Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(освітній рівень)

на тему: Методи та засоби розробки комп'ютерної системи моніторингу мікроклімату серверної кімнати

Виконав: студент (ка) VI курсу, групи СІм-61

Спеціальності: 123 "Комп'ютерна інженерія"
(шифр і назва спеціальності)

Василишин В. В.
підпис (прізвище та ініціали)

Керівник Луцик Н. С.
підпис (прізвище та ініціали)

Нормоконтроль Тиш Є.В.
підпис (прізвище та ініціали)

Завідувач кафедри Осухівська Г. М.
підпис (прізвище та ініціали)

Рецензент Никитюк В. В.
підпис (прізвище та ініціали)

м. Тернопіль – 2023

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

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

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ *Осухівська Г. М.*
(підпис) (прізвище та ініціали)
«__» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня *магістр*
(назва освітнього ступеня)
за спеціальністю *123 Комп'ютерна інженерія*
(шифр і назва спеціальності)
Студенту *Василишину Вадиму Віталійовичу*
(прізвище, ім'я, по батькові)
1. Тема роботи *Методи та засоби розробки комп'ютерної системи моніторингу мікроклімату серверної кімнати*

Керівник роботи *Луцик Надія Степанівна*
доктор філософії, доцент кафедри комп'ютерних систем та мереж
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
Затверджені наказом ректора *від « 01 » 12 2023 року № 4/7-1132*
2. Термін подання студентом завершеної роботи *22.12.2023*
3. Вихідні дані до роботи *Вимоги до систем моніторингу, технічні характеристики давачів документація на модуль Raspberry Pi Pico W.*

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ.
1. Аналіз систем моніторингу.
2. Схеми та засоби створення системи моніторингу.
3. Розробка системи моніторингу.
4. Охорона праці та безпека в надзвичайних ситуаціях.
Висновки.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Актуальність теми, мета і завдання дослідження, об'єкт та предмет дослідження.
2. Методи дослідження, наукова новизна та практичні результати.
3. Схема розробленої системи моніторингу.
4. Електрична принципова схема моніторингу параметрів.
5. Блок-схема алгоритм роботи блоку моніторингу на основі Raspberry Pi Pico W.
6. Блок-схема алгоритм роботи з базою даних та Telegram-Bot.
7. Приклади повідомлень системи моніторингу.
8. Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	*Осухівська Г.М., к.т.н., доцент*		
Безпека в надзвичайних ситуаціях	*Клепчик В.М., проректор з адміністративно-господарської роботи та будівництва*		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	*Вступ*		*Виконано*
2.	*Аналіз систем моніторингу.*		*Виконано*
3.	*Схеми та засоби створення системи моніторингу.*		*Виконано*
4.	*Розробка системи моніторингу.*		*Виконано*
5.	*Охорона праці та безпека в надзвичайних ситуаціях*		*Виконано*
6.	*Висновки.*		*Виконано*
7.	*Оформлення пояснювальної записки*		*Виконано*
8.	*Оформлення графічної частини*		*Виконано*
9.	*Попередній захист кваліфікаційної роботи магістра*		*Виконано*
10.	*Захист кваліфікаційної роботи*		

Студент _____ *Василишин В.В.*
(підпис) (прізвище та ініціали)

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

АНОТАЦІЯ

Методи та засоби розробки комп’ютерної системи моніторингу мікроклімату серверної кімнати // Кваліфікаційна робота магістра // Василишин Вадим Віталійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп’ютерно-інформаційних систем і програмної інженерії, кафедра комп’ютерних систем та мереж, група СІм-61 // Тернопіль, 2023 // с. – 72, рис. – 45, аркушів А1 – 8, додат. – 5, бібліогр. – 35.

Ключові слова: IoT, Raspberry Pi Pico W, MQTT, моніторинг, давач, Telegram-bot.

Магістерська робота спрямована на створення та перевірку ефективності відстеження важливих параметрів середовища, таких як температура, вологість, та інші важливі показники, що впливають на функціонування серверного обладнання.

Дослідження включало аналіз існуючих систем моніторингу, розробку та впровадження нової системи моніторингу на основі Raspberry Pi Pico W та давачів, які вимірюють температуру, вологість а також інші параметри. У разі виявлення критичних змін у стані мікроклімату серверної кімнати, система автоматично надсилає сповіщення користувачу. Це дозволяє швидко реагувати на потенційні проблеми та запобігати пошкодженням обладнання.

Результати кваліфікаційної роботи можуть бути використані для відстеження стану серверних кімнат і не тільки.

ABSTRACT

Methods and means of developing a computer monitoring system for the microclimate of a server room // Master's graduation thesis // Vasylyshyn Vadym // Ivan Pulyu Ternopil National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Systems and Networks, group СІм-61 // Ternopil, 2023 // p. – 72, fig. - 45, sheets A1 - 8, add. – 5, bibliography - 35.

Keywords: IoT, Raspberry Pi Pico W, MQTT, monitoring, sensor, Telegram-bot.

The master's thesis is aimed at creating and checking the effectiveness of tracking important environmental parameters, such as temperature, humidity, and other important indicators that affect the functioning of server equipment.

The research included analysis of existing monitoring systems, development and implementation of a new monitoring system based on Raspberry Pi Pico W and sensors that measure temperature, humidity and other parameters. If critical changes in the state of the microclimate of the server room are detected, the system automatically sends a notification to the user. This allows you to quickly react to potential problems and prevent equipment damage.

The results of qualification work can be used to track the state of server rooms and not only that.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

MQTT	Message Queue Telemetry Transport
I2C	Inter-Integrated Circuit
SPI	Serial Peripheral Interface
UART	Universal asynchronous receiver/transmitter
GSM	Groupe Special Mobile
GPRS	General Packet Radio Service
GPIO	General-purpose input/output
IoT	Internet of Things

ВСТУП

Актуальність теми. Оскільки значна частина діяльності компаній залежить від ефективності її IT-інфраструктури, забезпечення оптимального клімату у серверних приміщеннях стає критично важливим. Надмірне тепло, вологість або затоплення призводять до збоїв у роботі обладнання, що, у свою чергу, впливає на роботу бізнесу. Таким чином, розробка ефективних систем моніторингу параметрів серверних кімнат є важливою та актуальною темою.

Мета і завдання дослідження. Метою даного дослідження є розробка надійної, вартісно-ефективної системи моніторингу клімату для серверної кімнати.

Основні завдання включають:

- аналіз існуючих рішень;
- розробка системи на базі Raspberry Pi Pico W;
- інтеграція з MQTT та MySQL;
- реалізація Telegram-bot для сповіщення в реальному часі;
- підтвердження роботоздатності розробленої системи шляхом розгортання та тестування у серверній кімнаті;

Об’єкт дослідження. Об'єктом дослідження є процес моніторингу клімату в серверних кімнатах та оповіщення в критичних ситуаціях.

Предмет дослідження. Предметом дослідження є методи та засоби розробки системи моніторингу, зокрема можливість використання Raspberry Pi Pico W, давачів FC-37, MQ-2, DHT22 та SCT-013-000, а також програмного забезпечення на базі MQTT, MySQL, Python та MicroPyhton.

Методи дослідження. В дослідженні було використано методах аналізу, синтезу а також експерименту для практичного тестування розробленої системи.

Наукова новизна одержаних результатів кваліфікаційної роботи. Вперше спроектовано та розроблено комп’ютерну систему моніторингу мікроклімату серверної кімнати на основі Raspberry Pi Pico W, давачів та MQTT

з використанням мови програмування MicroPython та Python з можливістю оповіщення про критичні ситуації в Telegram-bot.

Практичне значення одержаних результатів. Кваліфікаційна робота магістра включає розробку та впровадження програмно-апаратних комплексів, здатних у режимі реального часу надавати інформацію щодо параметрів моніторингу та сповіщати про настання критичних ситуацій. Практичне значення роботи полягає у можливості впровадження розробленої системи в серверних кімнатах для моніторингу мікроклімату.

Публікації. Результати дослідження апробовано на XI Науково-технічної конференції «Інформаційні моделі, системи та технології», XII Міжнародна науково-технічна конференція молодих учених та студентів «Актуальні задачі сучасних технологій».

