика стану пристроїв.

Для реалізації передачі даних між сервером і клієнтом вирішено вибрати протокол MQTT. Протокол зроблений маленьким і легким, з урахуванням суворих умов експлуатації, та ідеальний для пристроїв з невеликою обчислювальною потужністю й обмеженим часом автономної роботи. До їх числа зараз відносяться і смартфони, і постійно зростаюче число датчиків і підключених пристроїв та ін.

Однією з головних особливостей протоколу є те, що ініціаторами підключення виступають клієнти, це може виявитися корисним, коли необхідно збирати дані з різних територіально розподілених джерел. При цьому брокер-сервер має білу ір-адресу, що дає змогу використовувати будь-який хостинг, а клієнти, які підключаються до брокера, мають просто вихід в інтернет.

MQTT має такі особливості:

- асинхронний протокол;
- робота в умовах нестабільного зв'язку на лінії передачі даних;
- ідеально підходить для розподілених комунікацій "один до багатьох" і роз'єднаних додатків;
- нейтральний до вмісту повідомлення;
- покладається на TCP/IP для базових завдань зв'язку;
- оснащений функцією LWT (Last Will and Testament, "остання воля і заповіт") для повідомлення сторін про аномальне відключення клієнта;
- підтримка декількох рівнів якості обслуговування (QoS): "максимум один раз", "мінімум один раз" і "рівно один раз".

Усі ці моменти зробили MQTT протоколом для потокового передавання даних між пристроями з обмеженою потужністю CPU та/або часом автономної роботи, а також для мереж із низькою пропускнуою спроможністю, непередбачуваною стабільністю або високою затримкою. Тому MQTT відомий як ідеальний транспорт для IoT. Він побудований на протоколі TCP/IP, але є відгалуження MQTT-SN для роботи через Bluetooth, UDP, ZigBee і в інших мережах IoT, відмінних від TCP/IP.

У протоколі MQTT обмін повідомленнями здійснюється між клієнтом (client), який може бути видавцем або передплатником (publisher/subscriber) повідомлень, і брокером (broker) повідомлень.

Видавець надсилає дані на MQTT-брокер, вказуючи в повідомленні певну тему, топик (topic). Передплатники можуть отримувати різні дані від безлічі видавців залежно від підписки на відповідні топіки. Учасник системи MQTT може взяти на себе роль видавця, споживача або відразу обидві ролі

Видавець не вимагає будь-яких налаштувань щодо кількості або розташування передплатників, які отримують повідомлення. Крім того, передплатникам не потрібне налаштування на конкретного видавця. У системі може бути кілька брокерів, які поширюють повідомлення.

Пристрої MQTT використовують певні типи повідомлень для взаємодії з брокером, нижче наведено основні:

- connect - встановити з'єднання з брокером;
- disconnect - розірвати з'єднання з брокером;
- publish - опублікувати дані в топик на брокері;
- subscribe - підписатися на топик на брокері;
- unsubscribe - відписатися від топика.

Схема простої взаємодії між передплатником, видавцем і брокером показана на рисунку 1.10.

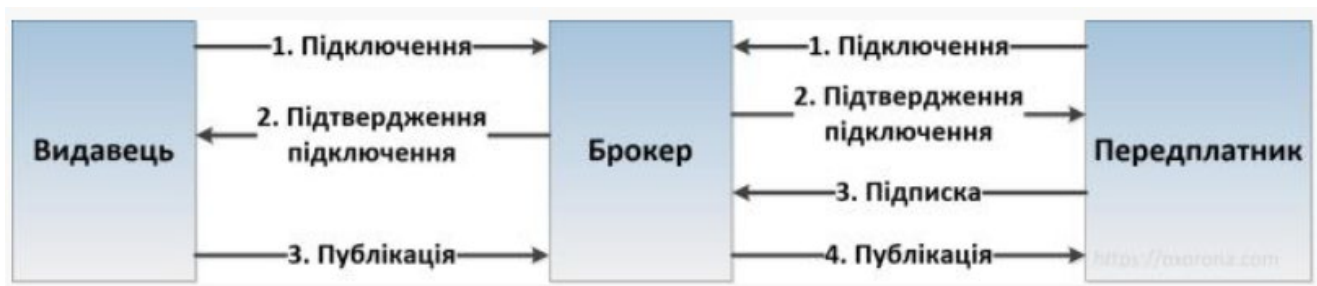


Рисунок 1.10 - Схема взаємодії протоколу mqtt

Топіки являють собою символи з кодуванням UTF-8. Ієрархічна структура топіків має формат "дерева", що спрощує їхню організацію та доступ до даних. Топіки складаються з одного або декількох рівнів, які розділені між собою символом "/". Щоразу, коли у видавця є нові дані для поширення серед клієнтів, повідомлення супроводжується приміткою контролю доставки. Клієнти вищого рівня можуть отримувати кожне повідомлення, тоді як клієнти нижчого рівня можуть отримувати повідомлення, які стосуються тільки одного або двох базових каналів, що "відгалужуються" в нижній частині ієрархії. Це полегшує обмін інформацією розміром від двох байт до 256 мегабайт.

Приклад топика, в який датчик температури, розташований у кухні, публікує дані брокеру: `/home/kitchen/kitchen1/temperature` .

Передплатник може так само отримувати дані одразу з кількох топиків, для цього існують підставні знаки (wildcard). Вони бувають двох типів: однорівневі та багаторівневі.

Однорівневий wildcard, для його використання застосовується символ "+", який ми можемо використовувати замість зазначення імені для будь-якого рівня у фільтрі теми.

Наприклад, нам необхідно отримати дані про температуру в усіх спальних кімнатах:

`/home/room-space+/temperature`

У результаті отримуємо дані з топиків:

`/home/room-space/kitchen1/temperature`

`/home/room -space/kitchen2/temperature`

Багаторівневий wildcard, для його використання застосовується символ "#", який ми можемо використовувати тільки наприкінці фільтра теми як останній рівень, що відповідає будь-якій темі, перші рівні якої збігаються з рівнями тем, зазначеними в лівій частині від спеціального знака "#".

Наприклад, щоб отримати дані з різних датчиків усіх кухонь у будинку:

`/home/room-space/#`

У результаті отримуємо дані з топиків:

`/home/room-space/kitchen1/temperature`

`/home/room -space/kitchen2/light1`

MQTT підтримує три рівні якості обслуговування (QoS - quality of service) під час передавання повідомлень.

QoS 0. На цьому рівні видавець один раз відправляє повідомлення брокеру і не чекає підтвердження від нього, тобто відправив і забув.

QoS 1. Цей рівень гарантує, що повідомлення точно буде доставлено брокеру, але є ймовірність дублювання повідомлень від видавця.

QoS 2. На цьому рівні гарантується доставка повідомлень передплатнику і унеможливлено дублювання відправлених повідомлень.

Для роботи з Raspberry Pi використовуватимемо Raspbian - безкоштовну операційну систему на основі Debian, яка спеціально оптимізована для обладнання Raspberry Pi [5].

Для зручності написання користувацького інтерфейсу вирішено використовувати фреймворк, написаний мовою Python. Python - високорівнева мова програмування загального призначення, орієнтована на підвищення продуктивності розробника і читабельності коду. Синтаксис Python'a мінімалістичний, водночас стандартна бібліотека містить великий обсяг корисних функцій. Фреймворк - це програмне середовище спеціального призначення, своєрідний каркас, який використовується для того, щоб істотно полегшити процес об'єднання певних компонентів під час створення програм. База, на якій можна сформувати веб-сервіс будь-якого призначення досить швидко і без особливих труднощів.

Вибір фреймворку відбувався з трьох варіантів: Flask, Django, TurboGears. TurboGears - open-source full-stack фреймворк для веб-додатків. Він дає змогу розробнику швидко створювати розширювані веб-додатки, керовані даними. TurboGears постачається зі зручними шаблонами та потужним і гнучким ORM. Найважливіші функції фреймворка: підтримка різних баз даних, підтримка SQLAlchemy і SQLObject, валідація для FormEncode, Pylons як веб-сервер, інструменти командного рядка.

Flask - це мікрофреймворк, який надається за ліцензією BSD. Його розробники надихалися фреймворком Sinatra Ruby. Він залежить від інструментарію Werkzeug WSGI та шаблону Jinja2. У Flask модульний дизайн, який дає змогу адаптувати його для виконання багатьох завдань. "З коробки" розробник отримує такі функції: вбудований сервер і дебаггер, шаблони Jinja2, підтримка безпечних кукіс, WSGI 1.0, Unicode, можливість підключення до будь-якої ORM.

Створювали фреймворк для підтримки розробників веб-додатків, які отримали можливість обирати розширення за смаком.

Django - вільний open-source full-stack фреймворк. Він дає змогу додати більшість стандартних функцій єдиним пакетом замість пошуку окремих бібліотек. Серед них - такі затребувані, як аутентифікація, URL-маршрутизація, міграція схеми даних тощо. Django використовує ORM для зіставлення об'єктів із таблицями баз даних. Один і той самий код працює з різними базами даних, тож перехід з однієї БД в іншу стає простим завданням. Основними базами, з якими працює Django, є PostgreSQL, MySQL, SQLite і Oracle. Можна додати й інші, але для цього знадобляться рішення сторонніх розробників.

Вирішено вибрати мікрофреймворк Flask, який найбільше підходить за рахунок своєї модульності та масштабованості. Від самого початку має базовий набір інструментів, що дає змогу під'єднувати і використовувати тільки ті модулі, які необхідні в проєкті.

На цьому етапі завершено вибір апаратних і програмних компонентів проєкту. Обрано мікроконтролер NodeMCU (esp8266), Raspberry Pi 3 як веб-сервер, датчики для отримання даних про температуру і вологість повітря та ґрунту всередині теплиці, і датчик руху.

2 РОЗРОБКА СИСТЕМИ КЕРУВАННЯ

2.1 Розробка та налаштування об'єкта розробки

Головною особливістю мікроконтролерів є контакти GPIO. Інтерфейс вводу/виводу загального призначення (англ. general-purpose input/output, GPIO) - інтерфейс для зв'язку між компонентами системи, наприклад, мікропроцесором і різними периферійними пристроями. Контакти GPIO можуть виступати як у ролі входу, так і в ролі виходу - це, як правило, конфігурується.

GPIO контакти не мають спеціального призначення і, як правило, залишаються невикористаними. Ідея полягає в тому, що іноді для побудови повної системи, що використовує той чи інший чип, може виявитися корисним мати кілька

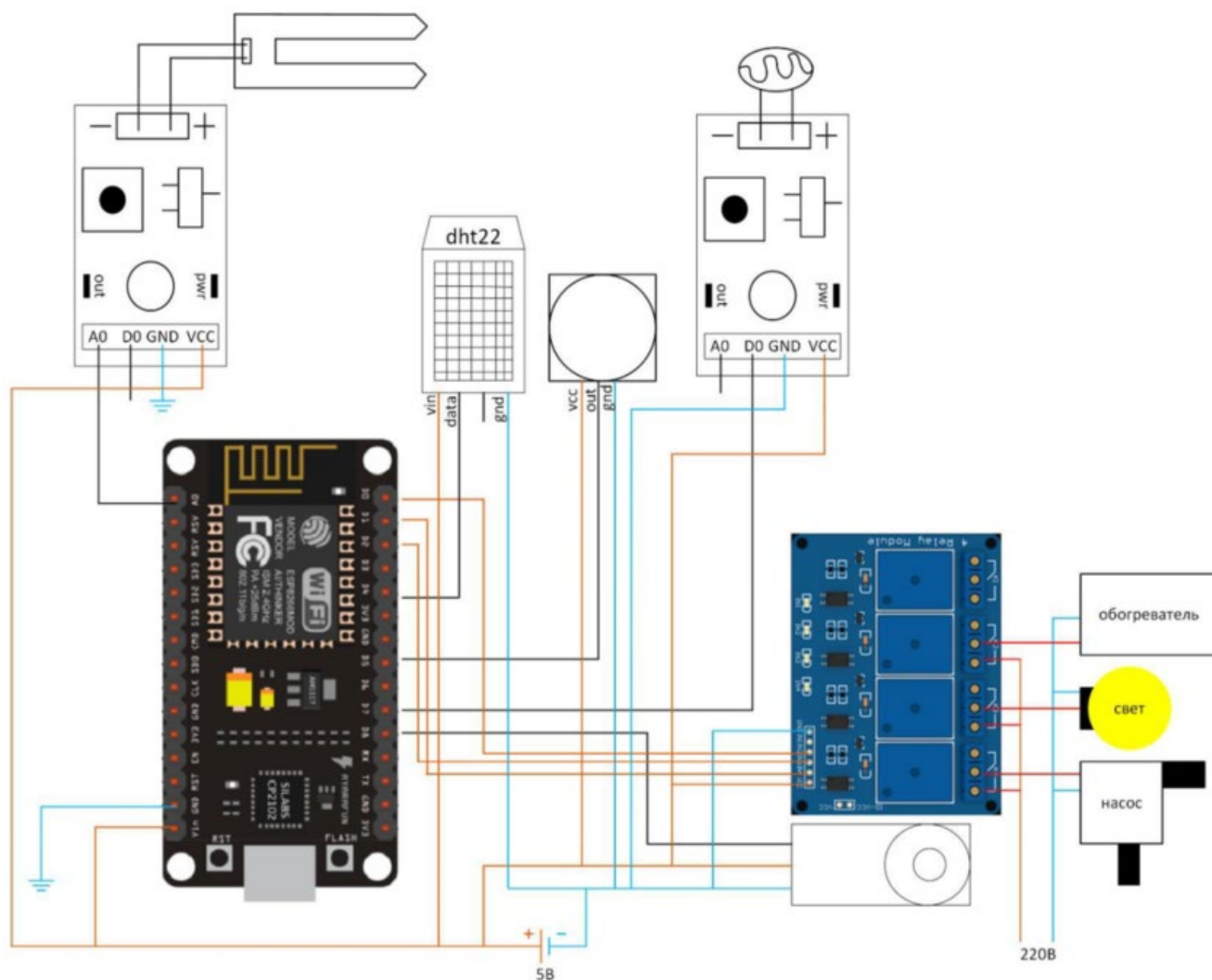


Рисунок 2.2 - Схема з'єднання необхідних компонентів

Зібрана схема поміщена в корпус, який розташовується в теплиці. На рисунках 2.3 та 2.4 - показано макет теплиці.

Якщо у вас є конкретний пристрій чи мікроконтролер, на якому ви хочете використовувати GPIO, будь ласка, надайте додаткові відомості, і я намагатимусь допомогти вам з контактами GPIO на цьому пристрої.

Якщо це мікроконтролер, такий як Arduino, то ви використовуватимете назви пінів, такі як D2, D3, тощо.

Якщо у вас є конкретне запитання щодо використання GPIO на певному пристрої або платформі, будь ласка, подайте більше інформації, і я намагатимусь надати вам конкретні вказівки або кодові приклади.