Структура роботи. Робота складається з пояснювальної записки та графічної частини. Пояснювальна записка складається із вступу, 4 розділів, висновків, списку використаних джерел та додатків. Обсяг роботи: пояснювальна записка – 72 аркушів формату А4, графічна частина – 8 аркушів формату А1.

РОЗДІЛ 1
АНАЛІЗ СИСТЕМ МОНІТОРИНГУ

1.1. Параметри моніторингу серверних кімнат

Серверні кімнати сьогодні є не лише центром обробки та зберігання даних для бізнесу, але й критично важливим елементом, що підтримує безперервність його діяльності. Правильне функціонування таких кімнат вимагає високої уваги до деталей щодо умов навколишнього середовища. Системи моніторингу, що відстежують різні параметри, відіграють ключову роль у забезпеченні оптимальних умов для серверного обладнання. Такий моніторинг охоплює не тільки температуру та вологість, але й безліч інших важливих параметрів, включаючи наявність диму, ризик затоплення та стабільність електропостачання.

Температурний режим у серверній кімнаті має вирішальне значення. Підвищення температури понад норму може призвести до перегріву серверів і, як наслідок, до їхнього некоректного функціонування або виходу з ладу. Такі збої не тільки вимагають витрат на ремонт або заміну обладнання, але й можуть призвести до втрати важливих даних або перерв у роботі важливих бізнес-процесів.

Вологість у серверній кімнаті також має велике значення. Занадто висока вологість збільшує ризик корозії металевих компонентів та може призвести до короткого замикання. Надмірно сухе повітря підвищує ризик створення статичної електрики, яка може пошкодити електронні компоненти. Моніторинг вологості дозволяє своєчасно вживати заходів для запобігання таких проблем, забезпечуючи стабільну та безпечну роботу обладнання.

Системи виявлення диму є необхідними для раннього виявлення пожеж у серверних кімнатах. Вони забезпечують можливість оперативної реакції на пожежу, що може врятувати не тільки обладнання, але й запобігти можливим

людським втратам. Ці системи зазвичай інтегруються з системами автоматичного пожежогасіння, що додатково збільшує рівень безпеки.

Також важливими є системи виявлення затоплення. Протікання води або затоплення можуть стати причиною серйозних пошкоджень електроніки. Раннє виявлення таких ситуацій дозволяє оперативно вживати заходів для запобігання або мінімізації збитків від води.

Стабільне електропостачання є важливим для серверних кімнат. Нестабільність напруги або відсутність електропостачання можуть викликати відмови у роботі обладнання, втрату даних або навіть пошкодження обладнання. Системи моніторингу, що відстежують напругу та частоту струму, є ключовими для забезпечення надійного живлення серверів. Вони також дозволяють виявляти відхилення в електромережі, що можуть свідчити про проблеми або потенційні перебої в подачі електроенергії.

Впровадження комплексних систем моніторингу у серверних кімнатах є важливою інвестицією для будь-якого підприємства. Такі системи забезпечують всебічний контроль над умовами середовища, що є вирішальним для забезпечення стабільності, безпеки та ефективності IT-інфраструктури. Вони не тільки допомагають запобігати непередбаченим збоям та втратам даних, але й сприяють оптимізації витрат та підвищенню загальної ефективності роботи серверного обладнання. У сучасному світі, де стабільність та надійність IT-систем мають вирішальне значення, роль ефективних систем моніторингу не може бути переоцінена.

1.2. Огляд існуючих варіантів рішень

Для моніторингу серверних кімнат існують як саморобні, так і комерційні рішення. Вибір між саморобними та комерційними рішеннями може залежати від бюджету, потреб безпеки, масштабу і складності інфраструктури, а також від готовності до інтеграції з існуючими системами та потреби у підтримці та обслуговуванні.

Система моніторингу серверної кімнати «tempCube» розроблена з застосуванням давачів та використовує бездротове з'єднання з блоком моніторингу [2]. Блок моніторингу може бути як окремим пристроєм або програмним забезпеченням на комп'ютері. Система оповіщення працює в реальному часі, оповіщення можуть надходити через додаток, електронну пошту та текстові повідомлення.

Недоліками даної системи моніторингу є те що давач потрібно заряджати. Також дана система є комерційним рішенням.

Наступна розробка демонструє систему моніторингу, яка включає моніторинг температури, вологості та руху [3]. Це саморобне рішення зі своїми очевидними недоліками.

Ця система складається з: давачів температури та вологості DHT-22, давача диму MQ-2, мікроконтролера Arduino та Arduino Ethernet Shield, вентилятор постійного струму, центру моніторингу (див.рис.1.2). Програмне забезпечення складається з вебдодатку, бази даних і вебсервера. Використовуючи сучасне обладнання та GSM для комунікацій, надаються гнучкі можливості для дистанційного моніторингу та управління. Завдяки вебінтерфейсу і GSM повідомленням система стає зручною для користувача.

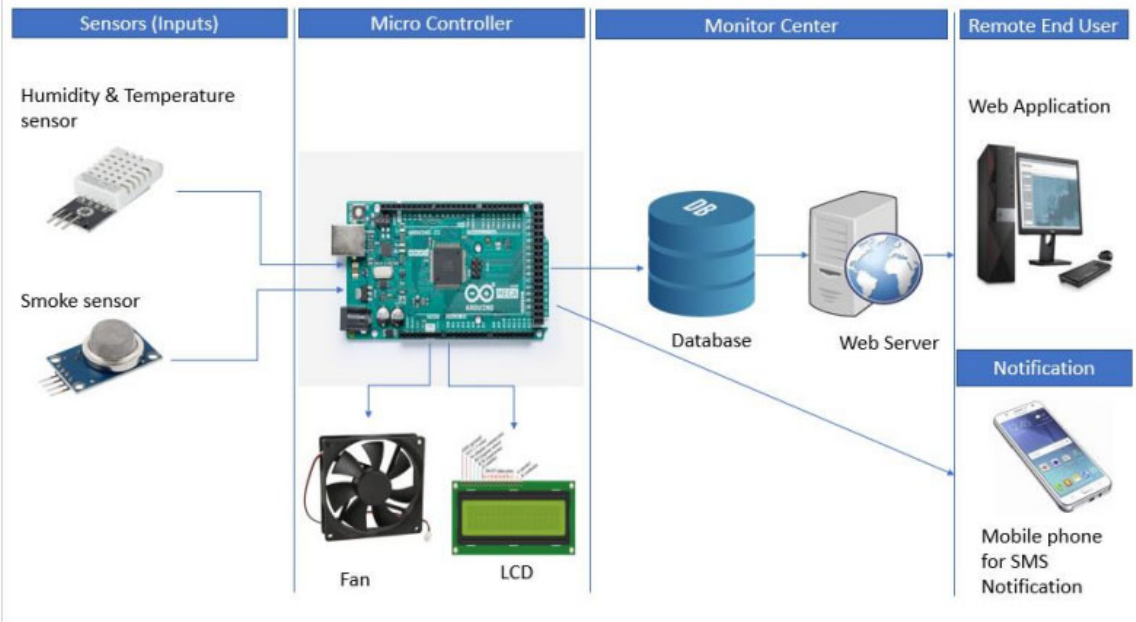


Рис. 1.2. Схема роботи системи

Певним недоліки системи є залежність від живлення. При відключенні електроенергії, можливості моніторингу та контролю будуть втрачені. Також може виникнути проблема з установкою системи, так як всі деталі потрібно налаштовувати самостійно.

Один з варіантів рішень це - багатофункціональне моніторингове обладнання «Monnit». Ця система містить 70 давачів, які здатні відслідковувати температуру, вологість, наявність води, змінний струм та інше [4].

Розробка складається з бездротових давачів, які контролюють умови в серверній кімнаті та надсилають цю інформацію через шлюз на платформу моніторингу (див.рис.1.3)

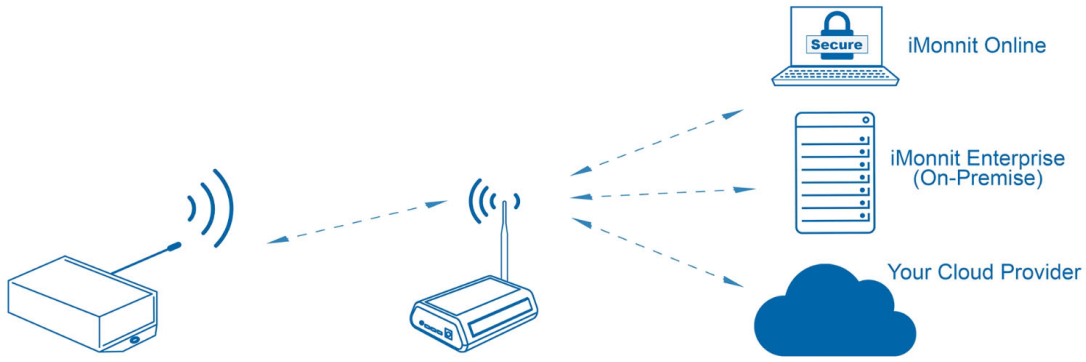


Рис. 1.3. Система моніторингу з бездротових давачів

Система моніторингу «ITWatchDogs» - це високоефективне рішення для моніторингу серверних кімнат, яке забезпечує надійний контроль та управління інфраструктурою даних [5]. Однією з ключових характеристик ITWatchDogs є здатність до комплексного моніторингу умов довкілля. Система забезпечує точне відстеження температури, вологості, потоку повітря, а також може виявляти наявність води чи інших рідин. Це дозволяє адміністраторам серверних кімнат вчасно реагувати на будь-які зміни умов, що можуть негативно вплинути на обладнання, наприклад, на збільшення температури, що може викликати перегрів серверів.

Система моніторингу ITWatchDogs також включає в себе функції безпеки, такі як детектори руху та дверні контакти, що забезпечують додатковий рівень захисту від несанкціонованого доступу.

Ще однією особливістю ITWatchDogs є його гнучкість у налаштуванні сповіщень та тривог. Система дозволяє користувачам налаштовувати оповіщення через електронну пошту, SMS або навіть через мобільні додатки.

Інтеграція з іншими системами управління та моніторингу - ще одна важлива особливість ITWatchDogs. Система може легко інтегруватися з існуючими рішеннями для керування інфраструктурою дата-центрів, що дозволяє централізувати моніторинг та управління всіма аспектами інфраструктури.

Завдяки своїм продуманим функціям, ITWatchDogs стає відмінним вибором для підприємств, які прагнуть забезпечити надійність та безпеку своїх серверних кімнат. Ефективний моніторинг умов довкілля, розширені можливості безпеки та гнучкість в налаштуванні сповіщень роблять ITWatchDogs незамінним інструментом для забезпечення стабільності та надійності критично важливих IT-операцій. Але це дороге комерційне рішення, яке не підходить для закладів освіти.

Система AVTECH є провідним рішенням для моніторингу серверних кімнат, відомим своєю здатністю забезпечувати докладний контроль та управління важливою інфраструктурою [6]. Це рішення зосереджене на забезпеченні надійності, безпеки та оптимальних умов роботи для серверного обладнання. Основною характеристикою AVTECH є його глибокий моніторинг довкілля. Система пропонує точне відстеження критичних параметрів, таких як температура, вологість, рівень звуку, а також може визначати наявність води або диму. Це особливо важливо для запобігання перегріву серверів та інших непередбачуваних ситуацій, що можуть призвести до збоїв у системі.

Безпека - ще один ключовий аспект системи моніторингу AVTECH. Вона включає давачі виявлення руху та доступу до серверної кімнати, забезпечуючи

захист від несанкціонованого втручання. Це важливо не тільки для фізичної безпеки обладнання, але й для забезпечення цілісності даних.

Однією з особливостей AVTECH є її система сповіщень та тривоги. Вона дозволяє налаштовувати автоматичні оповіщення, які можуть бути відправлені через електронну пошту, SMS або навіть через мобільні додатки, забезпечуючи тим самим, що управлінський персонал завжди буде в курсі потенційних проблем у серверній кімнаті. Що стосується інтеграції, AVTECH легко сумісна з іншими системами управління дата-центрами, що дозволяє централізувати управління та моніторинг усіх аспектів інфраструктури. Це забезпечує гнучкість та ефективність у роботі з різними IT-системами та обладнанням.

У підсумку, AVTECH стає ідеальним рішенням для тих, хто шукає надійний та всебічний контроль за умовами у серверних кімнатах. Від моніторингу умов довкілля до захисту від несанкціонованого доступу та інтеграції з іншими системами - AVTECH забезпечує комплексний підхід до управління критично важливим IT-обладнанням. Це також дороге комерційне рішення, на яке з очевидних причин не буде попиту як у малого бізнесу так і в освітніх закладів.

Оптимальним рішенням для освітніх закладів та малого бізнесу є створення власної системи моніторингу, яка може задовільнити мінімальні вимого до параметрів моніторингу.

1.3. Порівняння мікроконтролерів

Існує багато мікроконтролерів на ринку з різними характеристиками, можливостями та призначеннями. Для саморобної системи моніторингу можливе використання трьох основних мікроконтролерів ESP8266, Raspberry Pi Pico та Arduino UNO. Кожен з яких вирізняється власними особливостями, що робить їх придатними для різних застосувань.

Процесор в ESP8266 має 32-бітний Tensilica L106 мікроконтролер, який забезпечує хороший баланс між потужністю та енергоефективністю, особливо важливим для IoT застосувань, де важлива економія енергії [7]. Raspberry Pi Pico

використовує свій власний чіп RP2040 з двома ядрами ARM Cortex-M0+, що забезпечує вищу продуктивність у порівнянні з ESP8266 [8]. Це робить Pico більш придатним для завдань, які вимагають більшої обчислювальної потужності. Arduino, як правило, використовує 8-бітні мікроконтролери від Atmel, такі як ATmega328 у Arduino Uno [11]. Ці процесори менш потужні порівняно з ESP8266 та Pico, але добре підходять для основних електронних проектів і прототипування.

Пам'ять в ESP8266 може бути до 160 КБ внутрішньої SRAM і до 4 МБ зовнішньої Flash-пам'яті, що дозволяє розміщувати відносно складні програми та зберігати більше даних [7]. Raspberry Pi Pico має 264 КБ SRAM та 2 МБ флеш-пам'яті, що надає йому перевагу у пам'яті над Arduino, але менше, ніж у ESP8266 за обсягом флеш-пам'яті [8]. Arduino Uno, з іншого боку, обмежений 2 КБ SRAM, 32 КБ Flash пам'яті та 1 КБ EEPROM [10]. Це робить Arduino менш підходящим для додатків, які потребують значного обсягу зберігання або обробки даних.

Інтерфейс ESP8266 має обмежену кількість GPIO пінів, але включає вбудований Wi-Fi, що робить його одним з гарних рішень для IoT застосувань [7]. Raspberry Pi Pico надає велику кількість GPIO пінів (26) і підтримує широкий спектр інтерфейсів, включаючи I2C, SPI та UART, що робить його дуже гнучким для підключення різноманітних пристроїв [9]. Arduino зазвичай пропонує достатню кількість GPIO пінів (наприклад, 14 у Uno) і підтримує стандартні комунікаційні інтерфейси, такі як I2C, SPI та UART, але без вбудованого Wi-Fi [11].

ESP8266 і Arduino живляться через micro-USB або безпосередньо від батареї, що робить їх гнучкими у виборі джерел живлення. Raspberry Pi Pico також підтримує живлення через micro-USB, але відрізняється високою енергоефективністю, що є важливим для батарейних або енергонезалежних застосувань [9].

Програмування та розробка на платформах ESP8266, Raspberry Pi Pico, та Arduino, мають між собою відмінності, які впливають на вибір платформи залежно від конкретних потреб розробників.