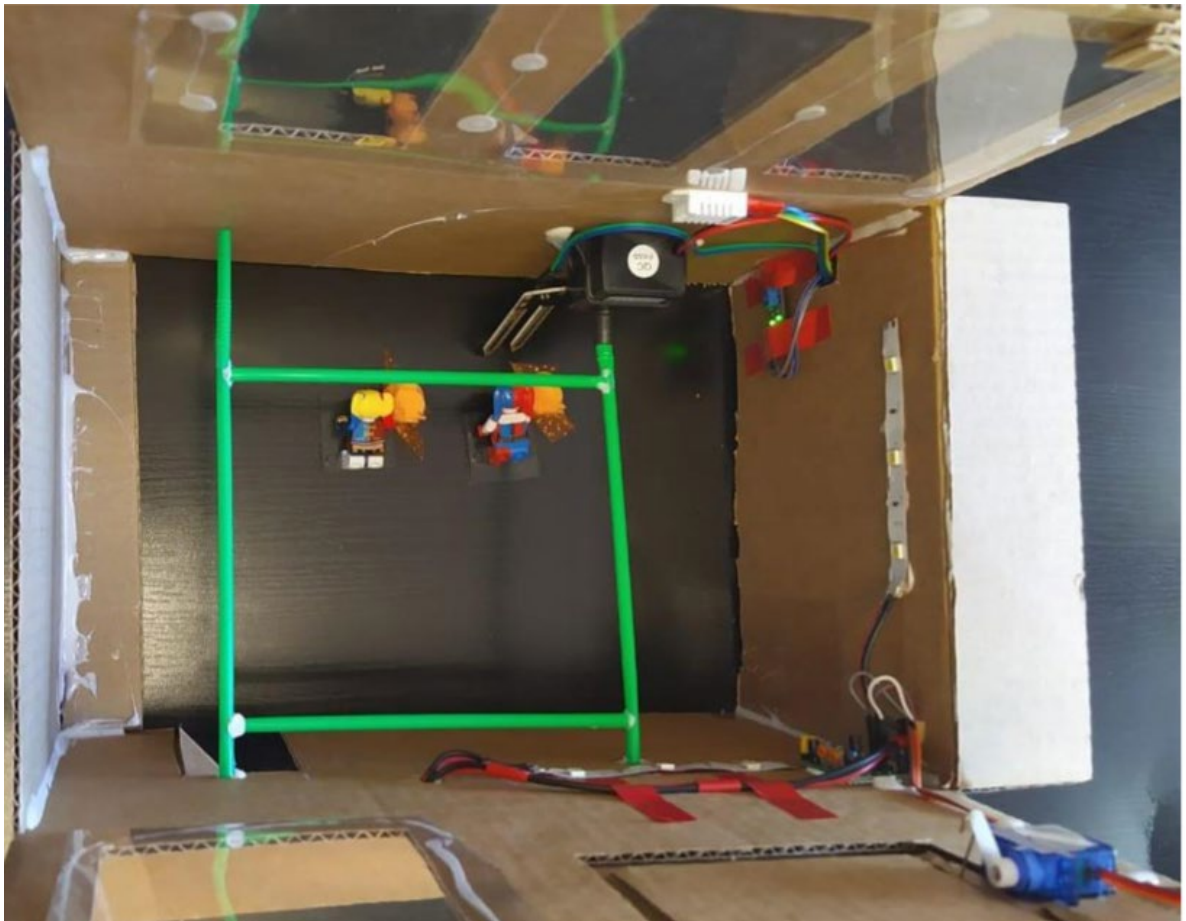


Рисунок 2.3 - Макет, вид зверху

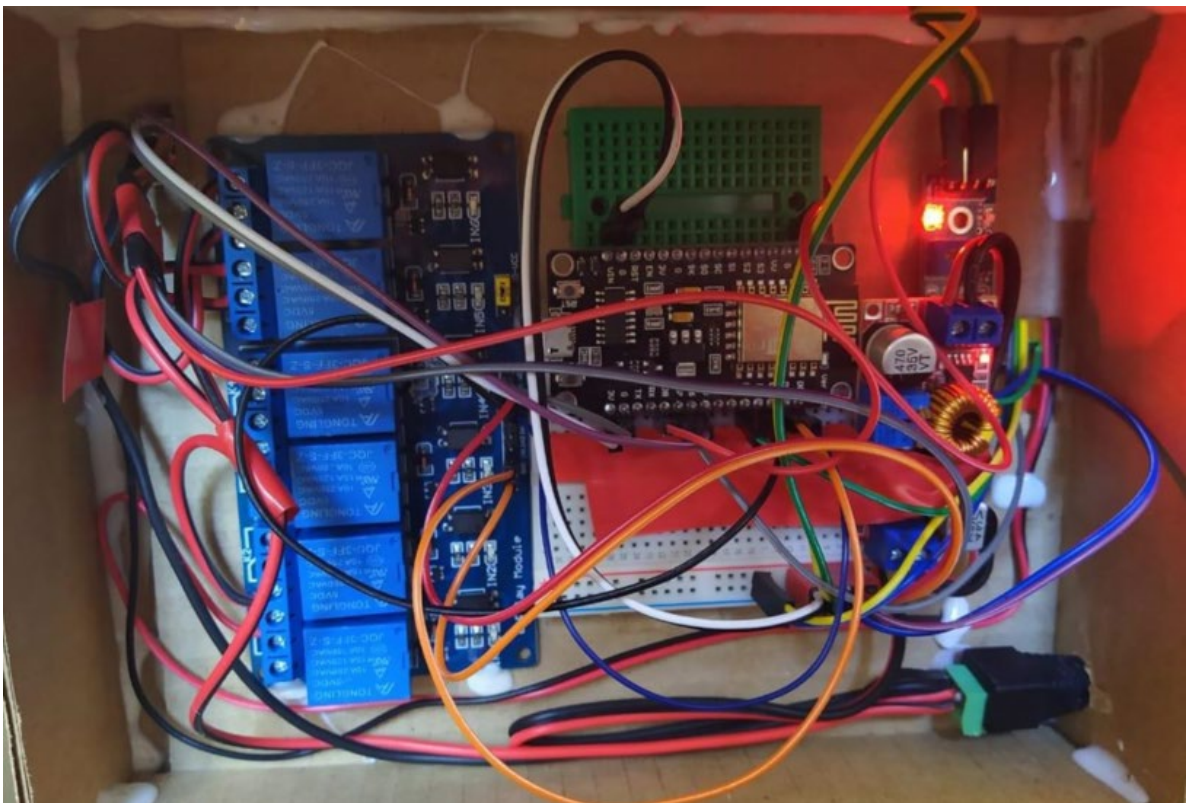


Рисунок 2.4 - Схема всередині макета

Проведено складання макета теплиці. Блок керування поміщено в невелику коробочку, яка встановлюється в теплиці. Розведено дроти для датчиків. У макет встановлено всі необхідні компоненти. Далі необхідно написати код і завантажити його в мікроконтролер.

Плата NodeMCU підтримує кілька мов програмування, такі як C, Lua, MicroPython, JavaScript, Basic, Lisp для роботи з ESP8266. У проєкті використовується C у програмному середовищі Arduino IDE. Воно є зручним інструментом, як для новачків, так і для тих, хто вже має досвід у цій програмі.

Для початку необхідно встановити середовище розробки Arduino IDE, для цього переходимо на сайт arduino.cc, він показаний на рисунку 19, завантажуюмо інсталятор і встановлюємо програму.

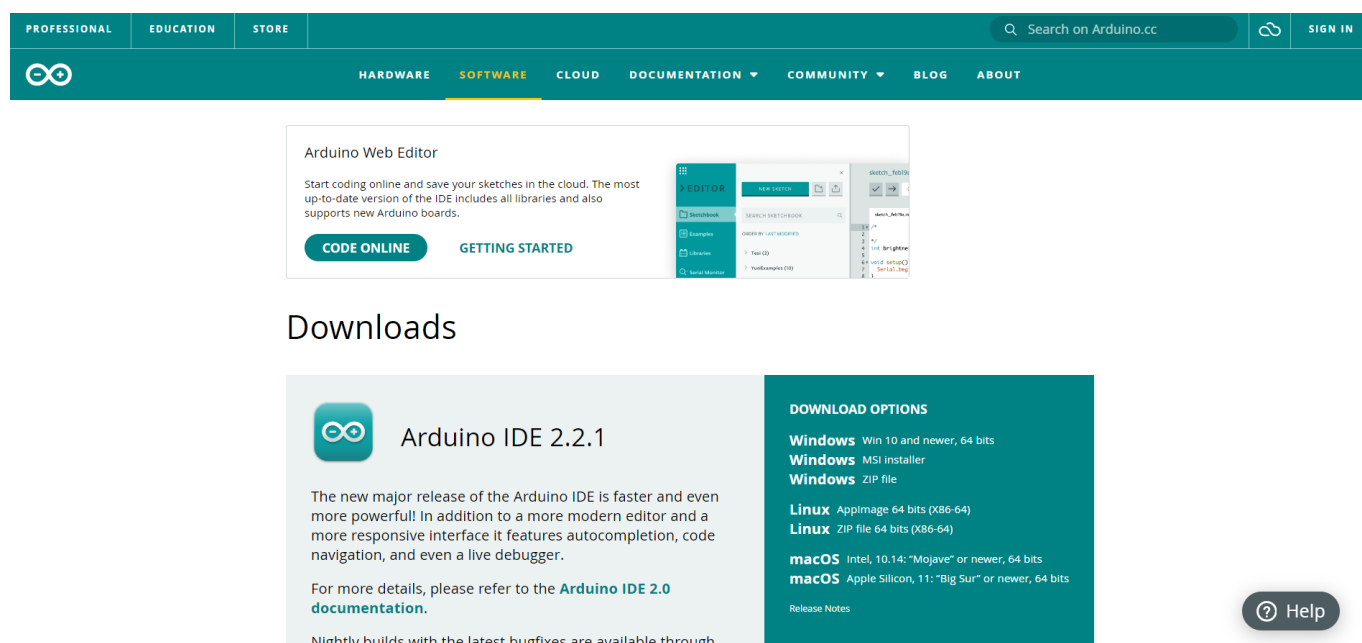


Рисунок 2.5 - Офіційний сайт проєкту Arduino

Запускаємо середовище розробки і переходимо в меню Файл - Налаштування і вставляємо в рядок "Додаткові посилання для Менеджера плат" посилання http://arduino.esp8266.com/stable/package_esp8266com_index.json. Налаштування показано на рисунку 2.6.

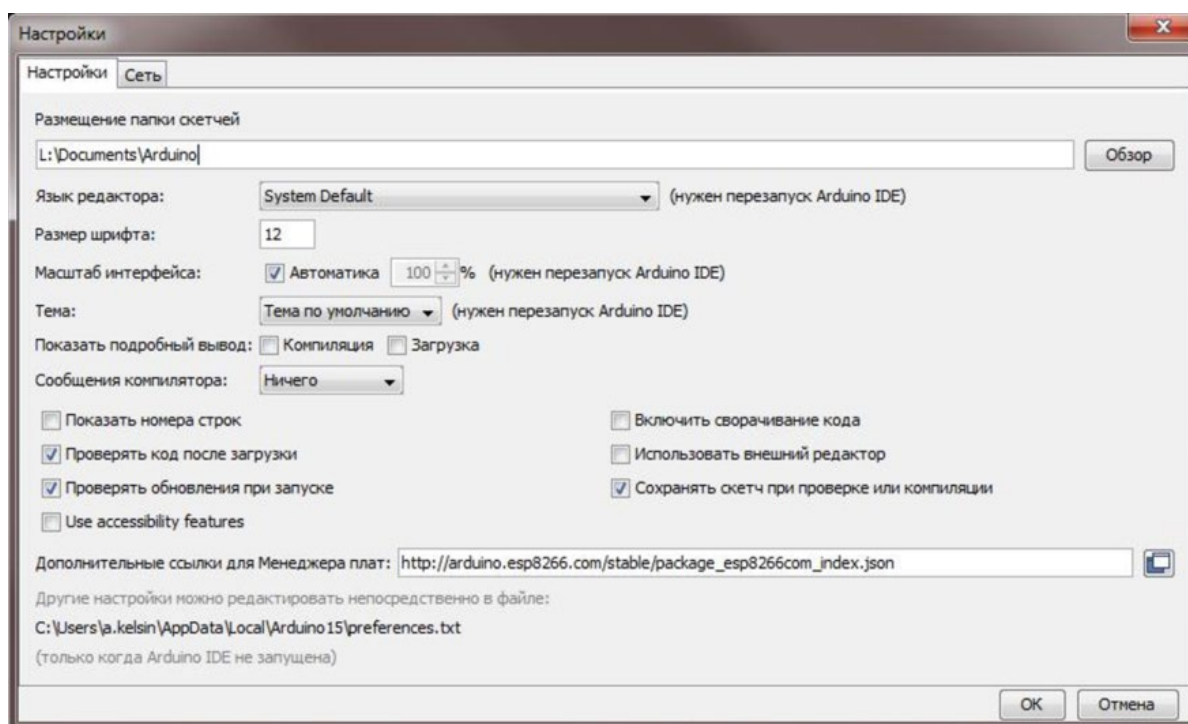


Рисунок 2.6 - Налаштування Arduino IDE

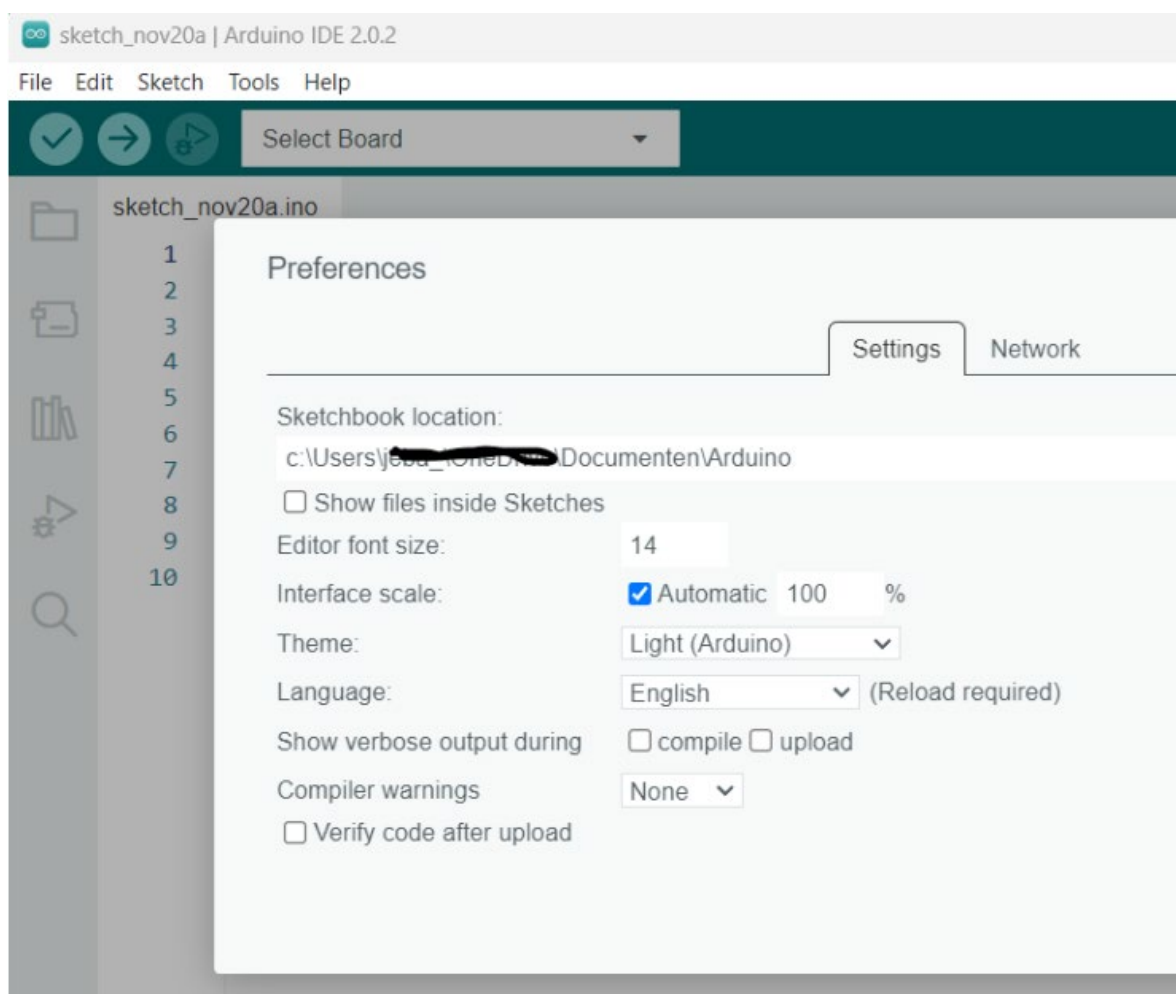


Рисунок 2.7 - Менеджер плат

Далі переходимо в менеджер плат і вводимо в рядок пошуку esp8266, обираємо пункт "esp8266 by ESP8266 Community" версії 2.6.2 і натискаємо "установка". Далі вікно має виглядати так, як показано на рисунку 2.7.

Далі переход в Інструменти і обираємо нашу плату "NodeMCU". Налаштування плати NodeMCU показано на рисунку 2.8.

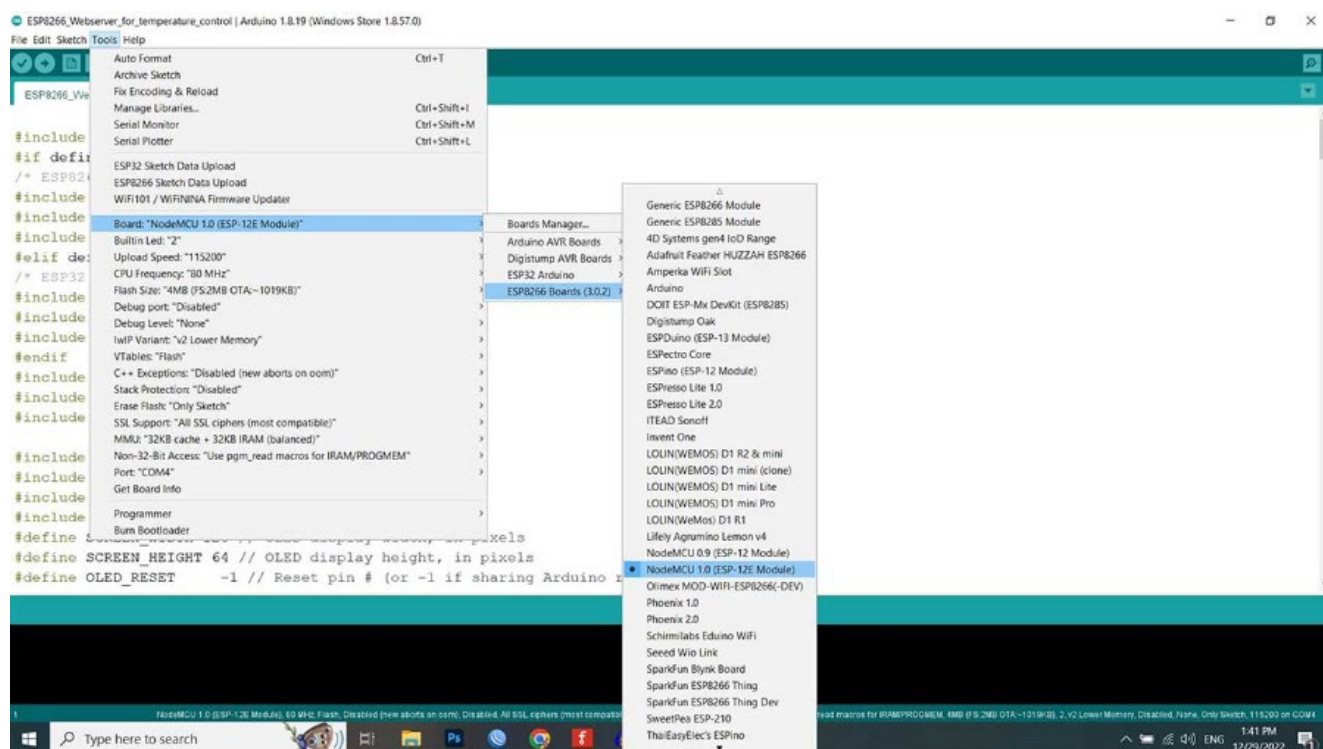


Рисунок 2.8 - Налаштування плати NodeMCU

Далі завантажуюмо необхідні для роботи бібліотеки, заходимо в програму Arduino IDE, далі натискаємо Скетч - Підключити бібліотеку - Додати .ZIP бібліотеку.... і далі вибираємо необхідну бібліотеку, рисунок 2.9. Також це можна зробити вручну, помістивши всі бібліотеки в папку libraries у папці зі встановленою програмою або ж у документах користувача.

Вибір та підключення бібліотек у середовищі Arduino IDE є стандартною процедурою для розробки програмного забезпечення для мікроконтролерів Arduino. Щоб підключити бібліотеку, ви можете виконати кілька кроків.