Платформа підтримує ряд мов програмування, включаючи Lua (через NodeMCU), Arduino C/C++, а також MicroPython [7]. Ця різноманітність мов робить ESP8266 доступним для широкого спектру розробників, від початківців до професіоналів. Наприклад, Lua пропонує простоту для тих, хто тільки починає, в той час як Arduino C/C++ ідеально підходить для тих, хто шукає більшу потужність та гнучкість. Використання Arduino IDE для програмування ESP8266 дозволяє легко інтегрувати його у вже існуючі проекти на Arduino, розширюючи їх функціональність.

Raspberry Pi Pico забезпечує розробникам потужну платформу з двома ядрами ARM Cortex-M0+ [8]. Що стосується мов програмування, Pico підтримує як C/C++, так і MicroPython, що дає можливість розробникам обирати мову в залежності від їхніх потреб та досвіду. MicroPython є привабливим вибором для тих, хто шукає простоту Python з можливостями мікроконтролерів, тоді як C/C++ відкриває ширші можливості для оптимізації та контролю на низькому рівні. Raspberry Pi Pico SDK, офіційне середовище для розробки, надає широкі можливості та документацію, що робить Pico привабливим для серйозних інженерних проектів.

Arduino, з іншого боку, є найбільш простішим до початківців варіантом в програмуванні. Програмування на Arduino виконується через Arduino IDE, яке пропонує простий і зрозумілий інтерфейс, ідеальний для освоєння основ програмування та електроніки. Arduino використовує варіацію C/C++, яка є простою у навчанні, але все ж має достатню потужність для реалізації широкого спектра проектів. Широка спільнота Arduino забезпечує безліч ресурсів, бібліотек та підтримки, що робить його відмінним вибором для освітніх цілей та хобі-проектів.

Raspberry Pi Pico пропонує значний потенціал у сфері освіти. Його доступність та простота використання роблять його відмінним інструментом для навчальних закладів, що прагнуть впроваджувати основи програмування та електроніки. Студенти мають можливість експериментувати з різними електронними компонентами, навчаючись основам мікроконтролерів та їх

застосування у реальному світі, від простих світлових дисплеїв до складніших автоматизованих систем.

У контексті промислового застосування, Raspberry Pi Pico відкриває можливості для розвитку невеликих, але потужних пристроїв. Його здатність до роботи з великою кількістю давачів і пристроїв дозволяє використовувати його у складних проектах, таких як контрольні системи, автоматизоване управління процесами, та інтеграція з іншими промисловими інтерфейсами.

Що стосується розширення можливостей, Pico може бути використаний у комбінації з різними модулями, такими як Bluetooth або LoRa, що значно розширює його функціональність. Це дозволяє розробникам інтегрувати Pico у більш складні мережеві проекти та IoT рішення, забезпечуючи бездротовий доступ та дистанційне управління.

Окрім технічних характеристик, важливою перевагою Raspberry Pi Pico є підтримка від Raspberry Pi Foundation, яка має сильну репутацію в галузі підтримки освітніх проектів. Ця підтримка виражається у постійних оновленнях програмного забезпечення, детальних посібниках для користувачів та активній спільноті користувачів, яка обмінюється знаннями та ресурсами.

У порівнянні з ESP8266 та Arduino, Raspberry Pi Pico виокремлюється своєю здатністю обробляти більш складні завдання, залишаючись при цьому доступним та легким у використанні. Це робить його універсальним інструментом, який відкриває широкі можливості для інноваційних проектів у різних сферах, від освіти до промисловості, і є ідеальним вибором для тих, хто шукає гнучкість, потужність і розширюваність у своїх проектах [12].

Розглядаючи застосування та спільноту/підтримку для ESP8266, Raspberry Pi Pico та Arduino, ми бачимо різні принципи використання та рівні підтримки, які визначають популярність та пристосування до різноманітних проектів.

Враховуючи наведені характеристики та можливості ESP8266, Raspberry Pi Pico та Arduino, можна зробити висновок, що Raspberry Pi Pico виокремлюється як найбільш універсальна та потужна платформа для широкого спектра застосувань.

Перш за все, Raspberry Pi Pico пропонує значно більшу обчислювальну потужність порівняно з ESP8266 та Arduino, завдяки своєму двоядерному процесору ARM Cortex-M0+. Це робить його ідеальним вибором для більш складних проектів, які вимагають паралельної обробки або виконання великої кількості обчислень. Наприклад, у проектах, пов'язаних з обробкою зображень, машинним навчанням або розширеним управлінням давачами, Pico здатний забезпечити необхідну продуктивність [13].

Щодо пам'яті, Raspberry Pi Pico з 264 КБ SRAM та 2 МБ флеш-пам'яті перевершує Arduino, надаючи більше простору для складніших програм та даних. Хоча він має менше флеш-пам'яті, ніж ESP8266, Pico компенсує це більшою оперативною пам'яттю та загальною продуктивністю.

У плані інтерфейсів вводу-виводу, Raspberry Pi Pico пропонує значно більшу кількість GPIO пінів та підтримує широкий спектр інтерфейсів, що робить його надзвичайно гнучким для інтеграції з різними давачами та пристроями. Ця гнучкість є ключовою для розробки різноманітних електронних пристроїв, від простих проектів до складних виробничих систем.

З точки зору живлення, висока енергоефективність Raspberry Pi Pico є важливою для проектів, що працюють від батарей або потребують енергонезалежності, що робить його підходящим для мобільних або віддалених застосувань.

Враховуючи ці аспекти, Raspberry Pi Pico, є найкращим мікроконтролером, який пропонує ідеальне поєднання потужності, гнучкості та доступності, що робить його найкращим вибором для широкого спектра застосувань, від освіти до промисловості.

1.4. Висновки до розділу

У розділі проведено огляд існуючих варіантів рішень систем моніторингу. Також здійснено порівняння мікроконтролерів ESP8266, Raspberry Pi Pico та Arduino UNO. Досліджено їхні можливості застосування в системах

моніторингу. Під час аналізу детально розглянута їх функціональність та можливість забезпечувати достовірний моніторинг серверних кімнат. Після ретельного аналізу мікроконтролерів Raspberry Pi Pico був обраний, як мікроконтролер для комп’ютерної системи моніторингу мікроклімату серверної кімнати.

РОЗДІЛ 2
СХЕМИ ТА ЗАСОБИ СТВОРЕННЯ СИСТЕМИ МОНІТОРИНГУ

2.1. Схема системи моніторингу

На початковому етапі розробки була розроблена схема системи моніторингу мікроклімату серверної кімнати (див.рис.2.1). Створення схеми - важливий крок для визначення компонентів, їх зв'язків та способів збору та обробки даних.

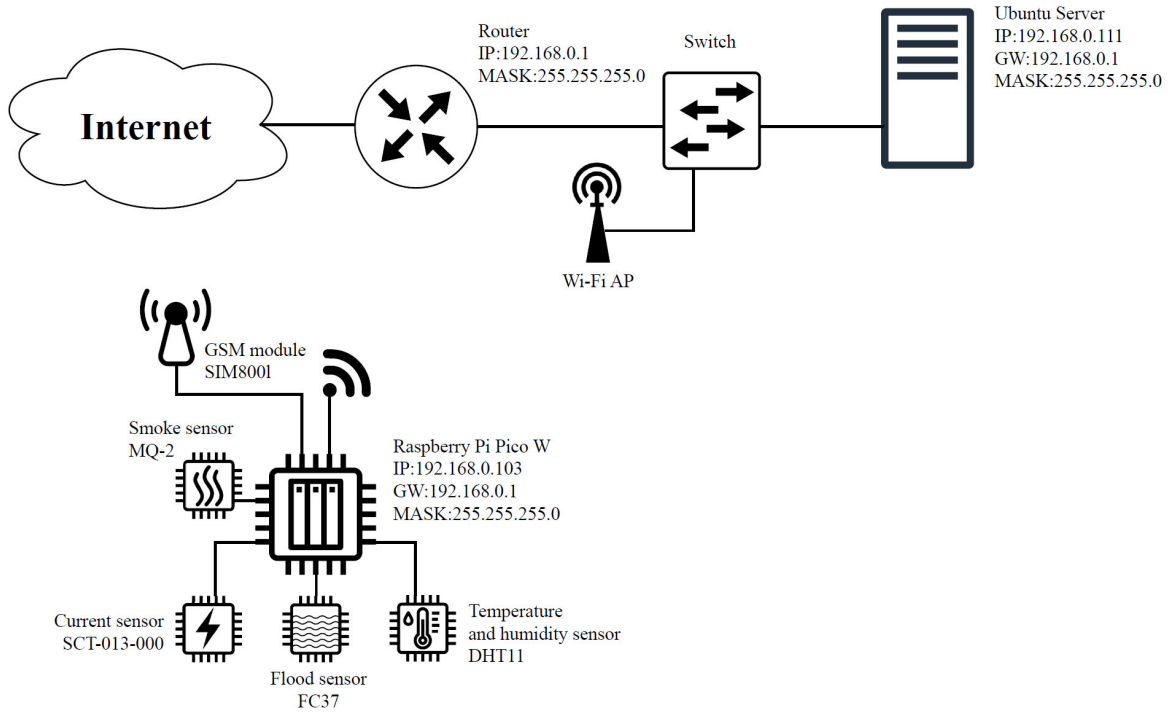


Рис. 2.1. Схема розробленої системи моніторингу

На представленій схемі зображено систему моніторингу параметрів серверної кімнати, яка складається з мікроконтролера Raspberry Pi Pico W з IP-адресою 192.168.0.103. Цей пристрій відповідає за збір даних з різних давачів та передає цю інформацію через мережу на сервер MQTT. Давачі, які включені до цієї системи, включають давач диму MQ-2, давач струму SCT-013-000, давач затоплення FC37, та давач температури і вологості DHT11.

Для екстреного інформування в систему інтегровано GSM модуль SIM800L, який використовується для відправки SMS повідомлень у разі виявлення аварійних ситуацій.

Подальша передача даних здійснюється через протокол MQTT на сервер Ubuntu, що має IP-адресу 192.168.0.111. Саме на цьому сервері встановлено програмне забезпечення для обробки та зберігання отриманих даних у базі даних, що дозволяє ефективно керувати статистичною інформацією про стан серверної кімнати.

Вихід в мережу Інтернет здійснюється через маршрутизатор з IP-адресою 192.168.0.1. Також на схемі зображено Wi-Fi точку доступу, яка забезпечує бездротовий зв'язок між мікроконтролером Raspberry Pi Pico W та маршрутизатором, а отже, і з інтернетом.

2.2. Опис компонентів системи моніторингу

2.2.1. Мікроконтролер Raspberry Pi Pico W.

Даний модуль, розроблений Raspberry Pi Foundation, є значним кроком вперед у світі мікроконтролерів, особливо в контексті доступності та функціональності. Цей пристрій, який побудований на базі RP2040 (див.рис.2.2), не лише продовжує успіх оригінального Raspberry Pi Pico, але й вносить значні вдосконалення, особливо з точки зору бездротового зв'язку.

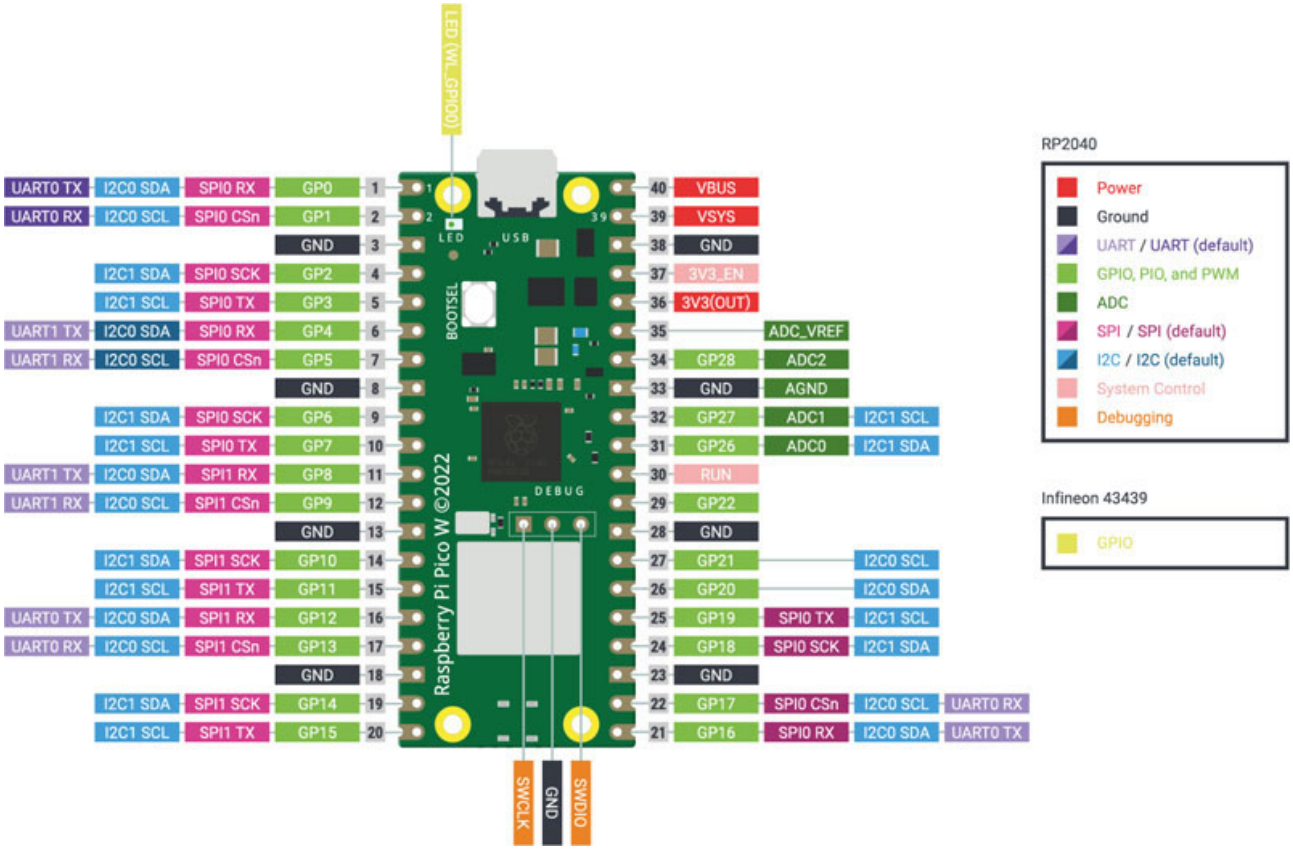


Рис. 2.2. Опис пінів модуля Raspberry pi Pico W

Чіп в Pico W, RP2040 (див.рис.2.3), є власною розробкою Raspberry Pi Foundation. Використання двоядерного ARM Cortex-M0+ процесора забезпечує достатньо обчислювальної потужності для обробки даних і управління різними задачами. Ці ядра об'єднані із системним вводом-виводом, що включає прямий доступ до пам'яті (DMA), дозволяючи ефективний обмін даними між різними компонентами мікроконтролера без надмірного навантаження на процесор. Тактова частота до 133 МГц, хоч і не є надзвичайно високою порівняно з повноцінними мікропроцесорами, є більш ніж достатньою для більшості проектів на основі мікроконтролерів [14].