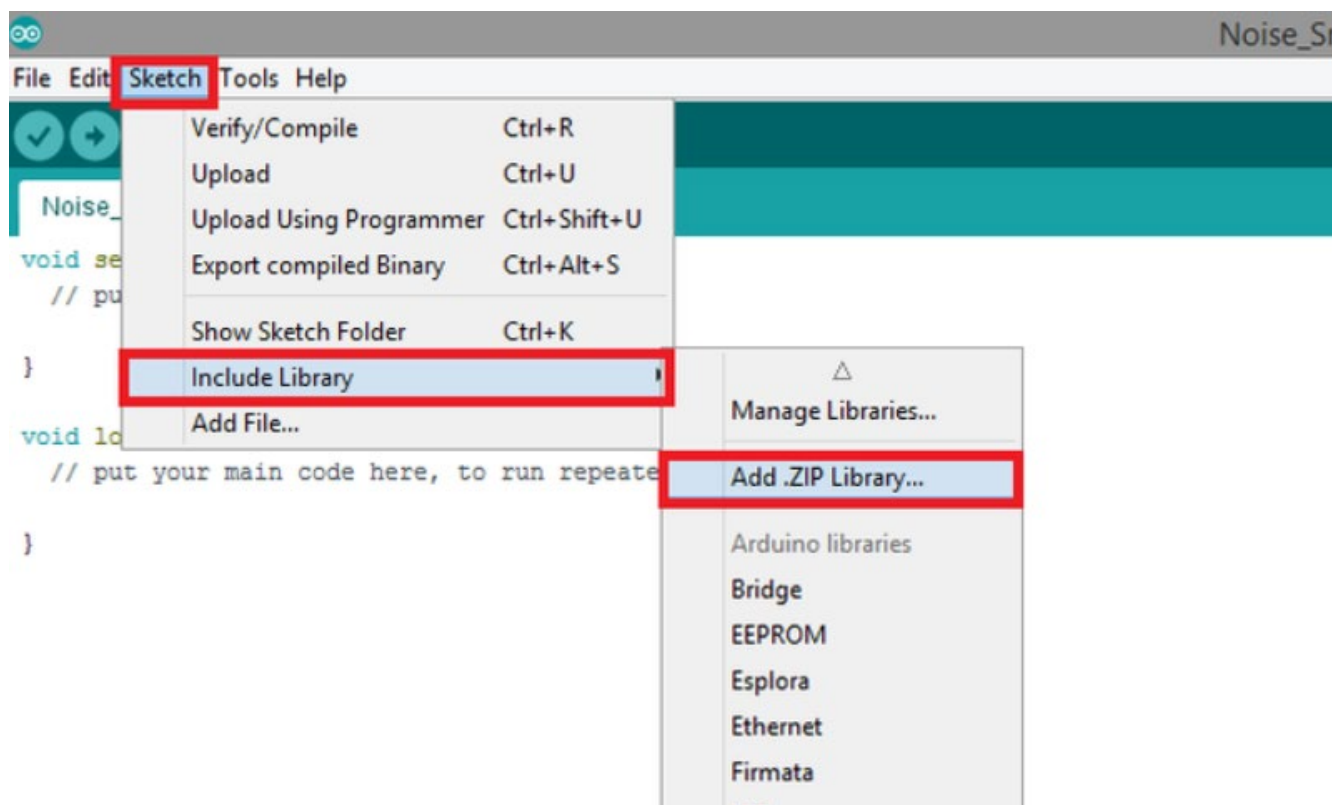


Рисунок 2.9 - Підключення бібліотеки

Для аналізу даних, що надходять з датчиків, і керування навантаженням у залежно від значень цих даних пишемо код, який буде слідувати за алгоритму. Алгоритм дій полягає в тому, що буде вибір ручного й автоматичного режимів.

У ручному режимі користувач сам вирішує, що ввімкнути, а що вимкнути. В автоматичному режимі відбувається визначення дня і ночі, залежно від яких відбувається виконання певного сценарію дій. Також знімаються показники температури, вологості ґрунту та вологості повітря, що, своєю чергою, теж призводить до певних дій (увімкнути обігрівач, відкрити квартиру, полити, увімкнути світло).

Якщо для автоматичної роботи теплиці достатньо лише ввімкнути плату, то для керування в ручному режимі необхідний веб-інтерфейс, за допомогою якого користувач взаємодітиме з теплицею, також це необхідно в разі обриву зв'язку та недоступності основного вузла керування, для під'єднання напряму без використання протоколу MQTT.

На ESP8266 є можливість налаштувати невеликий веб-сервер, який ми можемо використати, щоб організувати зв'язок між користувачем і теплицею, для цього напишемо код, але є кілька нюансів.

Необхідно ввімкнути обігрівач на 15 хвилин, ми можемо використовувати оператор `delay()`, але він затримує виконання програми на вказаний проміжок часу, і поки цей час не сплине, наступні команди програми не виконуватимуться, і доступ до веб-інтерфейсу в цей момент не отримаємо. Через це ми не можемо задати різну тривалість виконання для кожного завдання в циклі `loop()` програми.

Оскільки на мікроконтролері немає можливості паралельного виконання завдань, і всі вони виконуються послідовно один за одним, що не завжди буває зручно. У нашому випадку веб-сервер має бути завжди готовий відповісти на запит даних, і щоб навантаження не божеволіло з нескінченним увімкненням та вимкненням за умови порогових значень, отриманих із датчиків, використовуватимемо псевдопаралельне потокове виконання завдань, що дасть змогу задати проміжок часу, через який виконуватиметься конкретне завдання.

У всіх мікроконтролерах є таймер, включно з NodeMCU. Таймер - це пристрій, який вміє точно відраховувати час. Запускається і обнуляється він щоразу під час подачі живлення на контролер. Щоб дізнатися, скільки часу пройшло на таймері з моменту запуску, використовується функція `millis`. Суть у перевірці значення `millis`, якщо значення більше, ніж час, через який має виконатися функція, то записати новий час у змінну, додати до нього час, через який має виконатися функція, і виконати функцію. Якщо значення `millis` менше заданого періоду, то просто записати новий час у змінну. Але цей спосіб передбачає писати під кожну задачу перевірку, код якої займає значний обсяг, тому будемо використовувати бібліотеку

`Thread.h`, яка дає змогу значно скоротити код, використовуючи псевдопаралельні потоки [7].

У тілі програми створюємо 4 псевдопаралельні потоки для основної функції, що підтримує температуру в приміщенні, для функції керування поливу, функції

перевірки підключення до сервера та функція, що відповідає за зчитування даних з датчиків і їх публікацію.

```
Thread mainnightThread = Thread();
```

```
Thread nasosThread = Thread();
```

```
Thread reconnectThread = Thread();
```

```
Thread datchikiThread = Thread();
```

Призначаємо головному потоку функцію, яку він буде викликати, і задаємо інтервал спрацьовування в мсек. Аналогічно задається для інших потоків.

```
mainnightThread.onRun(mainnight);
```

```
mainnightThread.setInterval(10000);
```

Далі за допомогою рядків нижче підписуємося або перепідписуємося на топіки, можна підписатися не тільки на один, а на кілька одразу. Отримуючи дані з цих топіків від mqtt-брокера, здійснюється управління навантаженням за допомогою централізованого управління.

```
client.subscribe("espmain/auto_main");
```

```
client.subscribe("espmain/svet_main");
```

```
client.subscribe("espmain/nasos_main");
```

```
client.subscribe("espmain/okno_main");
```

```
client.subscribe("espmain/obogrev_main");
```

Щоб опублікувати дані, отримані від датчиків, використовується конструкція, показана нижче.

```
client.publish("/espmain/temperature",
```

```
temperatureTemp); client.publish("/espmain/humidity", humidityTemp);
```

Далі завантажуюємо написаний код у плату NodeMCU. Після закінчення завантаження плата під'єднується до Wi-Fi мережі, яка вказана в коді, і отримує свою ір-адресу, якщо налаштована автоматична видача мережевих реквізитів на роутері, за яким необхідно пройти в будь-якому браузері з будь-якого доступного пристрою і отримати доступ до панелі управління нашої плати NodeMCU. Також на платі можна налаштувати режим "точки доступу", де видачею мережевих

реквізитів розпоряджатиметься вже сама плата, тобто підключення здійснюватиметься без посередників.

Далі наше завдання складається з трьох рівнів. На нижньому рівні знаходяться датчики і контролери. Середина - це місце брокера MQTT, до того ж він має бути розташований локально, щоб проблеми з доступом до інтернету можна було частково вирішити внутрішнім обладнанням. А верхній рівень - хмара з інтерфейсом. У такій концепції дуже підходить енергоефективний і бюджетний мікро-сервер на основі ARM.

У нашому випадку використовується Raspberry Pi, яка поєднуватиме в собі другий і третій рівень, тобто міститиме в собі брокер і застосунок для управління теплицями, збору та зберігання даних.

Згідно з технічним завданням, серверна частина буде розгортатися на фізичному сервері, на базі операційної системи Raspbian. Також додатково буде потрібно встановити і конфігурувати таке програмне забезпечення:

- Apache2.4, Python 2.7 з додатковими бібліотеками, Flask, MariaDB, phpMyAdmin - компоненти веб-сервера;
- Mosquitto - mqtt-брокер повідомлень;
- OpenSSH - сервер віддаленого доступу за протоколом ssh.

На мінікомп'ютері необхідно встановити операційну систему і провести налаштування необхідного програмного забезпечення.

Для встановлення операційної системи на microSD-карту, необхідно завантажити образ Raspbian з графічним інтерфейсом з офіційного сайту. Розпаковуємо архів із файлом, що має розширення img. Вставляємо microSD-карту в комп'ютер. Для завантаження образу необхідно використовувати безкоштовну програму Win32 Disk Imager. Для цього завантажуюмо з офіційного сайту програму, встановлюємо і запускаємо її. Далі в рядку "Image File" вказуємо шлях до образу, в "Device" вибираємо потрібний пристрій і тиснемо "Write". Процес запису образу показано на рисунку 2.10.

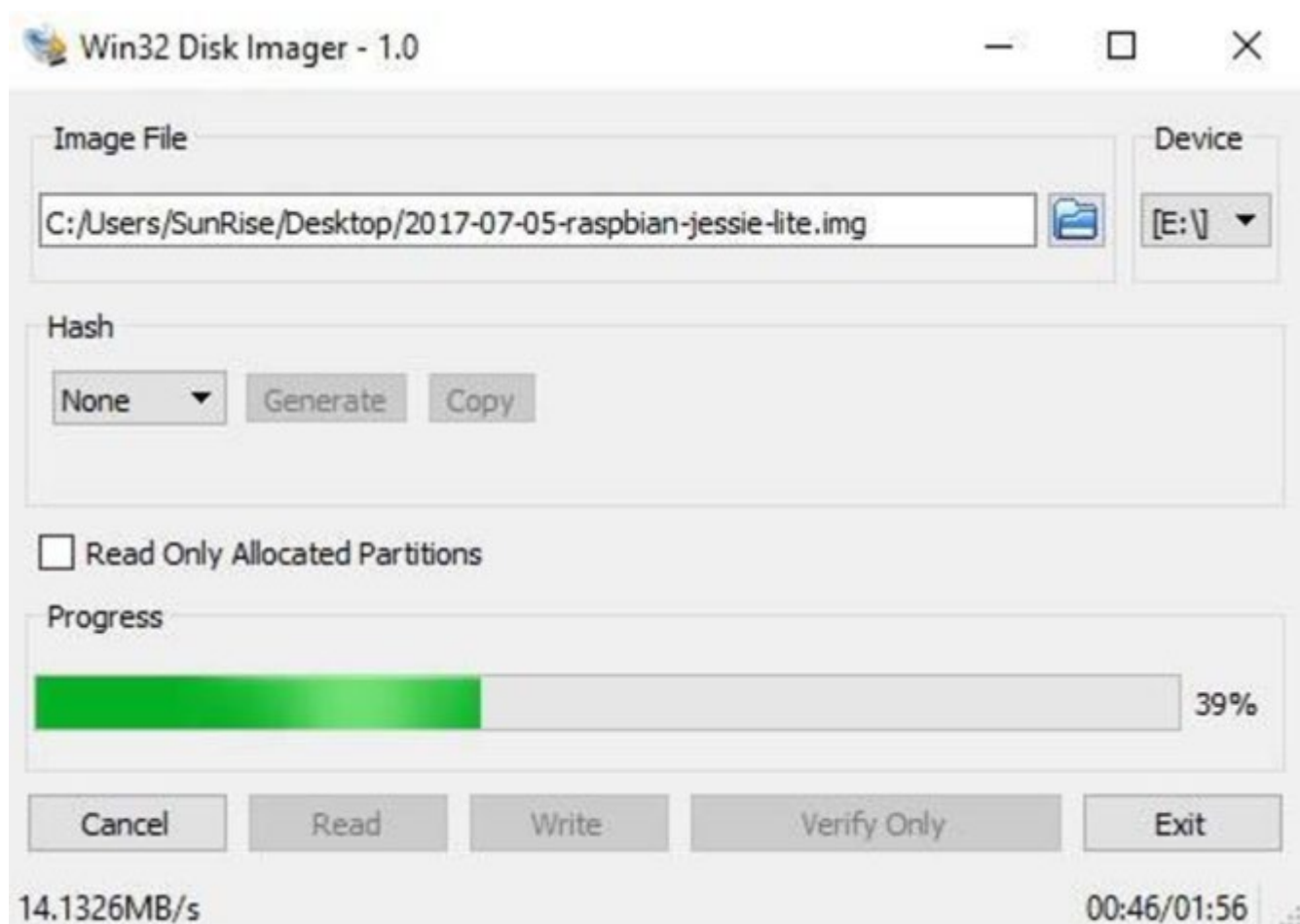


Рисунок 2.10 - Інтерфейс програми

Після успішного запису образу встановлюємо microSD-карту в Raspberry Pi, підключаємо необхідні кабелі [12]. Raspberry Pi вмикається автоматично, щойно на неї подається напруга живлення. На цьому встановлення системи завершено.

За замовчуванням логін - pi і пароль - raspberry. Робочий стіл Raspbian показано на рисунку 2.11.

Після успішного запису образу Raspbian на microSD-карту ви виконуєте наступні кроки для підготовки Raspberry Pi до роботи. Вставте підготовлену карту у гніздо microSD на Raspberry Pi і підключіть необхідні кабелі, такі як кабель живлення та HDMI для підключення до монітора.

Після цього подайте напругу живлення, що запустить Raspberry Pi. Процес завантаження може зайняти деякий час. Як тільки завантаження завершиться, ви побачите робочий стіл операційної системи Raspbian.

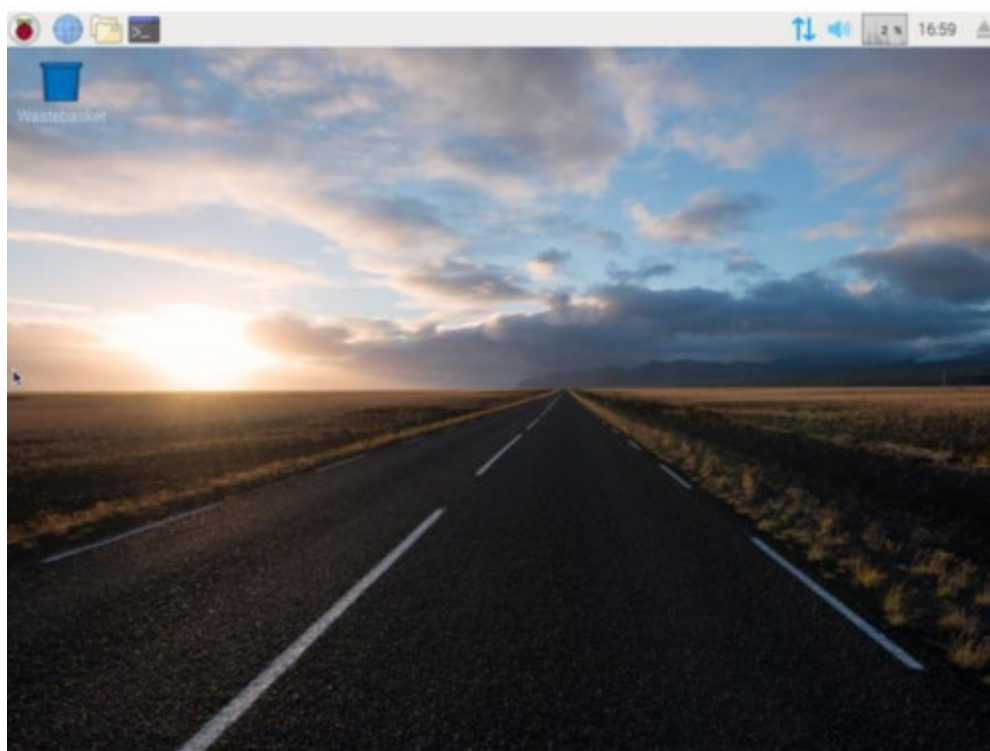


Рисунок 2.11 - Робочий стіл Raspbian

Далі встановлюємо Mosquitto - це брокер повідомлень з відкритим вихідним кодом, що реалізує протоколи MQTT версій 5.0, 3.1.1 і 3.1, рисунок 2.12. Mosquitto легкий і підходить для використання на всіх пристроях: від малопотужних одноплатних комп'ютерів до повноцінних серверів.