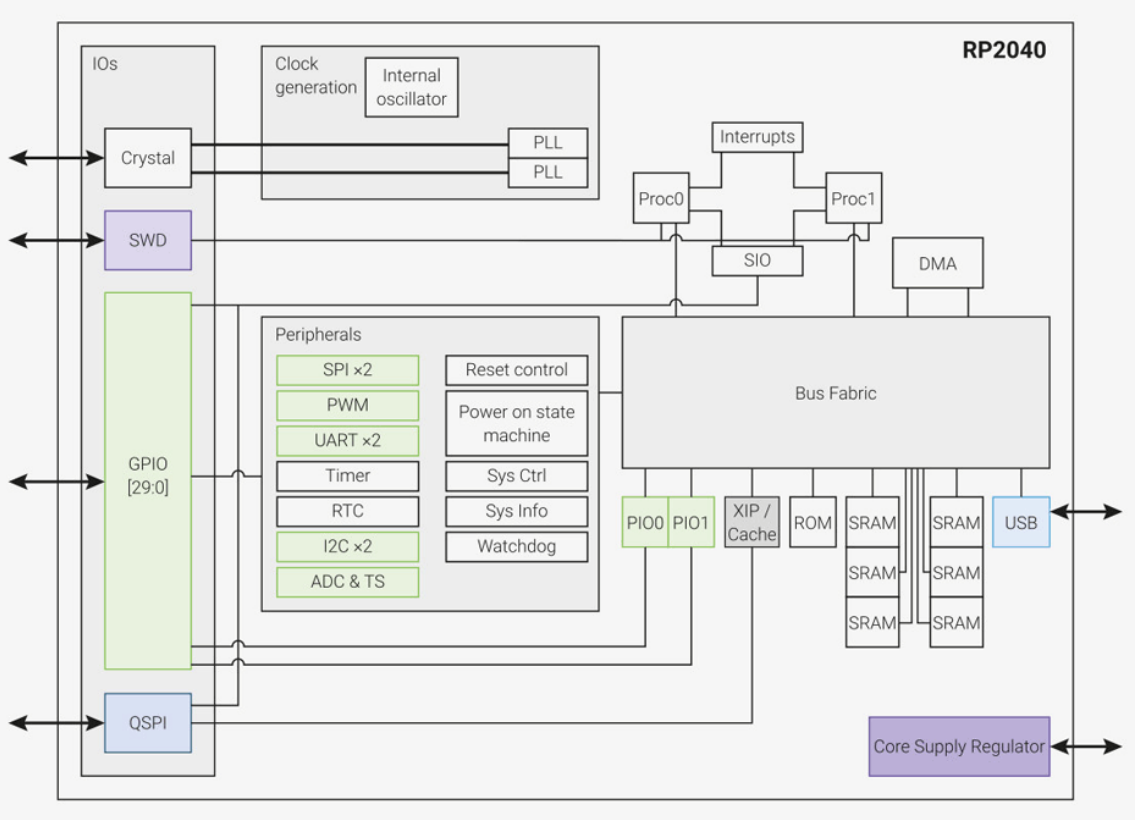


Рис. 2.3. Функціональна схема мікроконтролера RP2040

До складу мікроконтролера входять різноманітні блоки пам'яті, включаючи ROM, що містить базовий код для ініціалізації системи, та SRAM для зберігання тимчасових даних під час роботи. Інноваційний механізм XIP дозволяє виконувати код безпосередньо з Flash-пам'яті, підвищуючи загальну продуктивність за рахунок кешування найбільш часто використовуваних даних. 264 КБ внутрішньої SRAM пам'яті та 2 МБ зовнішньої Flash пам'яті в Pico W забезпечують достатньо простору для зберігання коду та даних [14]. Це важливо для розробників, які планують створювати складніші програми, що вимагають значної кількості зберігання.

Основна особливість Pico W - це його інтегрований Wi-Fi модуль. Завдяки підтримці стандарту 802.11b/g/n, цей мікроконтролер відкриває двері до широкого спектру IoT застосунків, включаючи віддалений збір даних, управління пристроями, та бездротову комунікацію [14]. Можливість підключення до інтернету розширює потенціал Pico W набагато далі, ніж це було можливо з оригінальним Pico.

З точки зору апаратного забезпечення, Pico W має 26 GPIO пінів, включаючи 3 аналогові входи. Це дозволяє підключати до Pico W широкий спектр сенсорів, актуаторів та інших пристроїв. Підтримка інтерфейсів, таких як I2C, SPI, та UART, додатково розширює можливості інтеграції Pico W з іншими пристроями та модулями.

Зовнішній кристал, що входить до системи генерації годинникових сигналів, забезпечує стабільність роботи системи, а фазовані петлі зчеплення (PLL) відіграють ключову роль у управлінні частотою годинника процесорів. Додатково, інтерфейс Serial Wire Debug (SWD) надає засоби для налагодження програми, а USB інтерфейс спрощує з'єднання мікроконтролера з зовнішніми пристроями.

Завершують структуру системні блоки управління, які відповідають за контроль стану живлення, перезапуску системи, нагляд за її станом через системний контролер та інформаційні реєстри, а також сторожовий таймер, що запобігає зависанню програми. Регулятор живлення ядра забезпечує надійність живлення внутрішніх компонентів.

Щодо програмування, Pico W підтримує мови програмування C/C++ та Python. Зокрема, підтримка MicroPython є важливою, оскільки вона дозволяє швидко та легко розпочати роботу з Pico W, використовуючи високорівневу мову програмування.

Компактні розміри Pico W (51 x 21 мм) роблять його ідеальним для використання в обмежених просторах або в складних проектах, де кожен міліметр має значення [14]. Це робить його особливо привабливим для розробників вбудованих систем, освітніх проектів.

2.2.2. Модуль GSM SIM800L. Це компактний модуль бездротового зв'язку, який забезпечує можливості GSM/GPRS у діапазонах 850/900/1800/1900 МГц, що робить його придатним для глобального мобільного зв'язку [15]. Він має LGA пакування, яке оптимізовано для зручної інтеграції в різні проекти та пристрої (див.рис.2.4).

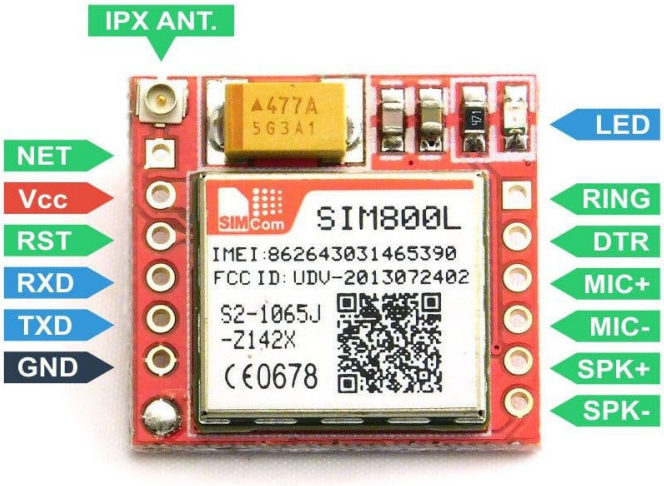


Рис. 2.4. Зовнішній вигляд модуля SIM800L

Ключовою особливістю SIM800L є його вбудована підтримка протоколу TCP/IP, яка дозволяє модулю підключатися до інтернету через GPRS [15]. Це робить його ідеальним для IoT застосунків, таких як смарт-трекери, розумне домашнє обладнання, та інші пристрої, що потребують віддаленого зв'язку.

Модуль має вбудовану підтримку для надсилання SMS-повідомлень та здійснення голосових викликів, що дає можливість створювати застосунки, які можуть взаємодіяти з користувачем через стандартні мобільні засоби комунікації.

Щодо розширених функцій, SIM800L підтримує передачу даних за допомогою GPRS класу 12, надаючи швидкості передачі до 85.6 kbps (див.рис.2.5).

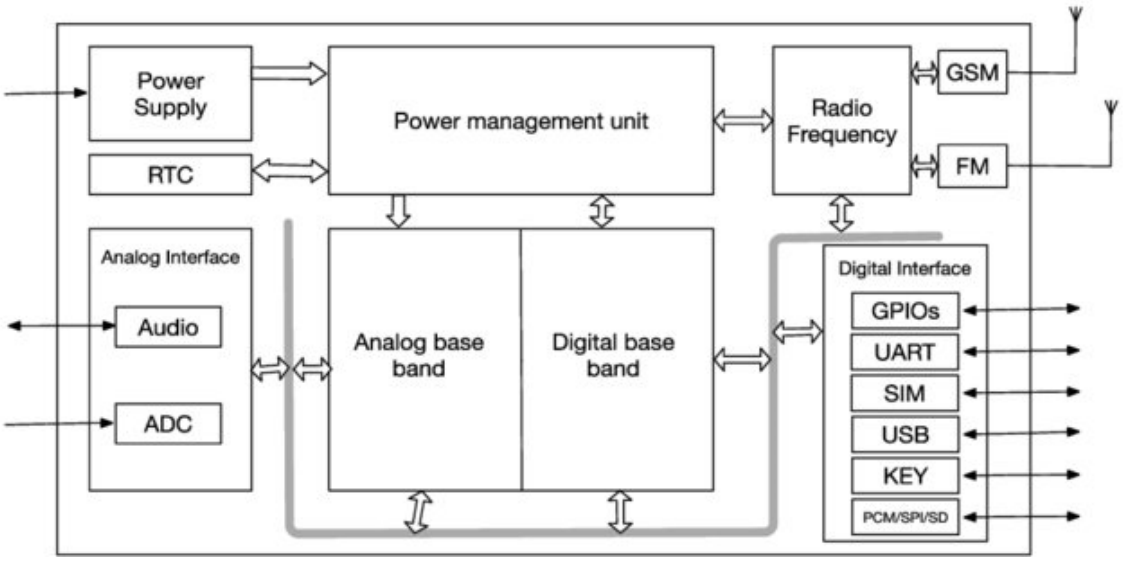


Рис. 2.5. Функціональна схема модуля SIM800L

Також модуль підтримує різні кодеки для здійснення голосових викликів та має ехокомпенсатор та шумопоглинач для покращення якості звуку.