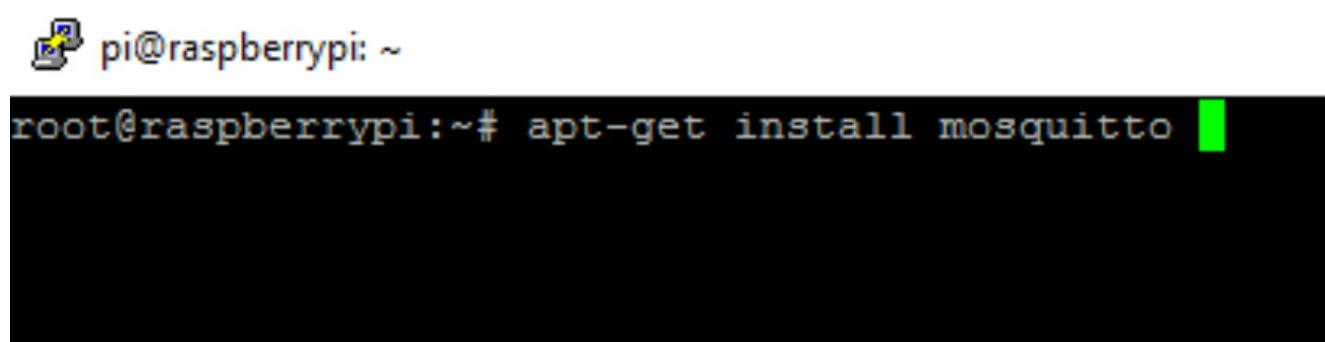
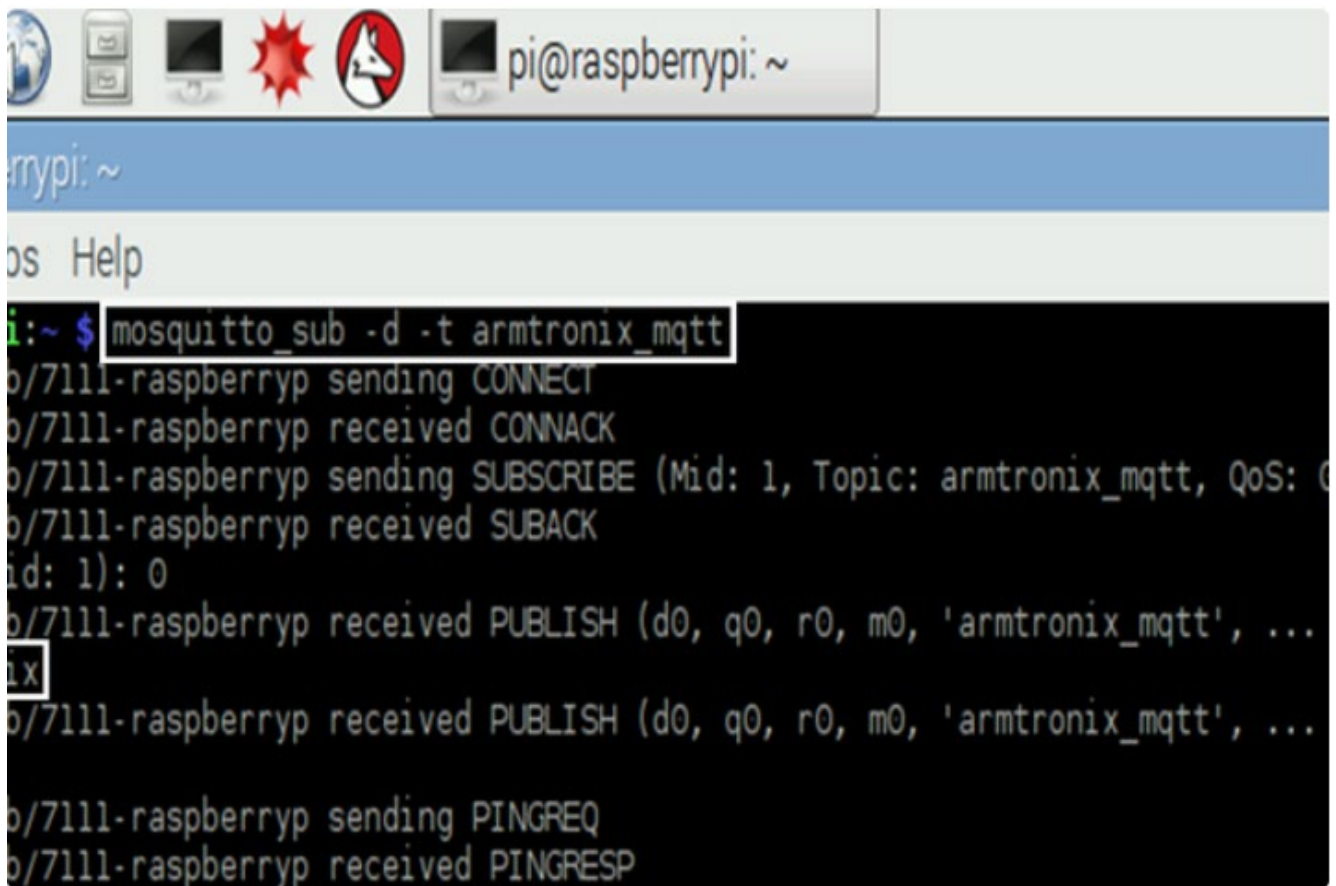


Рисунок 2.12 - Встановлення брокера Mosquitto

Проведемо перевірку роботи брокера, для цього відкриємо два вікна терміналу. У першому вікні підписуємося на топик test, за допомогою команди `mosquitto_sub -d -t test`. У другому вікні публікуємо дані в топик test командою

mosquitto_pub -d -t test -m "hello kels". Дані успішно отримано, що показано нарисунк 2.13.



```

pi@raspberrypi: ~
mosquitto_sub -d -t armtronix_mqtt
b/7111-raspberryp sending CONNECT
b/7111-raspberryp received CONNACK
b/7111-raspberryp sending SUBSCRIBE (Mid: 1, Topic: armtronix_mqtt, QoS: 0)
b/7111-raspberryp received SUBACK (Mid: 1): 0
b/7111-raspberryp received PUBLISH (d0, q0, r0, m0, 'armtronix_mqtt', ...
ix
b/7111-raspberryp received PUBLISH (d0, q0, r0, m0, 'armtronix_mqtt', ...
b/7111-raspberryp sending PINGREQ
b/7111-raspberryp received PINGRESP
  
```

Рисунок 2.13 - Перевірка роботи брокера Mosquitto

Для роботи сайту необхідно встановити обраний нами веб-сервер, СКБД програму для зручного адміністрування баз даних через графічний інтерфейс, а не в терміналі.

Спочатку встановимо всі необхідні компоненти, а потім будемо налаштовувати їх і перевіряти правильність роботи в міру необхідності. Але спочатку потрібно оновити систему до останньої версії:

```
sudo apt-get update && sudo apt-get upgrade
```

Далі встановимо веб-сервер та інтерпретатор мови програмування PHP:

```
sudo apt-get install apache2 php php-mysql libapache2-mod-php php-mbstring php-zip php-gd
```

Проведено встановлення не тільки веб-сервера, а й основних бібліотек PHP, які можуть знадобитися для роботи з phpMyAdmin. Пакет libapache2-php відповідає за інтеграцію підтримки php для веб-сервера Apache.

Наступним встановимо систему управління базами даних - MariaDB. Для її встановлення необхідно виконати команду:

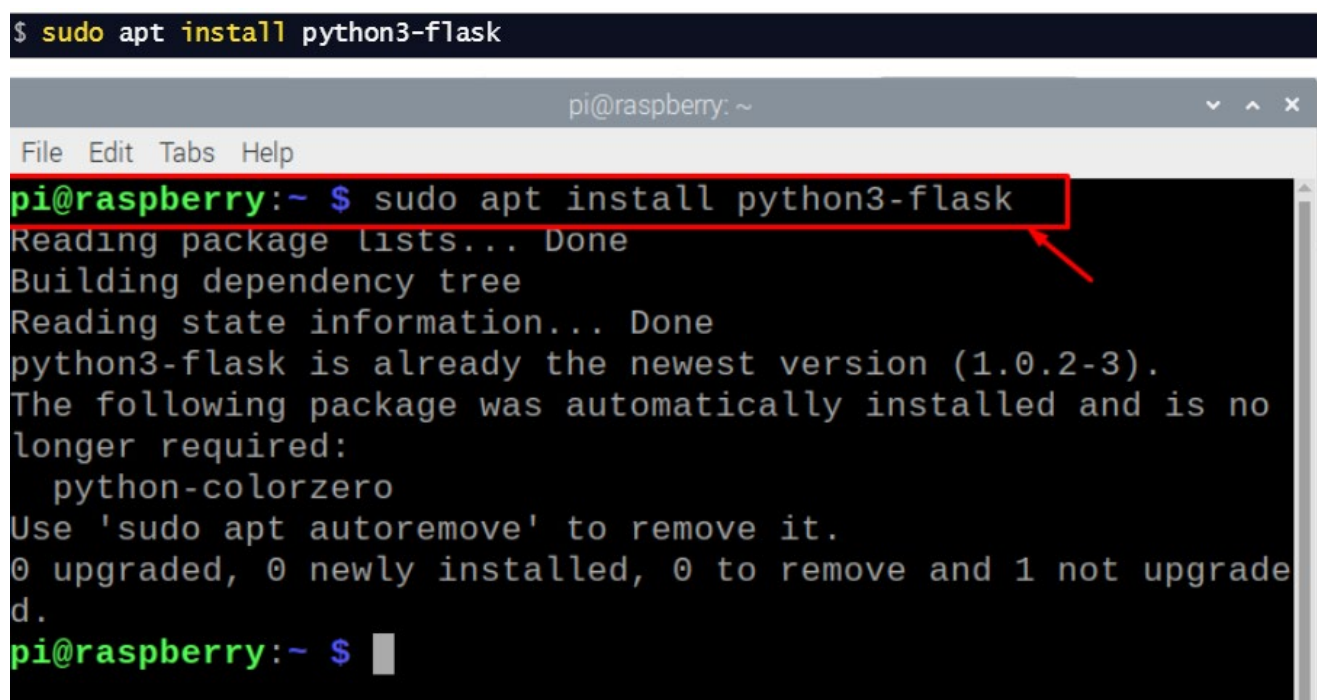
```
sudo apt-get install mariadb-server mariadb-client mariadb-common
```

В якості основної мови програмування, якою буде написано сайт, обрано Python, уже вбудований у Raspbian. Використовуватися буде версія Python 2.7, тому встановлюємо mod_wsgi - модуль для веб-сервера Apache, який надає WSGI-сумісний (англ. Web Server Gateway Interface) інтерфейс для роботи з web-додатками, написаними мовою Python. Сумісний із версіями мови Python 2 і Python 3. Дає змогу запускати web-додатки у двох режимах: режимі вбудовування, за якого використовуються однакові ідентифікатори користувача і групи (uid і gid відповідно) для всіх застосунків, які використовують модуль, а також у режимі демона (доступний для Apache 2.x) - де можна задати користувача, від імені якого запускатиметься кожен застосунок, що підвищує безпеку в разі спільного хостингу для різних користувачів. У нашому випадку використовуємо другий варіант. Встановлення цього модуля показано на рисунку 2.14 [14].

```
--
-- A start job for unit apache2.service has finished successfully.
--
-- The job identifier is 2818.
Mar 02 19:32:20 cormaxon.site sudo[4433]: pam_unix(sudo:session): session closed for user root
Mar 02 19:32:51 cormaxon.site sudo[4541]: pi : TTY=pts/0 ; PWD=/var/www/html/prox ; USER=root ; COMMAND=/bin/nano /etc/apache2/sites-avail
Mar 02 19:32:51 cormaxon.site sudo[4541]: pam_unix(sudo:session): session opened for user root by pi(uid=0)
Mar 02 19:35:33 cormaxon.site rngd[357]: stats: bits received from HRNG source: 120064
Mar 02 19:35:33 cormaxon.site rngd[357]: stats: bits sent to kernel pool: 63584
Mar 02 19:35:33 cormaxon.site rngd[357]: stats: entropy added to kernel pool: 63584
Mar 02 19:35:33 cormaxon.site rngd[357]: stats: FIPS 140-2 successes: 6
Mar 02 19:35:33 cormaxon.site rngd[357]: stats: FIPS 140-2 failures: 0
Mar 02 19:35:33 cormaxon.site rngd[357]: stats: FIPS 140-2(2001-10-10) Monobit: 0
Mar 02 19:35:33 cormaxon.site rngd[357]: stats: FIPS 140-2(2001-10-10) Poker: 0
Mar 02 19:35:33 cormaxon.site rngd[357]: stats: FIPS 140-2(2001-10-10) Runs: 0
Mar 02 19:35:33 cormaxon.site rngd[357]: stats: FIPS 140-2(2001-10-10) Long run: 0
Mar 02 19:35:33 cormaxon.site rngd[357]: stats: FIPS 140-2(2001-10-10) Continuous run: 0
Mar 02 19:35:33 cormaxon.site rngd[357]: stats: HRNG source speed: (min=395.098; avg=682.760; max=958.683)Kibits/s
Mar 02 19:35:33 cormaxon.site rngd[357]: stats: FIPS tests speed: (min=6.588; avg=6.946; max=7.154)Mibits/s
Mar 02 19:35:33 cormaxon.site rngd[357]: stats: Lowest ready-buffers level: 2
Mar 02 19:35:33 cormaxon.site rngd[357]: stats: Entropy starvations: 0
Mar 02 19:35:33 cormaxon.site rngd[357]: stats: Time spent starving for entropy: (min=0; avg=0.000; max=0)us
Mar 02 19:40:25 cormaxon.site sudo[4541]: pam_unix(sudo:session): session closed for user root
Mar 02 19:45:58 cormaxon.site sudo[4565]: pi : TTY=pts/0 ; PWD=/var/www/html/prox ; USER=root ; COMMAND=/bin/nano app.wsgijournalctl -xe
Mar 02 19:45:58 cormaxon.site sudo[4565]: pam_unix(sudo:session): session opened for user root by pi(uid=0)
Mar 02 19:46:02 cormaxon.site sudo[4565]: pam_unix(sudo:session): session closed for user root
lines 3482-3515/3515 (END)
```

Рисунок 2.14 - Встановлення mod_wsgi

Для реалізації веб-інтерфейсу раніше обрано мікрофреймворк Flask, установку якого показано на рисунку 2.15. Використовувалася команда `sudo apt-get install python-pip python-flask git-core`. Також встановлено менеджер пакетів PIP - систему управління пакетами, яка використовується для встановлення та управління програмними пакетами, написаними на Python.



```
$ sudo apt install python3-flask
```

```
pi@raspberrypi:~ $ sudo apt install python3-flask
Reading package lists... Done
Building dependency tree
Reading state information... Done
python3-flask is already the newest version (1.0.2-3).
The following package was automatically installed and is no longer required:
  python-colorzero
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
pi@raspberrypi:~ $
```

Рисунок 2.15 - Встановлення Flask

Встановлено бібліотеку `paho-mqtt`, необхідну для роботи python з протоколом mqtt, з репозиторіїв pip, за допомогою команди `sudo pip install paho-mqtt`.

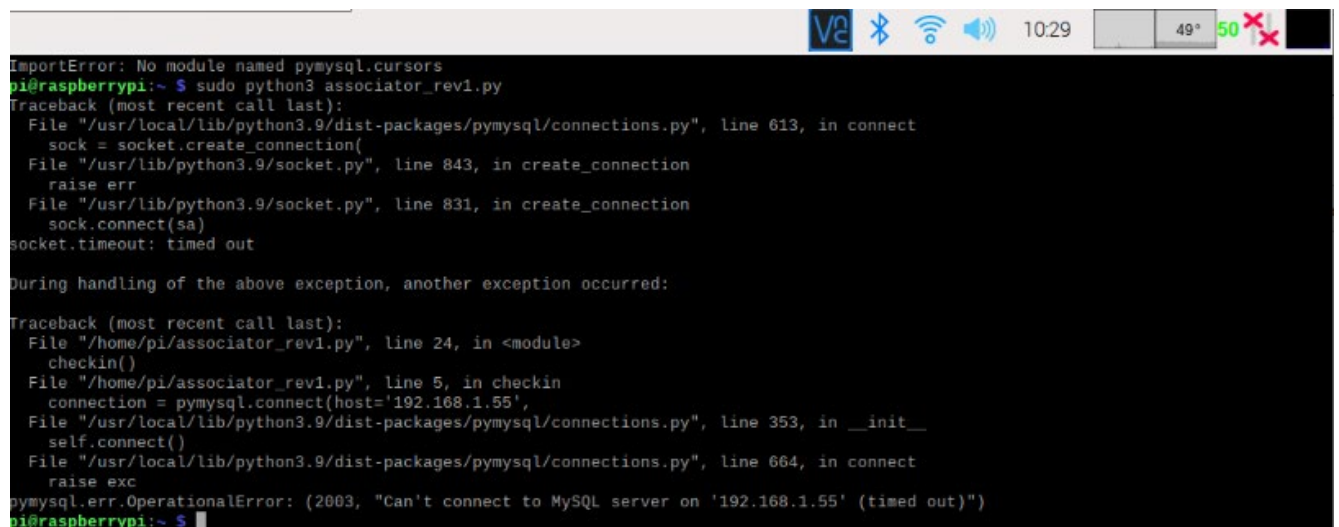
Далі для того, щоб дані про середовище в теплиці та час їхнього отримання оновлювалися на сайті в режимі реального часу, вирішено встановити Socket.IO - JavaScript-бібліотеку для веб-додатків командою `sudo pip install flask-socketio`.