З точки зору інтеграції з іншими пристроями, SIM800L має різні інтерфейси вводу-виводу, включаючи послідовний інтерфейс, інтерфейс для підключення SIM-карт, а також можливості підключення до аналогових аудіо пристроїв. Це дає можливість легко інтегрувати модуль у більш складні системи.

Модуль також характеризується низьким споживанням енергії у режимі очікування, що є важливим для портативних пристроїв та застосунків, які працюють від батареї [15].

Аналоговий базовий блок обробляє аналогові сигнали в модулі, а цифровий базовий блок відповідає за обробку цифрових сигналів. Обидва блоки підключені до блоку управління живленням.

З лівого боку до аналогового інтерфейсу підключено аудіо блок, який в свою чергу з'єднаний з АЦП (аналого-цифровий перетворювач), це говорить про те, що модуль може обробляти аудіо сигнали.

Справа знаходиться цифровий інтерфейс, який включає в себе ряд різних комунікаційних інтерфейсів, таких як загально призначені входи/виходи (GPIO), універсальний асинхронний приймач/передавач (UART), інтерфейс для SIM-

карти, універсальна послідовна шина (USB), інтерфейс для кнопок (KEY) та інші цифрові комунікаційні інтерфейси (PCM/SPI/SD) [15].

Стрілки на схемі вказують напрямки потоків живлення та сигналів між блоками, демонструючи, як вони пов'язані в межах модуля. Цей модуль призначений для інтеграції в широкий спектр застосунків телекомунікації та радіозв'язку.

2.2.3. Давач температури та вологості DHT11. Даний давач є пристроєм, призначеним для вимірювання відносної вологості та температури в різних умовах (див.рис.2.6). Цей давач забезпечує калібрований цифровий сигнал, що відображає точні показники вологості та температури. Для цього використовується технологія збору цифрових сигналів та технологія вимірювання вологості, які забезпечують надійність та стабільність давача.

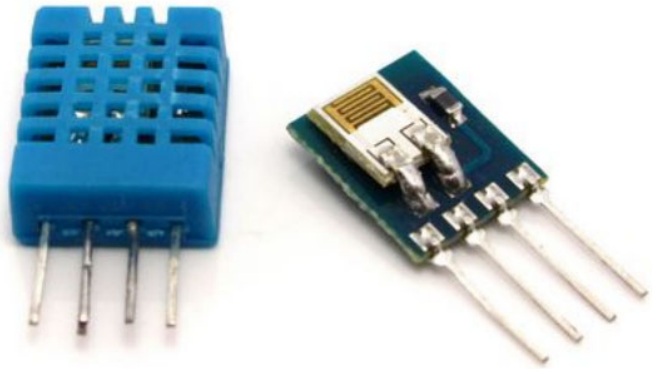


Рис. 2.6. Вигляд давач температури та вологості DHT11

Основним елементом вимірювання в DHT11 є полімерний резистор. Давач компенсує коливання температури та калібрується в спеціалізованій камері, а коефіцієнти калібрування зберігаються в пам'яті OTP. Це забезпечує високу точність показників: вологості в межах ±4%RH (максимум ±5%RH) і температури в межах ±2.0°C [16].

DHT11 працює від джерела живлення з напругою 3-5.5V DC і використовує цифровий сигнал для передачі даних на однопровідній шині. Давач має довгий термін служби та стабільність показників, забезпечуючи надійність у використанні.

У практичному застосуванні DHT11 підходить для різноманітних складних умов, зокрема завдяки далекій дистанції передачі сигналів (до 20 метрів), низькому споживанню енергії та зручності підключення завдяки чотирьом контактам [16]. Це робить його ідеальним вибором для інтеграції в різні системи автоматизації та моніторингу, включаючи мікрокомп'ютерні системи.

2.2.4. Давач струму SCT-013-000. Даний давач є неінвазивним пристроєм для вимірювання сили струму, здатним працювати в діапазоні від 0 до 100 ампер [17]. Особливістю цього пристрою є його безконтактний метод вимірювання, що представляє собою значне відмінність від більшості аналогічних пристроїв на ринку. Безконтактне вимірювання забезпечується завдяки тому, що силова частина пристрою не з'єднана електрично з вимірювальною (див.рис.2.7). Це підвищує безпеку використання давача, оскільки виключає ризик короткого замикання або пошкодження через прямий контакт з силовою ланцюгом.

Рис. 2.7. Давач струму SCT-013-000

Як основний вимірювальний елемент у SCT-013-000 використовується феритовий сердечник з обмоткою (див.рис.2.8). Через цей сердечник пропускається силовий провід, діаметр якого не повинен перевищувати 13 мм. Принцип роботи пристрою заснований на струмовому трансформаторі, де зміни струму в силовому проводі перетворюються в зміни вихідного сигналу давача. Рівень цього сигналу прямо пропорційний силі струму, що протікає через провід.

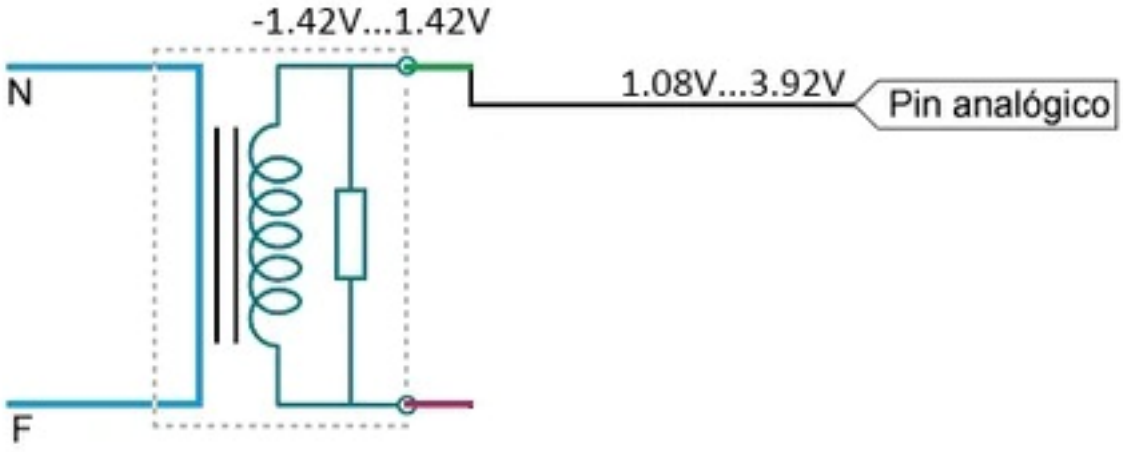


Рис. 2.8. Електрична схема давача SCT-013-000

Технічні характеристики давача включають діапазон вимірювань від 0 до 100 А та вихідний сигнал, який складає 50 мА на кожні 100 А струму. Розмір отвору, через який проходить силовий провід, складає 13х13 мм. Давач оснащений роз'ємом типу 3.5 мм (stereo jack) для підключення до вимірювального обладнання. Робочий діапазон температур, при якому гарантується нормальна робота пристрою, знаходиться в межах від -25 до +70°C. Розміри давача складають 32х22х57 мм, що робить його досить компактним для використання в різних електротехнічних застосуваннях [17].