Оскільки для збереження даних про середовище буде використовуватися СУБД, то було вирішено використовувати бібліотеку SQLAlchemy - це бібліотека на мові Python для роботи з реляційними СУБД із застосуванням технології ORM (Object-Relational Mapping), яка пов'язує бази даних з концепціями об'єктно-орієнтованих мов програмування, таких як Python. SQLAlchemy дає змогу

описувати структури баз даних і способи взаємодії з ними прямо мовою Python. Встановлення Flask-SQLAlchemy-2.4.4 і SQLAlchemy версії 1.3.20 показана на рисунку 2.16, також встановлюємо додаткову бібліотеку PyMySQL для роботи SQLAlchemy з СУБД MariaDB, рисунок 2.17.

```
C:\Users\admin\AppData\Local\Programs\Python\Python36\Scripts>pip install flask-sqlalchemy
Collecting flask-sqlalchemy
  Downloading Flask_SQLAlchemy-2.3.2-py2.py3-none-any.whl
Requirement already satisfied: SQLAlchemy>=0.8.0 in c:\users\admin\appdata\local\programs\python\python36\lib\site-packages (from flask-sqlalchemy)
Requirement already satisfied: Flask>=0.10 in c:\users\admin\appdata\local\programs\python\python36\lib\site-packages (from flask-sqlalchemy)
Requirement already satisfied: Jinja2>=2.4 in c:\users\admin\appdata\local\programs\python\python36\lib\site-packages (from Flask>=0.10->flask-sqlalchemy)
Requirement already satisfied: itsdangerous>=0.21 in c:\users\admin\appdata\local\programs\python\python36\lib\site-packages (from Flask>=0.10->flask-sqlalchemy)
Requirement already satisfied: Werkzeug>=0.7 in c:\users\admin\appdata\local\programs\python\python36\lib\site-packages (from Flask>=0.10->flask-sqlalchemy)
Requirement already satisfied: click>=2.0 in c:\users\admin\appdata\local\programs\python\python36\lib\site-packages (from Flask>=0.10->flask-sqlalchemy)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\admin\appdata\local\programs\python\python36\lib\site-packages (from Jinja2>=2.4->Flask>=0.10->flask-sqlalchemy)
Installing collected packages: flask-sqlalchemy
Successfully installed flask-sqlalchemy-2.3.2
```

Рисунок 2.16. Встановлення SQLAlchemy



```
pi@raspberrypi:~ $ sudo python3 associator_rev1.py
ImportError: No module named pymysql.cursors
Traceback (most recent call last):
  File "/usr/local/lib/python3.9/dist-packages/pymysql/connections.py", line 613, in connect
    sock = socket.create_connection(
  File "/usr/lib/python3.9/socket.py", line 843, in create_connection
    raise err
  File "/usr/lib/python3.9/socket.py", line 831, in create_connection
    sock.connect(sa)
socket.timeout: timed out

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/home/pi/associator_rev1.py", line 24, in <module>
    checkin()
  File "/home/pi/associator_rev1.py", line 5, in checkin
    connection = pymysql.connect(host='192.168.1.55',
  File "/usr/local/lib/python3.9/dist-packages/pymysql/connections.py", line 353, in __init__
    self.connect()
  File "/usr/local/lib/python3.9/dist-packages/pymysql/connections.py", line 664, in connect
    raise exc
pymysql.err.OperationalError: (2003, "Can't connect to MySQL server on '192.168.1.55' (timed out)")
pi@raspberrypi:~ $
```

Рисунок 2.17. Встановлення PyMySQL

Далі створюємо папку, де буде розміщено айт командою:

```
sudo mkdir /var/www/teplica.org/
```

Створюємо користувача flask, без домашнього каталогу, і додаємо в групу, пов'язану з веб-сервером, www-data за допомогою команди:

```
sudo nano useradd flask -M -g www-data -p password
```

Задаємо віртуальну доменну адресу, за якою ми будемо отримувати доступ до нашого сайту. Конфігураційні файли віртуальних хостів зберігаються в каталозі `/etc/apache2/sites-available/` створюємо нову конфігурацію `teplica.conf` командою:

```
sudo nano /etc/apache2/sites-available/teplica.conf
```

Тепер в отриманому файлі прописуємо директиви, необхідні для роботи сайту.

Задаємо на якому порту працюватиме сайт.

```
<VirtualHost *:80>
```

Задаємо ім'я віртуального домену, за яким буде доступний сайт. `ServerName` `teplica.org`

Прописуємо шлях до папки з сайтом.

```
DocumentRoot /var/www/teplica.org
```

Можна задати будь-яку електронну адресу для відображення в помилках.

```
ServerAdmin root@localhost
```

Далі задаємо від якого користувача, якої групи буде запускатися процес `р` додатком, а також кількість потоків і домашній каталог.

```
WSGIDaemonProcessreplica.orguser=flaskgroup=www-datathreads
```

home=/var/www/teplica.org threads - визначає кількість потоків для опрацювання запитів по кожному з процесів демона всередині групи процесів, визначених директивою `WSGIProcessGroup`.

Задається місце розташування скрипта, що обробляє запити до додатка. `WSGIScriptAlias` `/var/www/teplica.org/teplica.py`

Далі задаємо параметри для директорії кореня сайту.

```
<Directory /var/www/teplica.org>
```

Групуємо процеси демона:

```
WSGIProcessGroup replica.org
```

Кожен сайт віртуального хоста Flask має використовувати окрему групу процесів демона, тому група додатків `%{GLOBAL}` змушує використовувати контекст основного інтерпретатора у відповідних групах процесів.

```
WSGIApplicationGroup ${GLOBAL}
```

Щоб полегшити роботу інструментів встановлення, вмикаємо підтримку автоматичного перезавантаження, коли wsgi-скрипт зміниться, mod_wsgi перезавантажить для нас усі процеси демона.

WSGIScriptReloading On

Задаємо дозвіл на доступ до сайту.

Розпорядження дозволити, відмовити

Дозволити від усіх

</Directory>

Задаємо шлях до каталогу static.

Alias /static /var/www/teplica.org/teplica/static

Задаємо дозволи каталогу static.

<Directory /var/www/teplica.org/teplica/static>

Розпорядження дозволити, відмовити

Дозволити від усіх

</Directory>

Далі прописуємо, куди мають записуватися помилки та системні повідомлення.

ErrorLog \${APACHE_LOG_DIR}/error.log

LogLevel warn

CustomLog \${APACHE_LOG_DIR}/access.log

combined </VirtualHost>

На рисунку 2.18 показано конфігураційний файл сайту replica.org.

Автоматичне перезавантаження. Для полегшення розробки і випробувань зазвичай встановлюється підтримка автоматичного перезавантаження. Це означає, що якщо ваш wsgi-скрипт або додаток змінився, сервер автоматично перезапустить процеси демона, щоб відобразити внесені зміни.

Наприклад, ви можете використовувати параметр reload у конфігурації mod_wsgi:

```

GNU nano 2.7.4 Файл: /etc/apache2/sites-available/teplica.conf
VirtualHost *:80>
ServerName teplica.org
DocumentRoot /var/www/teplica.org/
ServerAdmin mr.keisin@mail.ru

WSGIDaemonProcess teplica.org user=flask group=www-data threads=5 home=/var/www/teplica.org
WSGIScriptAlias / /var/www/teplica.org/teplica.py

<Directory /var/www/teplica.org>
WSGIProcessGroup teplica.org
WSGIApplicationGroup %{GLOBAL}
WSGIScriptReloading On
Order allow,deny
Allow from all
#Require all granted
</Directory>

Alias /static /var/www/teplica.org/teplica/static

<Directory /var/www/teplica.org/teplica/static/>
Order allow,deny
Allow from all
#Require all granted
</Directory>

ErrorLog ${APACHE_LOG_DIR}/error.log
LogLevel warn
CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>

[ Read 80 lines ]
G Help      ^O Write Out ^W Where Is  ^K Cut      ^T Execute  ^G Location
X Exit      ^R Read File ^\ Replace   ^U Paste    ^J Justify  ^_ Go To Line

```

Рисунок 2.18 -. Конфігураційний файл

Потім створюємо WSGI-скрипт у кореневій папці сайту, за допомогою команди. *sudo nano /var/www/teplica.org/teplica.py*

У створеному файлі прописуємо параметри, зазначені нижче.

Вказуємо, що використовується python.

```
#!/usr/bin/python
```

Імпортуємо необхідні бібліотеки.

```
import os, sys, flask
```

```
import logging
```

Вказуємо, що для запису логів буде використовуватися основний потік. *logging.basicConfig(stream=sys.stderr)*

Вказуємо, що насамперед необхідно імпортувати папку, в якій будуть розташовані основні файли програми.

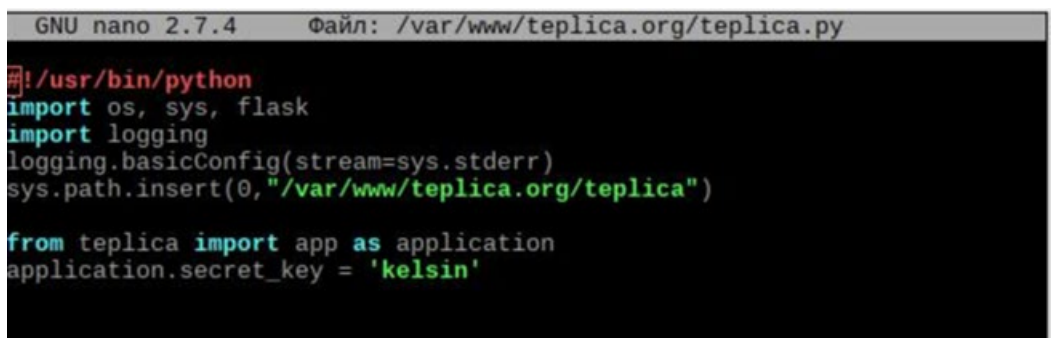
```
sys.path.insert(0, "/var/www/teplica.org/teplica")
```

Далі вказуємо, що необхідно імпортувати наш сайт як додаток. *from teplica import app as application*

Задаємо для додатка секретний ключ.

`application.secret_key = "kelsin"`

На рисунку 2.19 показано wsgi-скрипт сайту teplica.org.



```
GNU nano 2.7.4 Файл: /var/www/teplica.org/teplica.py
#!/usr/bin/python
import os, sys, flask
import logging
logging.basicConfig(stream=sys.stderr)
sys.path.insert(0, "/var/www/teplica.org/teplica")

from teplica import app as application
application.secret_key = 'kelsin'
```

Рисунок 2.19 - Файл teplica.py

Після виконаних дій, створюємо в корені сайту каталог teplica, в якому зі свого боку поміщаємо каталоги templates, де створюємо порожні файли html-сторінок, і static, усередині якого поміщаємо статичні файли, такі як картинки і стилі сторінок, а також файл `__init__.py`, в якому буде прописана логіка програми. На рисунку 2.20 показано отримане дерево каталогів і файлів.



```
pi@rpi:~ $ tree /var/www/teplica.org
/var/www/teplica.org
├── teplica
│   ├── __init__.py
│   ├── static
│   │   ├── 00bd.png
│   │   ├── logo.gif
│   │   ├── pic1.gif
│   │   └── style.css
│   └── templates
│       ├── base.html
│       ├── info.html
│       ├── main.html
│       ├── panel.html
│       ├── stat.html
│       └── tepl.html
└── teplica.py

3 directories, 12 files
pi@rpi:~ $
```

Рисунок 2.20 - Дерево каталогів і файлів

Далі необхідно прописати код у файлі `__init__.py`, який завантажується найпершим і містить логіку, за якою додаток працюватиме.

У файлі підключаємо всі потрібні бібліотеки. Створюємо підкласи, що звертаються до об'єктів класів для управління даними в таблицях БД, за допомогою SQLAlchemy, тому писати запит на чистому sql не потрібно, що спрощує написання і читабельність коду [34].

```
class Temper(db.Model):
```

Вказуємо з якою таблицею працювати.

```
__tablename__ = 'temperatura'
```

Вказуємо поля в таблиці.

```
id_temp = db.Column(db.Integer(), primary_key=True)
```

```
board = db.Column(db.Text, nullable=False)
```

```
date = db.Column(db.DateTime(), default=datetime.utcnow)
```

```
temp = db.Column(db.Float, nullable=False)
```

__repr__ ж викликається при операторі print, для зручності відображення.

```
def __repr__(self):
```

```
    return "<{}: {}: {}>".format(self.id_temp, self.date, self.board,  
self.temp)
```

2.2 Вимоги до моніторингу

Вибір методів і об'єктів моніторингу залежить від багатьох чинників – конфігурації мережі, які у ній сервісів і служб, конфігурації серверів і встановленого ними ПЗ, можливостей ПЗ, що використовується для моніторингу і т.п.

- перевірка фізичної доступності устаткування;
- перевірка стану занедбаних служб і сервісів;
- детальна перевірка не критичних, але важливих параметрів функціонування: продуктивності, завантаження тощо;

- перевірка параметрів, специфічних сервісів і служб даного конкретного оточення (наявність деяких значень у таблицях БД, вміст лог-файлів).

Проектом, над яким здійснюватиме моніторинг, було висунуто такі конкретизуючі уточнення до загального переліку елементів моніторингу.

У першому пункту можна віднести перевірку доступності мережі хоста, віртуальної машини. Тут достатньо тестування наявності з'єднання та швидкості пригоди сигналу.

До другого пункту в конкретному проекті належить доступність IIS-сервісів, стан призначених завдань та служб.

До третього пункту відносяться перевірка довжини дискової черги, стану оперативної пам'яті, синхронізації серверних систем за часом, заповненості «кошика» тощо.

У четвертому, найважливішому пункту відносяться перевірки доступності сховища Apache Cassandra, наявності або відсутності в ній певного кіспейсу та значення показника його фактора реплікації. Для систем Windows також перевіряється, чи встановлено важливий хот-фікс на машині з сервісами, значення оперативної пам'яті, яку займає служба установки модулів і оновлень. Також необхідно було реалізувати очищення логів у Logstash-ElasticSearch комбінації за певними умовами.

Далі написана функція, яка приймає ім'я мікроконтролера і температуру та завантажує їх у базу.

```
def insert_temp(board, temperature):
```

Використано конструкцію try-ехсепт, яка намагається виконати код, і в разі невдачі, виконує код після ехсепт.

```
спробуй:
```

Задаємо змінну з простим запитом, оскільки використовується

```
SQLAlchemy.slqinsert = Temper(board=board, temp=temperature)
```

Далі запит додається в сесію роботи з базами даних. SQLAlchemy автоматично виконує дії із сесією

```
db.session.add(slqinsert)
```

Застосовуємо внесені зміни до баз даних.

```
db.session.commit()
```

Задаємо змінну і виводимо, що все добре.

```
message = "OK mysql"
```

```
друк повідомлення
```

Задаємо змінну і виводимо, якщо станеться помилка.

за винятком:

```
message = "Проблеми з mysql у temp"
```

```
друк повідомлення
```

створеною функцією `on_connect` підписуємося на топіки, дані з яких надходитимуть із мікроконтролерів ESP8266. Далі функція `on_message`

приймає дані з топіків, на які підписаний клієнт і відповідає за їхню обробку.

Наступним задали параметри для роботи MQTT клієнта і задаємо, що раніше створені функції відповідають за обробку даних.

Сучасні веб-додатки використовують "красиві" URL, за допомогою маршрутизації, тому в додатку використовується декоратор `route()`, який необхідний для прив'язки функції до URL. Це допомагає людям запам'ятати ці URL, щоб користувач міг перейти відразу на бажану сторінку, без попереднього відвідування початкової сторінки. Задаємо маршрутизацію для головної сторінки.

Усередині Flask уже автоматично попередньо налаштований шаблонізатор Jinja2. Тому для візуалізації шаблону використовується метод `render_template()`. Усе, що необхідно - це вказати ім'я шаблону, а також змінні у вигляді іменованих аргументів, які необхідно передати рушію обробки шаблонів.

```
@app.route("/")
```

```
def main():
```

```
    return render_template('main.html')
```

Задаємо маршрутизацію для сторінки з даними про температуру і вологість конкретної теплиці. Щоб додавати до адреси URL змінні частини, можна ці особливі частини виділити в `<variable_name>`. Потім подібні частини передаються у функцію як аргумент - у вигляді ключового слова.

```
@app.route("/stat/<board>", methods=['post','get'])
```

Оскільки на сторінці з даними про температуру і вологість реалізовано форму, за допомогою якої можна задавати параметри виведення даних, то додаємо перевірку, чи надходили дані з форми, чи ні. Параметри можна задавати по одному, завдяки реалізованій перевірці.

```
if request.method == 'POST':
    if request.form.get('rows') is None:
        rows = 100
    іще:
```

Зводимо в модуль і округляємо кількість рядків.

```
rows = abs(int(request.form.get('rows')))
calendar = request.form.get('calendar')
```

Далі виконуємо два запити відповідно до заданих параметрів із форми.

```
sqltemp =
db.session.query(Temper).order_by(db.desc(Temper.date)).filter(Temper.board
==board,
    Temper.date.startswith(calendar)).limit(rows).all()
sqlhum =
db.session.query(Humidity).order_by(db.desc(Humidity.date)).filter(Humidity.b
oard==bo
ard, Humidity.date.startswith(calendar)).limit(rows).all()
```

Задаємо маршрутизацію для сторінки з панеллю керування навантаженням теплиць, при натисканні на кнопку.

```
@app.route("/<board>/<changePin>/<action>")
def action(board, changePin, action):
```

Переводимо змінну в рядковий тип даних.

```
changePin = str(changePin)
```

Присвоюємо ім'я плати зі словника відповідно до прийнятого changePin.

```
devicePin = pins[changePin]['name']
```

Далі проводимо перевірку, на якій платі і в який стан перевести навантаження.

якщо `action == "1" i board == 'espmain'` або `action == "1" i board == 'espone'`:

Публікуємо дані в потрібний топик.

```
mqttc.publish(pins[changePin]['topic'],
"1")
```

Змінюємо значення про стан у словнику `pins`.

```
pins[changePin]['state'] = 'True'
```

якщо `action == "0" i board == 'espmain'` або `action == "0" i board == 'espone'`:

```
mqttc.publish(pins[changePin]['topic'], "0")
```

```
pins[changePin]['state'] = 'False'
```

```
templateData = {
    'pins': pins
}
```

```
return render_template('panel.html', async_mode=socketio.async_mode,
**templateData)
```

Повний лістинг файлу `__init__.py` з коментарями подано в додатку 4. Далі створюємо файли шаблонів сайту. Файл `base.html` - це основний

2.3 Загальні відомості про побудову моніторингу

Враховуючи всі вищеописані вимоги до моніторингу, було вирішено використовувати систему, яка могла б працювати з будь-якими операційними системами, могла б виконувати скриптові сценарії на більшості поширених мов і не вимагала встановлення агентського програмного забезпечення на кожную машину, за станом якої здійснюватиметься стеження.

Було помічено, що система `TeamCity`, яка вже використовується в інфраструктурі проекту (за допомогою неї виконувалося розгортання проекту на

бойові сервери), повністю відповідає необхідним умовам, а також не вимагає вкладення додаткових коштів та встановлення пакетів сервера та агента.

У загальному випадку моніторинг у TeamCity є окремим проектом (проектами) з однією або декількома конфігураціями всередині: розбиття на конфігурації краще здійснювати залежно від необхідності сповіщень тих чи інших користувачів про певні проблеми, тобто. оповіщення спрацьовує за будь-якої помилки всередині конфігурації, яка призводить до невдалої побудови. Також можна створювати окремий проект, а додати конфігурації (конфігурацію) до існуючого проекту розробки, на серверах якого реалізується моніторинг.

Перше зручно у разі великої компанії, коли розробкою та ремонтом поломок сервера займаються різні люди, другий випадок можна використовувати в маленькій компанії (у невеликому проекті), коли всі маніпуляції здійснюються силами розробників. Вид проекту представлений рисунку 2.1.

Шаблон сайту, який за допомогою мікрофреймворку flask поєднуватиметься з іншими сторінками сайту і віддаватиметься користувачеві як повноцінна сторінка. Параметри `{% block content %}{% endblock %}` вказують, що в цьому місці міститиметься вміст іншої html-сторінки. В інших же файлах міститься така структура:

```
{% extends 'base.html' %}
{% block content %}
    html код
{% endblock content %},
```

що вказує на те, що необхідно додати вміст файлу `base.html`. Цей спосіб дуже ефективний під час редагування, а також зменшує кількість файлів і рядків коду. Повний лістинг основних файлів сайту можна побачити в додатку 6 [6].

Далі зберігаємо зміни й активуємо сайт командою:

```
sudo a2ensite teplica.conf
```

Тепер залишилося перезапустити Apache за допомогою команди:

```
sudo service apache2 restart
```

як СУБД використовуємо MariaDB. Щоб почати її налаштування виконуємо у

командному рядку:

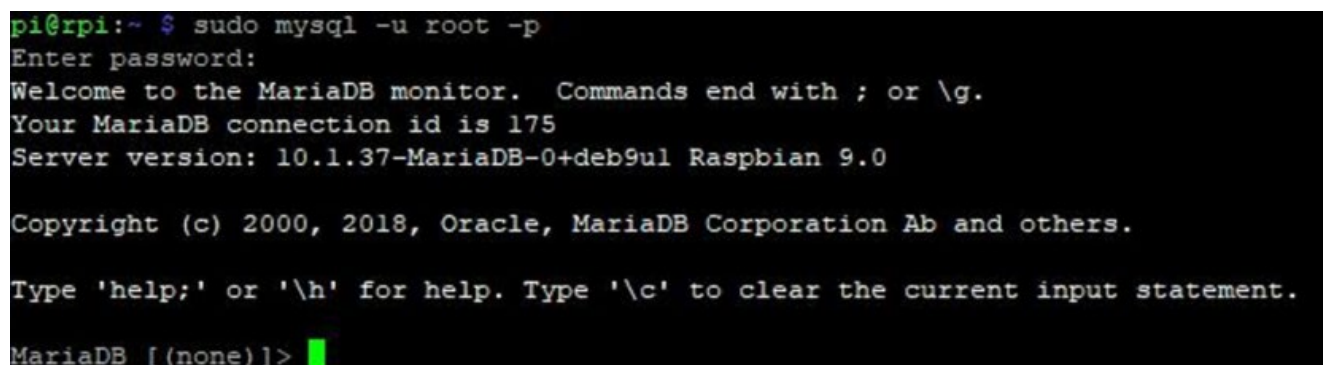
mysql_secure_installation

Далі буде запропоновано задати новий пароль для СУБД, потім видалити анонімних користувачів, видалити тестову базу даних, перезавантажити всі права на доступ до бази даних - погоджуємося з усім.

Після того як процес початкового налаштування завершиться, перевіряємо працездатність СУБД із раніше створеним паролем:

sudo mysql -u root -p

На рисунку 2.21 показано успішний результат входу.



```
pi@rpi:~ $ sudo mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 175
Server version: 10.1.37-MariaDB-0+deb9u1 Raspbian 9.0

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

Рисунок 2.21 - Результат входу

Для зручності адміністрування бази даних зробимо установку phpMyAdmin - це вільне програмне забезпечення, призначене для роботи з MySQL (MariaDB) у мережі. Ця програма дає змогу використовувати зручний зовнішній інтерфейс для MySQL (MariaDB). Можна завантажити з сайту розробника пакет цього програмного забезпечення, але оскільки цей пакет додано в репозиторій нашої операційної системи, скористаємося командою:

sudo apt-get install phpmyadmin

Під час встановлення phpMyAdmin необхідно виконати базову конфігурацію. Коли процес встановлення запущено, виконуємо такі дії:

- як сервер вибираємо Apache2;
- на запитання, чи потрібно налаштувати базу даних для phpmyadmin за допомогою dbconfig-common, відповідаємо YES;

- вводим пароль, раніше створений під час налаштування MariaDB;
- встановлюємо пароль для входу в phpmyadmin.

Після завершення встановлення вносимо phpMyAdmin у конфігурацію Apache:

```
sudo nano /etc/apache2/apache2.conf
```

Допишуємо в конфігураційний файл phpMyAdmin:

```
Включити /etc/phpmyadmin/apache.conf
```

Потім перезапускаємо Apache:

```
sudo service apache2 restart
```

Тепер відкриваємо phpMyAdmin, вводим в рядок адреси браузера `terpica.org/phpmyadmin/index.php`, це виведе сторінку привітання phpMyAdmin, яка має такий вигляд, як показано на рисунку 2.22.

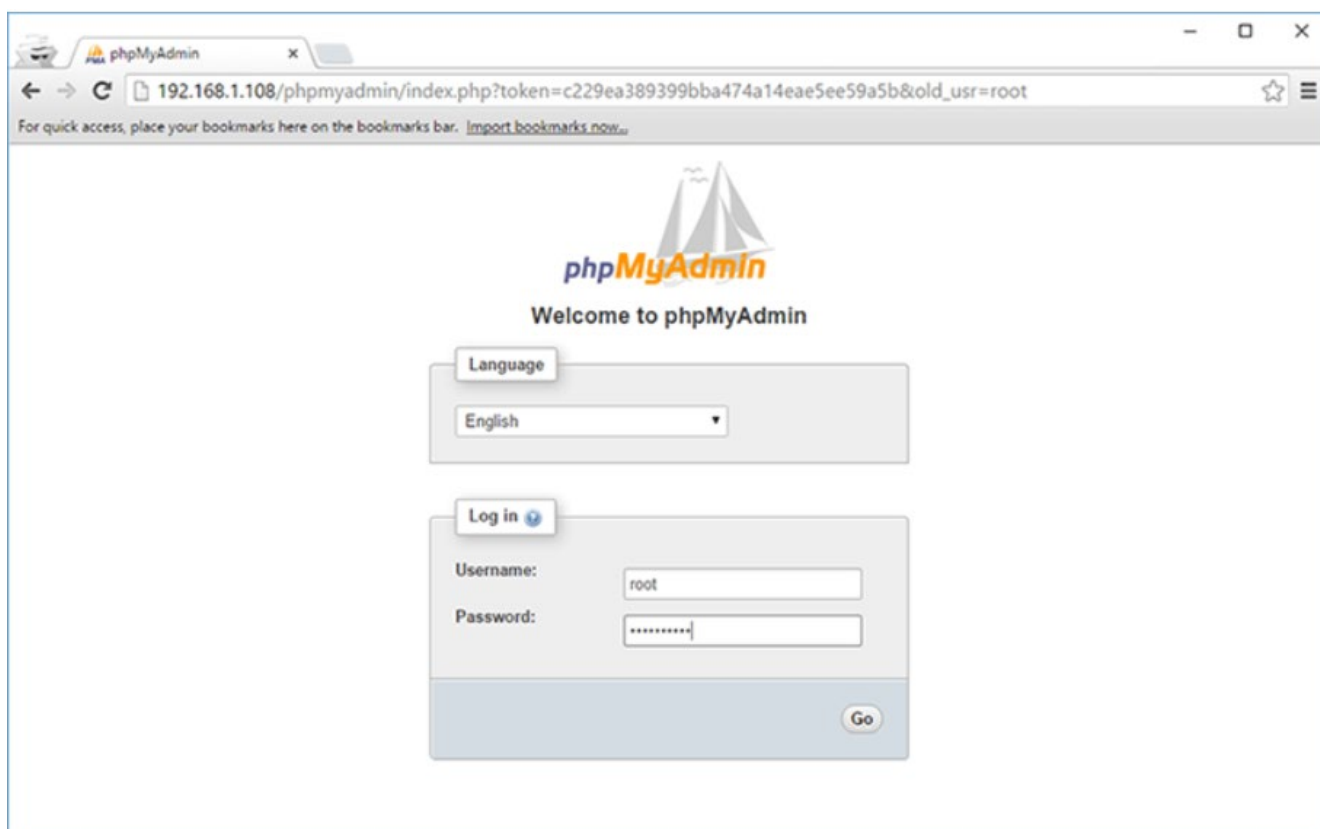


Рисунок 2.22 - phpMyAdmin

Вводим логін root і пароль, який задавали раніше, і потрапляємо в інтерфейс програми, у лівій частині натискаємо на "Створити БД", вказуємо ім'я

нашої бази даних "terpica" і натискаємо "Створити". Цей момент показано на рисунку 2.23.

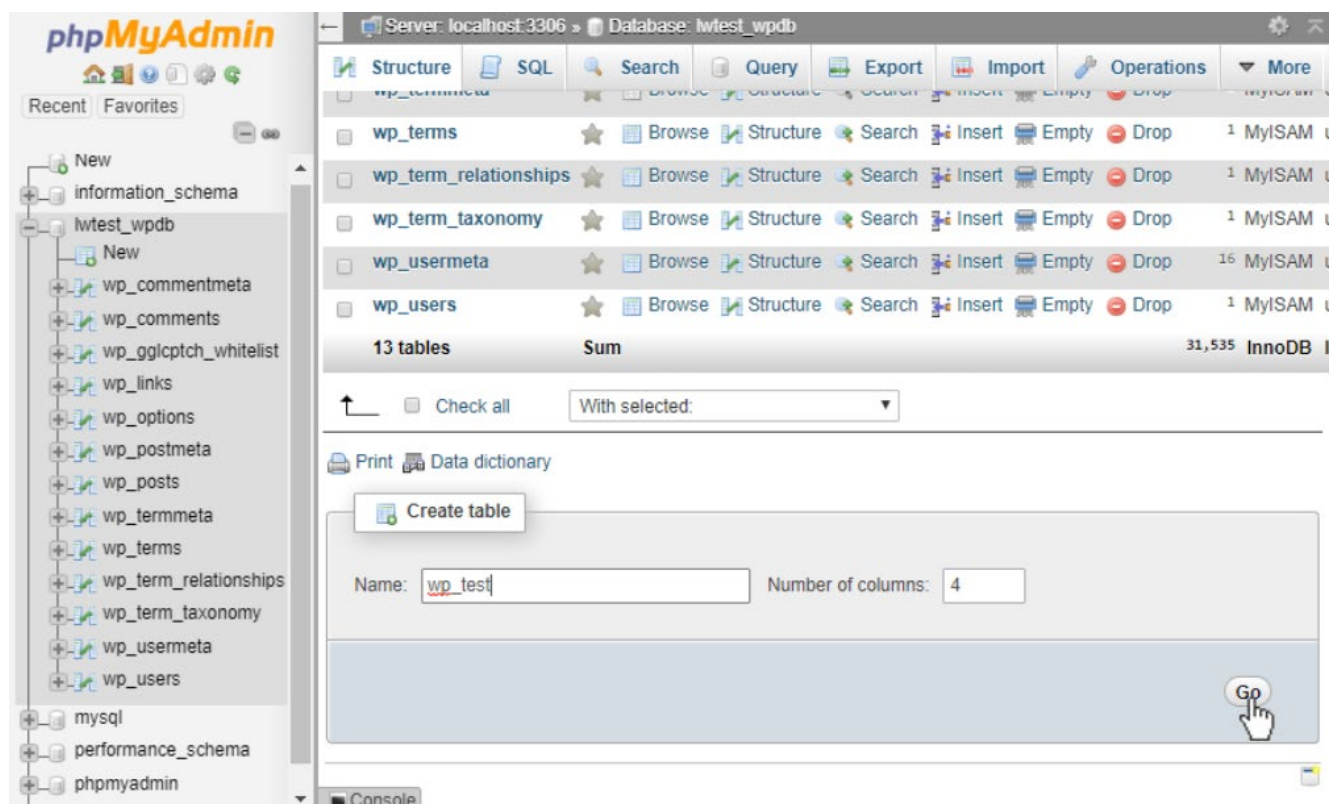


Рисунок 2.23 - Створення БД

Далі необхідно створити таблицю для даних, що надходять із датчика вологості. Натискаємо в розділі бази даних terpica "Нова таблиця" і задаємо назву і тип полів. У поля "id_temp" ставимо галочку параметра A_I, що означає автоматичне додавання значення на 1, у полі "board" записуватиметься ім'я плати, з якої надходять дані, тому воно має текстовий тип даних. Поле "date" має тип дати і час, також у значенні за замовчуванням задано параметр для отримання поточного часу отримання даних.

Останнє поле має тип дійсного числа, куди буде записана температура в одиничний проміжок часу. На рисунку 38 показано поля таблиці для збереження даних температури.

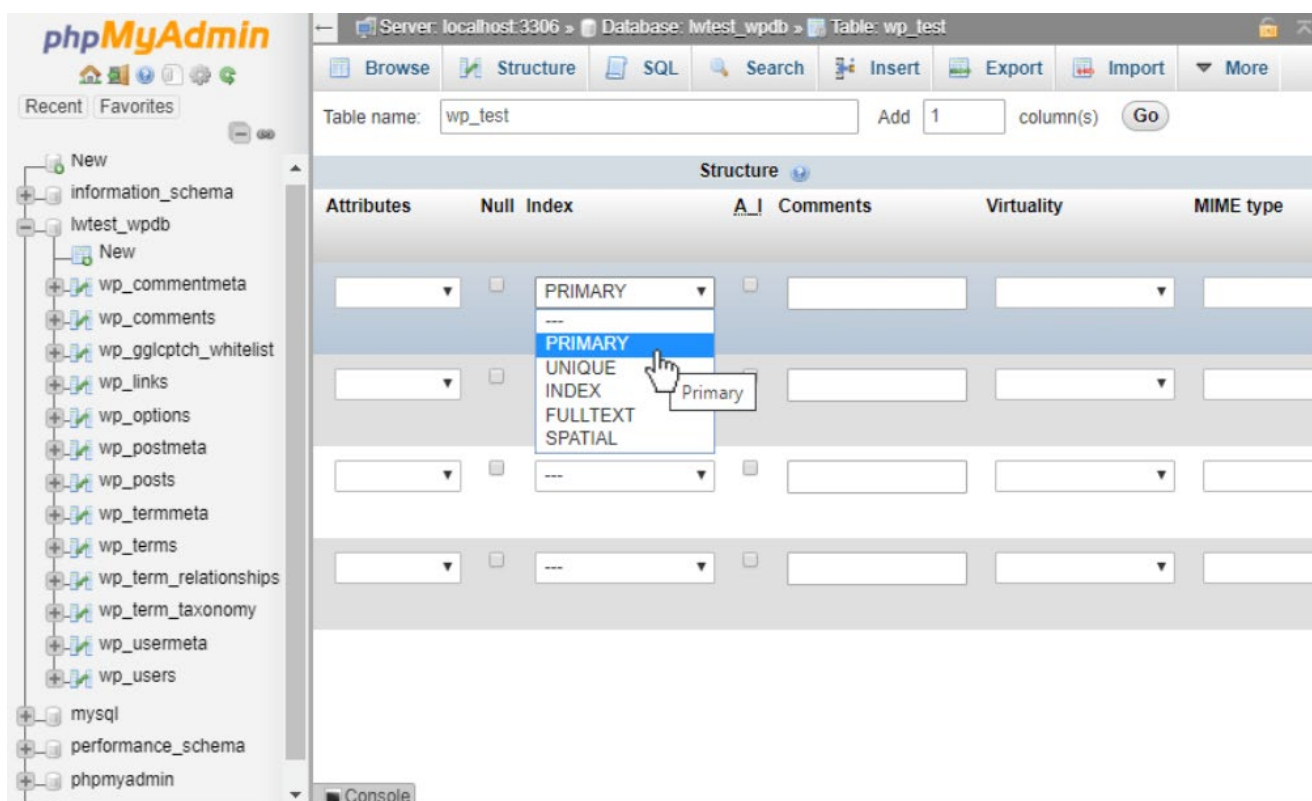


Рисунок 2.24 - Створення таблиці з даними про температуру

Далі зберігаємо таблицю, і створюємо аналогічну для збереження даних про вологості повітря. Далі створюємо таблицю з найменуваннями плат podetcsi рисунок 2.25.

У підсумку створили три таблиці, не пов'язані між собою, зроблено це для подальшої розширюваності й для того, що датчики збирають різні дані і приходять вони в різні проміжки часу. Дані з однотипних датчиків будуть зберігатися в одну таблицю.

Оскільки у нас локальний сайт, то необхідно прописати асоціацію 127.0.0.1 teplica.org у файлі /etc/hosts.

Після визначення полів натисніть "Зберегти" чи еквівалентну опцію для збереження таблиці.

Тепер у вас повинна бути створена таблиця "temperature_data" з визначеними полями. Вона буде використовуватися для збереження даних про температуру, які надходять із датчика вологості.

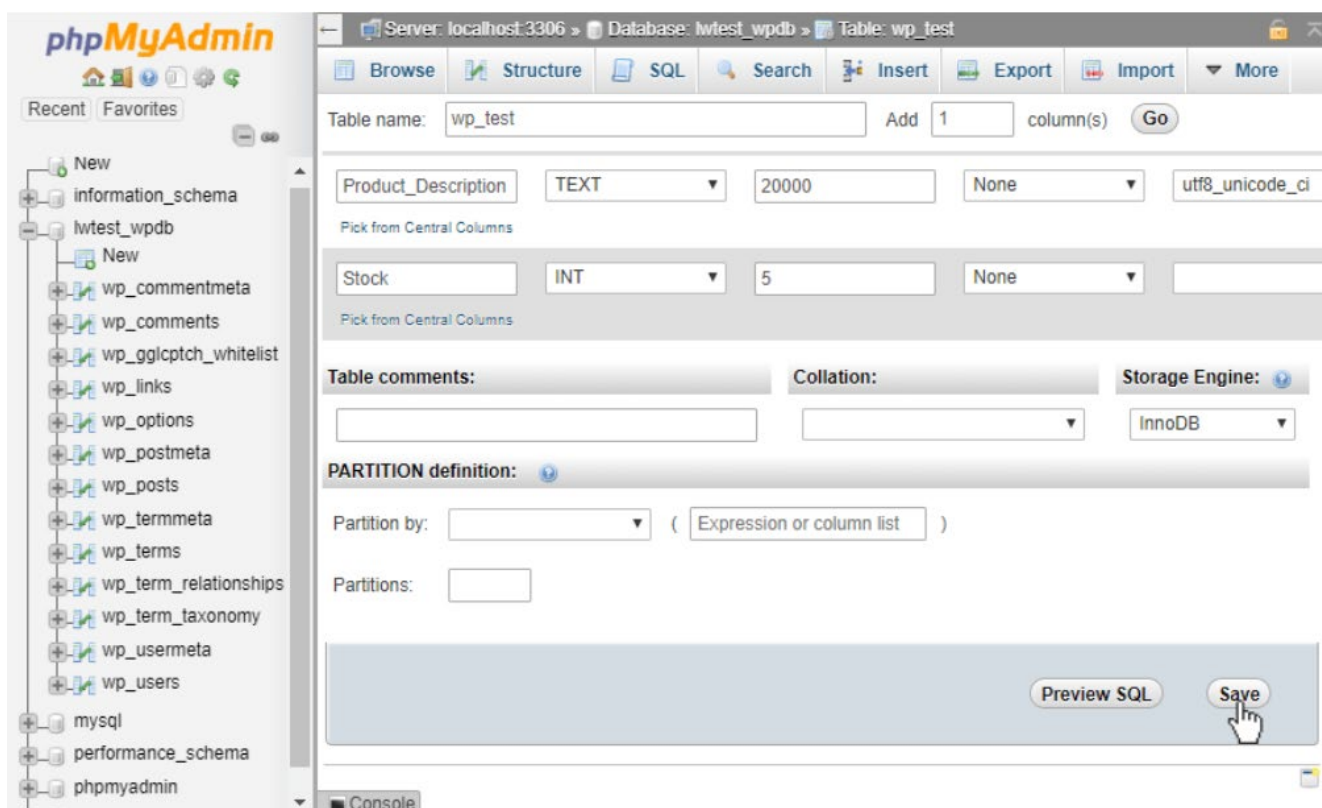


Рисунок 2.25 - Таблиця в базі даних

На рисунку 2.26 показано підсумку ву структуру таблиць у базі даних.

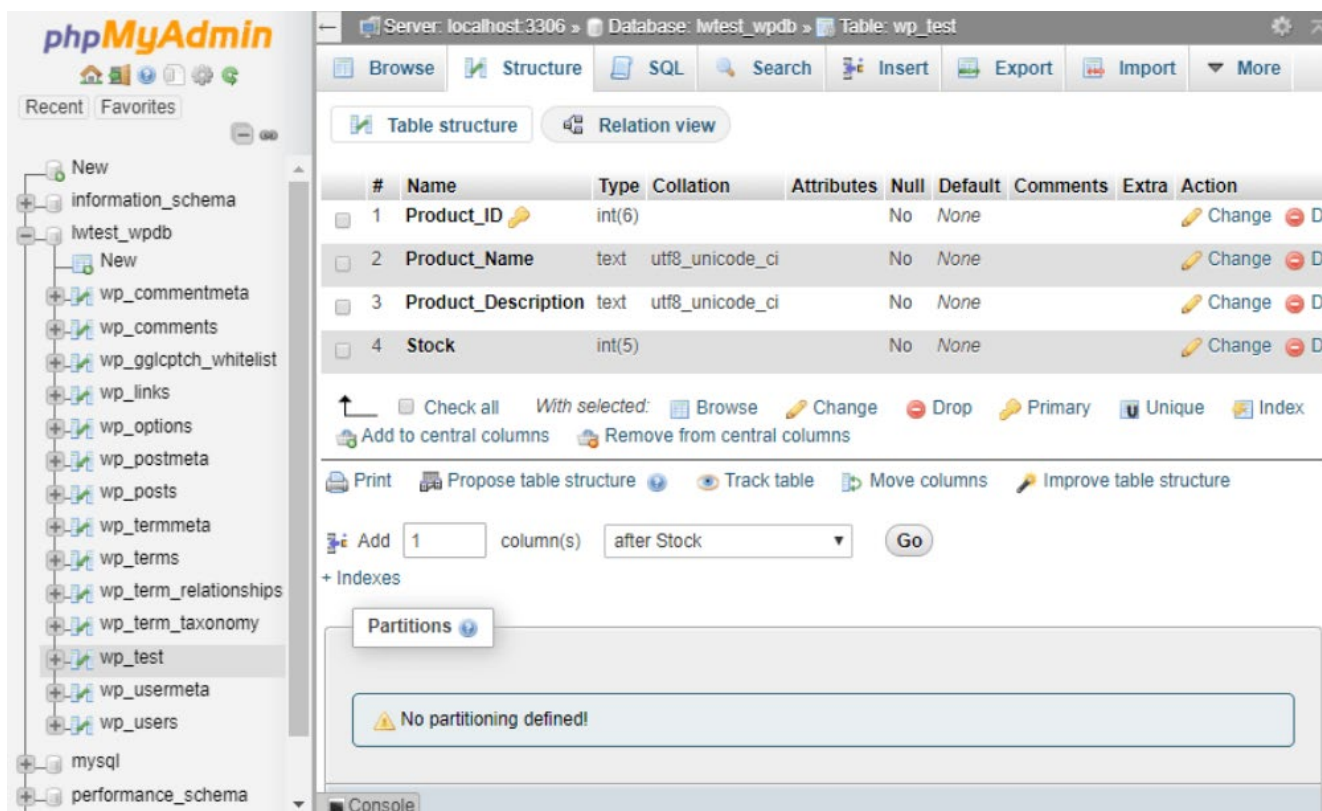


Рисунок 2.26 - Структура таблиці з усіма таблицями

Для того щоб адмініструвати Raspberry Pi віддалено, для застосування або зміни будь-яких параметрів програми, активуємо доступ за SSH (англ. Secure Shell - "безпечна оболонка") - це мережевий протокол прикладного рівня, який дає змогу виконувати віддалене керування операційною системою та тунелювання з'єднань, шифрує весь трафік і паролі. SSH допускає вибір різних алгоритмів шифрування. Здійснюємо увімкнення за допомогою команди `sudo systemctl enable ssh`
`sudo systemctl start ssh` [15].

На цьому розробку програмно-апаратного комплексу для управління тепличним господарством завершено.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Опис продукту

Згідно з вимогами технічного завдання було реалізовано програмно-апаратний комплекс автоматизованої системи управління тепличним господарством.

Реалізовано модель теплиці, в якій встановлено налаштований мікроконтролер із датчиками. Реалізовано автоматичний і ручний режим управління підключеним навантаженням. Для ручного керування на мікроконтролері реалізовано веб-сервер з інтерфейсом. Це рішення необхідне для резервного керування теплицею в разі втрати зв'язку з центральним сервером. На сторінці відображається стан режиму роботи, поливу, обігріву, відчинене чи зачинене вікно і чи увімкнене світло, а також дані про температуру і вологість повітря. На рисунках 41-43 зображено web-інтерфейс для керування теплицею в режимі "AUTO" і в ручному режимі.

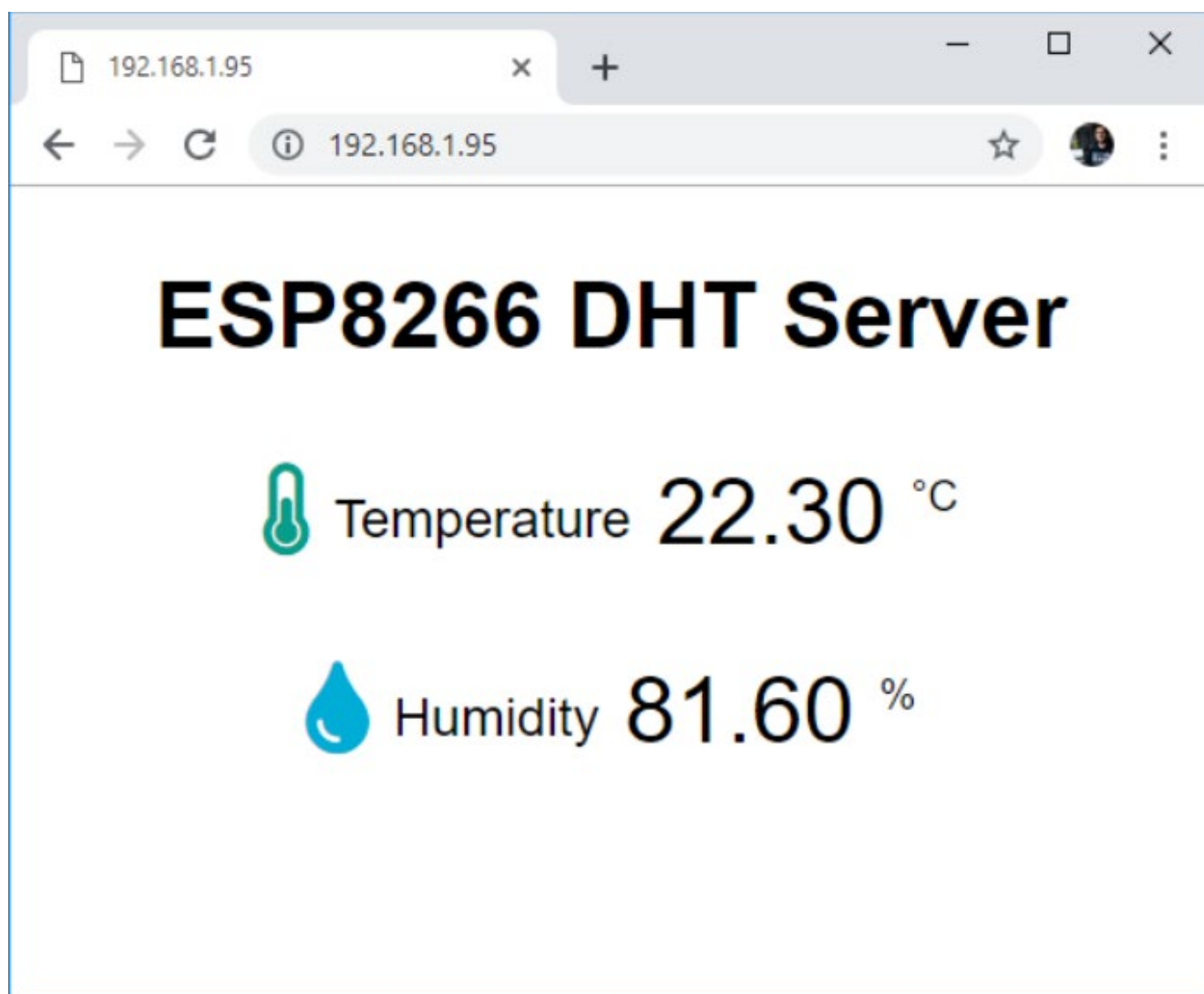


Рисунок 3.1 - Увімкнено режим "AUTO"

Опис вказує на реалізацію централізованого управління теплицями з використанням MQTT-брокера та веб-інтерфейсу для зручного керування та відображення даних. Давайте розглянемо кілька ключових аспектів цієї системи:

Домен `teplica.org`. Домен `teplica.org`, ймовірно, налаштований для вказаного IP-адреси сервера чи хостується через DNS-сервіс.

Ця система дозволяє ефективно збирати дані від теплиць за допомогою MQTT та надає зручний інтерфейс для їх управління та моніторингу через веб.

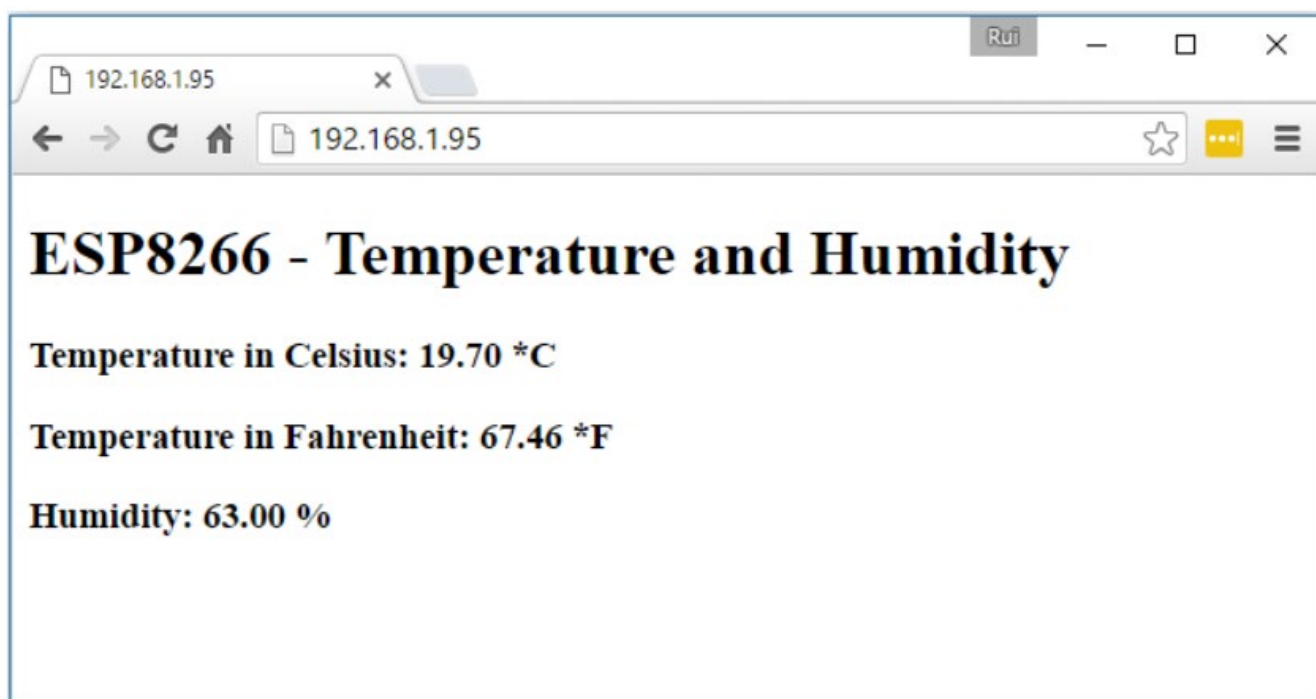


Рисунок 3.2 - Ручний режим керування

На сервері реалізовано централізоване управління теплицями. Встановлено та налаштовано mqtt-брокер, для обробки публікованих даних від мікроконтролерів. Написано веб-інтерфейс для зручного керування теплицями і відображення збережених даних, для переходу в браузері прописується адреса teplica.org.

Встановили та налаштували MQTT-брокер. MQTT (Message Queuing Telemetry Transport) - це протокол передачі повідомлень, який часто використовується для спілкування між IoT-пристроями. Брокер відповідає за прийом, обробку та розподіл повідомлень.

Мікроконтролери, ймовірно, використовують MQTT для публікації даних про температуру (імовірно, з датчиків вологості) до MQTT-брокера.

Веб-інтерфейс для зручного керування теплицями та відображення збережених даних.

Веб-інтерфейс показано на рисунках 3.3-3.4.

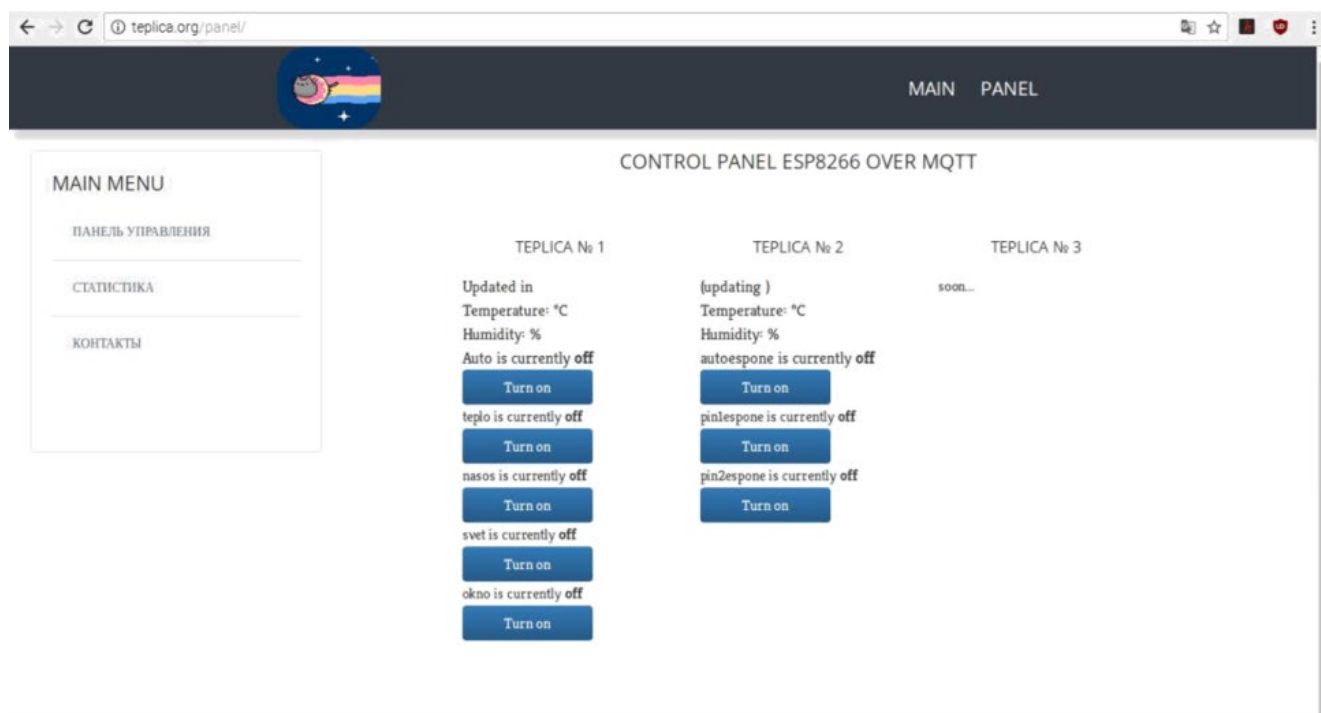


Рисунок 3.3 - Сторінка управління теплицями

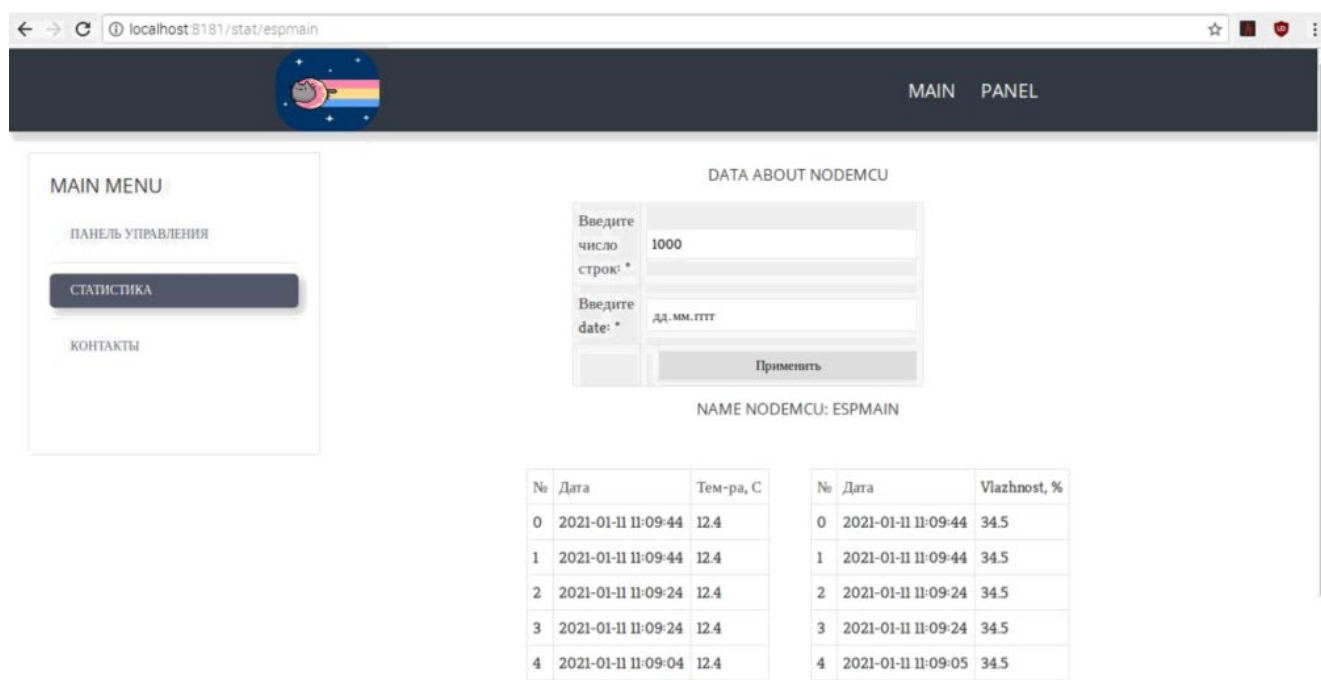


Рисунок 3.4 - Сторінка з даними теплиці ESPMAIN

Для зручності доступу до веб-інтерфейсу ви налаштували перехід за допомогою адреси `teplica.org` в браузері. Це може бути реалізовано через конфігурацію DNS або налаштування веб-сервера.

3.2 Спеціалізовані показники. Підключення бібліотек

Для під'єднання до операційної системи по SSH можна використовувати програму безкоштовну програму PuTTY. Для цього вводимо ір-адресу нашої машинки і порт 22, за яким працює SSH. На рисунках 47 показано інтерфейс програми PuTTY, а на рисунку 3.5 термінал, у якому показано відкриту сесію [32].

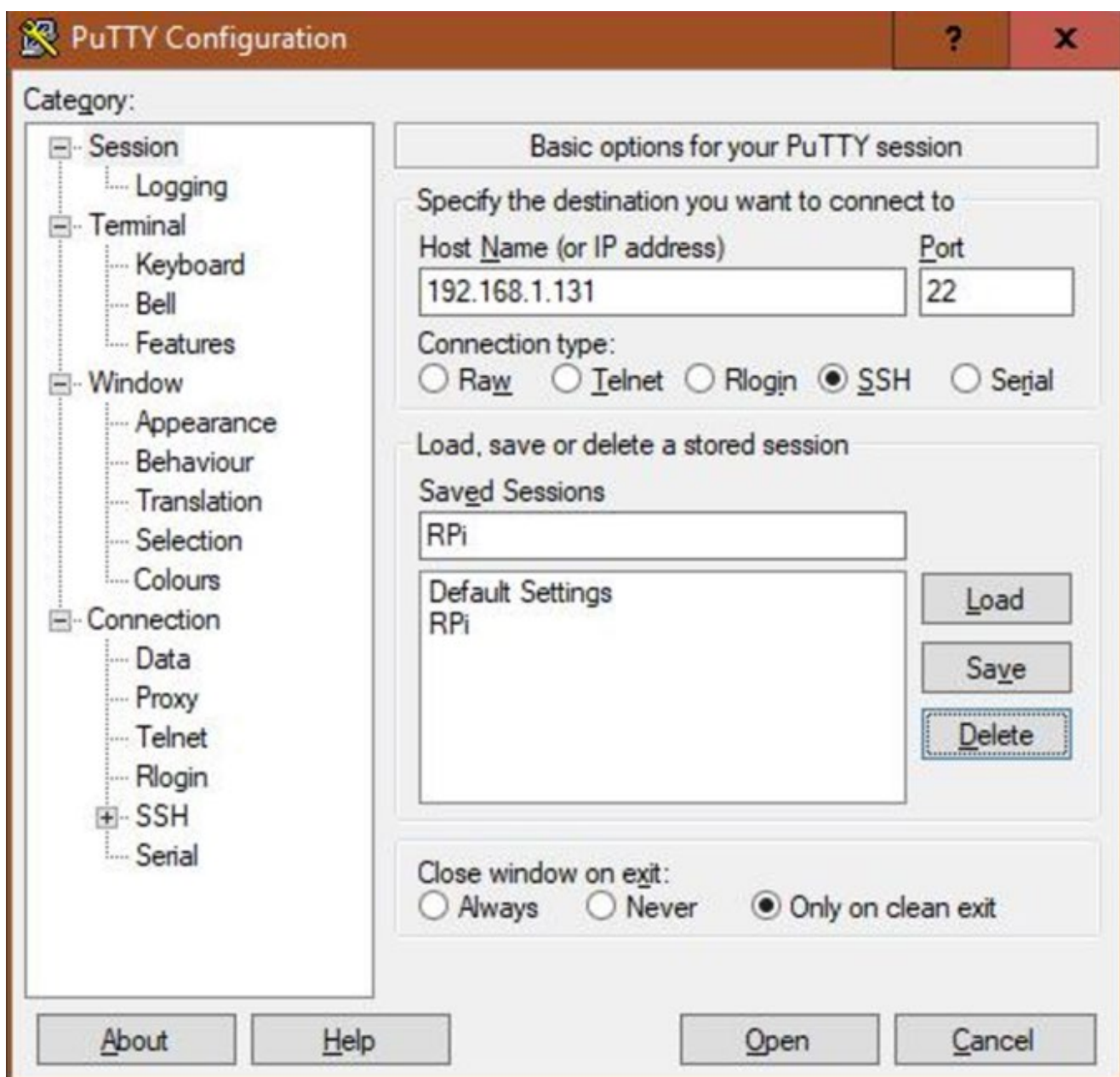
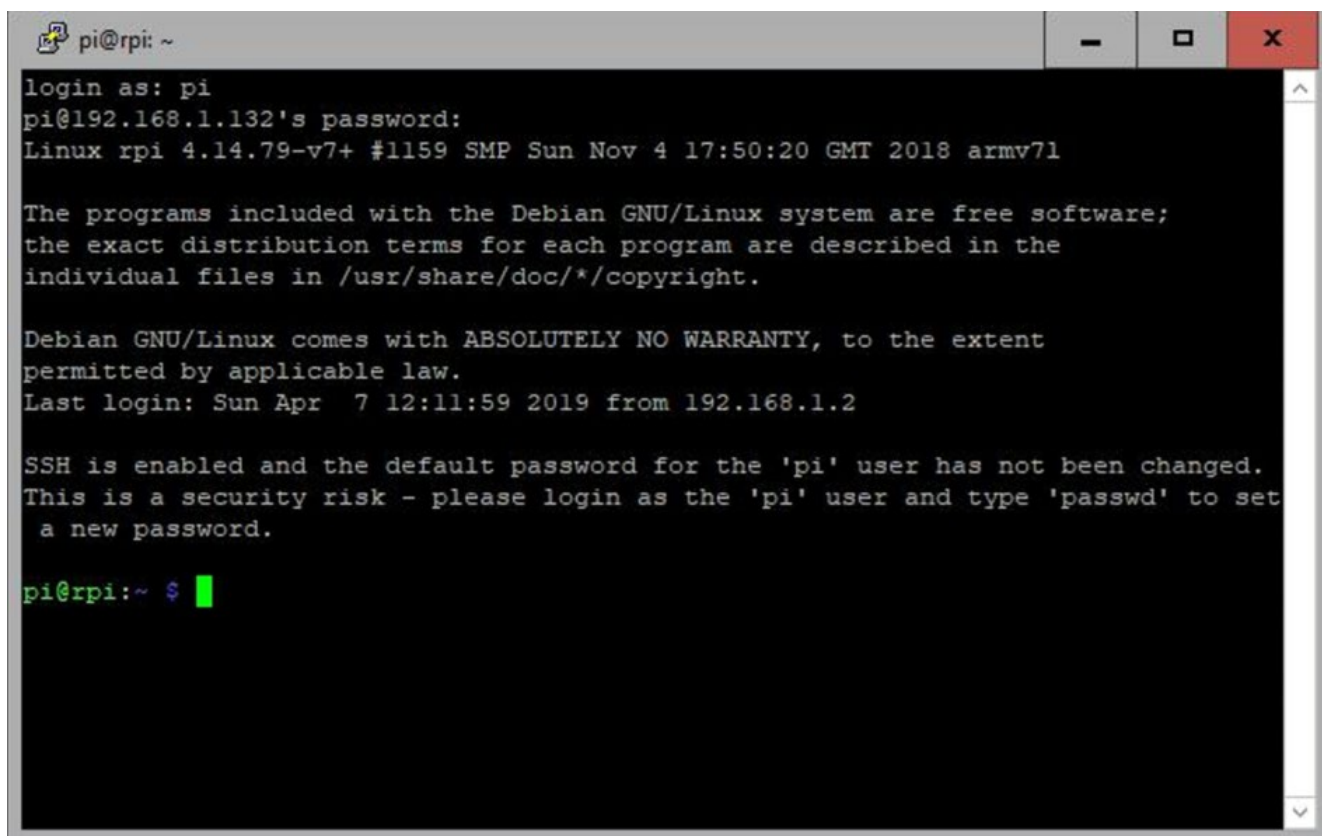


Рисунок 3.5 - Програма PuTTY

PuTTY - це безкоштовний клієнт для роботи з різними мережевими протоколами, такими як SSH, Telnet, та інші. Програма дозволяє вам встановлювати з'єднання з віддаленими серверами або мережевими пристроями. Нижче наведено кілька ключових аспектів та кроків для користування PuTTY:



```
pi@rpi: ~
login as: pi
pi@192.168.1.132's password:
Linux rpi 4.14.79-v7+ #1159 SMP Sun Nov 4 17:50:20 GMT 2018 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Apr  7 12:11:59 2019 from 192.168.1.2

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
a new password.

pi@rpi:~ $
```

Рисунок 3.6 - Відкрита сесія з Raspberry Pi

Також є можливість підключатися за протоколом SFTP для заміни файлів або розширення можливостей. SFTP (англ. SSH File Transfer Protocol) - це протокол прикладного рівня, призначений для роботи з файлами поверх надійного та безпечного з'єднання. Підключення можна здійснювати за допомогою програми WinSCP - вільний графічний клієнт протоколу SFTP. Забезпечує захищене копіювання файлів між комп'ютером і серверами, що підтримують ці протоколи.

У цьому розділі представлено макет теплиці, написано код, який виконує певний алгоритм дій, що залежить від часу доби та погодних умов, також реалізовано веб-інтерфейс для взаємодії між системою та користувачем, а також присутнє виведення інформації.

ВИСНОВКИ

Під час роботи було повністю реалізовано програмно-апаратний комплекс автоматизованої системи управління тепличним господарством на базі мікроконтролерів.

У процесі роботи над проєктом було виконано всі поставлені завдання: було вивчено принципи побудови автоматизованих систем, зроблено аналіз і вибір програмного та апаратного забезпечення для реалізації проєкту. Було створено фізичну модель теплиці. Показано використовувані датчики та елементи, мікроконтролер і серверну частину проєкту, а також проведено налаштування всіх компонентів. Створено програмно-апаратний комплекс на базі мікроконтролера плати NodeNCU (ESP8266). Написано веб-інтерфейс на кожному мікроконтролері. Теплиця з автоматичним регулюванням температури і поливом, датчиком руху для вмикання світла в темний час доби. Також реалізовано mqtt-брокер і веб-сервер, на якому розміщено написаний веб-інтерфейс централізованого управління всіма теплицями, з відображенням поточного стану навантаження (освітлення, вікно, обігрівач, насос для поливу), а також температури і вологості повітря. Реалізовано автоматичний і ручний режим роботи. У ручному режимі керує навантаженням користувач сам, це буде зручно для тих, хто хоче самостійно контролювати процеси.

Цей програмно-апаратний комплекс допоможе тим, хто хоче вирощувати гарний урожай у теплиці. Також можна змінити автоматичний алгоритм, змінити навантаження і встановити в акваріум або тераріум, щоб підтримувати сприятливі умови в замкнутому просторі.

Розроблений проєкт відповідає технічному завданню і повністю готовий до експлуатації

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. D. Jones Creating Unified IT Monitoring and Management in Your Environment/ D. Jones: Realtime Publishers, 2012 – 92с
2. Alistair Croll Complete Web Monitoring/Alistair Croll, Sean Power: O'Reilly Media, 2009 – 672 с.
3. Andrea Dalle Vacche Zabbix Mastering. Andrea Dalle Vacche, Stefano Kewan Lee: Packt Publishing Ltd, 2013 – 358 с.
4. Nagios Nagios Documentation. URL: <http://www.nagios.org/> (Last accessed: 30.10.2023).
5. Zabbix URL: What is Zabbix – <http://www.zabbix.com> (Last accessed: 30.10.2023).
6. TechNet Magazine. System Center Operations Manager URL: <https://www.technetmagazine.com/> (Last accessed: 30.10.2023).
7. 2018: Простота розширення можливостей моніторингу – URL: <http://technet.microsoft.com> (Last accessed: 30.10.2023).
8. N. Berardi. Your First Fluent Cassandra Application- URL: <http://coderjournal.com/2010/06/your-first-fluent-cassandra-application/>(Last accessed: 30.10.2023).
9. CQL Scepification. CQL BINARY PROTOCOL v1 specification URL: https://github.com/apache/cassandra/blob/cassandra-1.2/doc/native_protocol.spec (Last accessed: 30.10.2023).
10. Cassandra Wiki: The Cassandra Thrift API Overview URL: <http://wiki.apache.org/cassandra/API10> (Last accessed: 30.10.2023).
11. Википедия: Apache Thrift URL: http://en.wikipedia.org/wiki/Apache_Thrift (Last accessed: 30.10.2023).
12. Elastic Search: Elastic Search Guide URL: <http://www.elasticsearch.org/guide/> (Last accessed: 30.10.2023).
13. Solarwinds: Server & Application Monitor URL:

<http://www.solarwinds.com/server-application-monitor.aspx> (Last accessed: 30.10.2023).

14. Zabbix Documentation: Zabbix documentation for version 2.0. URL: <https://www.zabbix.com/documentation/2.0> (Last accessed: 30.10.2023).

15. ITFB: Моніторинг серверів, сервісів і сайтів URL: <http://www.itfb.com.ua/monitoring.html> (Last accessed: 30.10.2023).

16. Jet Brains: TeamCity User Guide URL: <http://www.jetbrains.com/teamcity/documentation/userguide.jsp> (Last accessed: 30.10.2023).

17. microsoft.com: Introduction to the Deployment Guide for Monitoring Server - URL: [http://technet.microsoft.com/library/dd630725\(v=office.12\).aspx](http://technet.microsoft.com/library/dd630725(v=office.12).aspx) (Last accessed: 30.10.2023).

18. Oreilly: Could They Do It?: Real User Monitoring URL: <http://oreilly.com/web-development/excerpts/9780596155131/chapter-10.html> (Last accessed: 30.10.2023).

19. Арнольд М. Адміністрування Apache / Арнольд М., Алмейда Дж., Міллер К. - Лорі, 2016. - 418 с.

20. Широтно-імпульсна модуляція. - Текст: електронний // URL:<http://electricalschool.info/electronica/1759-shirotno-impulsnaja-moduljacija.html>, (дата звернення: 10.10.2023).