2.2.5. Давач диму MQ-2. Давач є високочутливим сенсором, призначеним для виявлення різних газів, включаючи ЗПГ, і-бутан, пропан, метан, алкоголь, водень і дим (див.рис.2.9). Основною перевагою цього давача є його широкий діапазон виявлення разом зі швидкою відгукою та високою чутливістю, що робить його незамінним в обладнанні для виявлення витоків газу як у домашніх, так і промислових умовах.

Рис. 2.9. Давач диму MQ-2

На рис. 2.10 показана конструкція давача, яка включає керамічну трубку з мікро алюмінієвої оксиду, чутливий шар з двоокису олова (SnO2), вимірювальний електрод та нагрівач, що вмонтовані в корпус з пластику та нержавіючої сталі [18]. Нагрівач забезпечує необхідні умови для роботи чутливих компонентів. Обгорнутий MQ-2 має 6 контактів, чотири з яких використовуються для отримання сигналів, а два інших - для надання струму нагрівання.

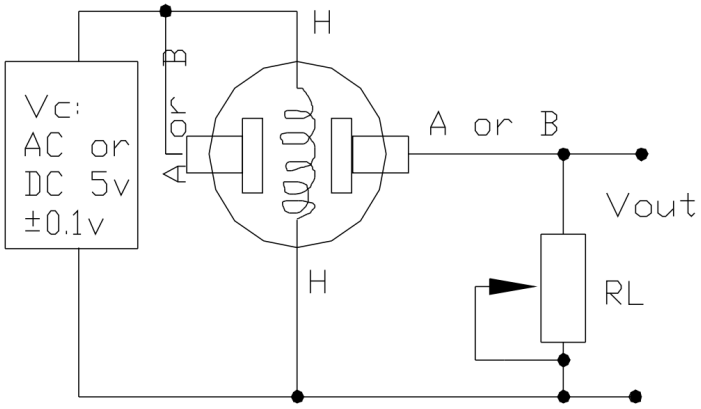


Рис. 2.10. Електрична схема давача MQ-2

Для ефективної роботи MQ-2 необхідно провести калібрування, оскільки опір давача варіюється в залежності від типу та концентрації газів. Рекомендується калібрувати детектор для 1000 ppm зрідженого нафтового газу (ЗНГ) або 1000 ppm і-бутану в повітрі та використовувати навантажувальний опір приблизно 20 кОм (від 5 кОм до 47 кОм) [18].

Особливості MQ-2 також включають його стабільність та довговічність, а також простоту використання в схемах. Це робить його ідеальним вибором для тривалого та надійного моніторингу в присутності газів та диму.

2.2.6. Давач наявності води FC-37 з модулем YL-38. Давач є простим інструментом для виявлення рідини, яка може використовуватися як перемикач (див.рис.2.11). Даний модуль складається з чутливої плати та окремої керуючої плати, яка забезпечує більшу зручність використання. Він включає в себе індикатор живлення LED та регульований потенціометр для налаштування чутливості.

Рис. 2.11. Давач FC-37 з модулем YL-38

Специфікації давача включають високоякісний двосторонній матеріал RF-04 з площею плати 5 см х 4 см з нікелевим покриттям з одного боку. Плата стійка до окислення. Вихід компаратора надає чіткий сигнал форми хвилі. Чутливість регулюється потенціометром, робоча напруга становить 5V. Формат виходу включає цифровий перемикач (0 та 1) та аналоговий вихід напруги AO. Модуль оснащений отворами для болтів для зручності монтажу та має маленьку розмірність плати PCB: 3.2 см х 1.4 см. Використовується широковольтний компаратор LM393. Конфігурація виводів включає VCC (5V постійного струму), GND (заземлення), DO (цифровий вихід) та AO (аналоговий вихід).

2.2.7. LCD-дисплея 1602 з I2C-інтерфейсом. Дисплей відрізняється компактністю та функціональністю, що робить його ідеальним для використання в широкому спектрі електронних проектів. Він забезпечує відображення текстової інформації на 2 рядках по 16 символів кожен, що дозволяє користувачам легко відслідковувати важливі дані або повідомлення.

Основні технічні характеристики цього дисплея включають його розміри (80мм х 36мм х 12мм) та вагу (близько 28 г), що робить його зручним для інтеграції в різноманітні пристрої (див. рис.2.12). Дисплей використовує LED

підсвітку, яка є регульованою, дозволяючи користувачам налаштувати інтенсивність світла відповідно до своїх потреб. Його робоча напруга варіюється від 4.5V до 5.5V, а споживана потужність залежить від рівня підсвітки та відображуваного вмісту.

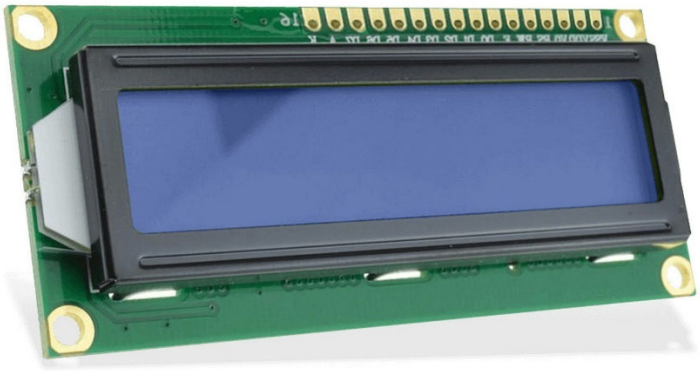


Рис. 2.12. LCD-дисплей 1602 з I2C-інтерфейсом

Принцип роботи LCD дисплея базується на технології рідких кристалів. Кожен символ на дисплеї складається з набору пікселів, які контролюються індивідуально, що дозволяє відтворювати чіткий та яскравий текст. Інтерфейс I2C, використовуваний у цьому дисплеї, забезпечує легке підключення до мікроконтролерів та інших пристроїв, дозволяючи передавати дані для відображення та керувати налаштуваннями дисплея.

LCD-дисплей 16x2 I2C знаходить застосування у різних сферах, включаючи системи мікроконтролерів, системи автоматизації, прототипування електронних пристроїв та навчальні комплекти. Його компактні розміри та простота інтеграції роблять цей дисплей популярним вибором для багатьох розробників.

2.2.8. Сервіс MQTT Mosquitto. Mosquitto є найпопулярніший брокер, який часто використовуються в інтернеті речей (IoT) через їхні унікальні характеристики, які виділяють їх серед інших протоколів передачі даних.

MQTT виділяється своєю легкістю та ефективністю, що робить його ідеальним для пристроїв з обмеженими ресурсами або ненадійними мережами. Це контрастує, наприклад, з HTTP, який може бути більш ресурсомістким і менш підходящим для деяких IoT сценаріїв. Іншою особливістю MQTT є publish/subscribe модель, яка дозволяє пристроям легко підписуватися на дані, які їм потрібні, і отримувати оновлення в реальному часі [19]. Це відрізняється від протоколу, як-от CoAP, який також орієнтований на IoT, але використовує більш традиційну клієнт-серверну модель [20].

Крім того, MQTT підтримує рівні якості обслуговування (QoS), що дозволяє розробникам балансувати між надійністю доставки та ресурсними витратами. Це відрізняється від протоколів, таких як WebSocket, які хоча й забезпечують швидке двостороннє спілкування, але можуть не мати такої гнучкості в управлінні якістю обслуговування [19].

Особливість Mosquitto як брокера MQTT полягає у його легкості та високій продуктивності, роблячи його вибором для багатьох IoT застосувань, де потрібна масштабованість та надійність. Це відрізняє його від більш складних брокерів, які можуть пропонувати додаткові функції, але за рахунок збільшення складності та ресурсних вимог.

2.2.9. Операційна система Ubuntu Server. Це високопродуктивна, стабільна та легка у використанні операційна система для серверів. Вона заснована на Debian Linux, одному з найстаріших і найбільш стабільних дистрибутивів Linux. Ubuntu Server включає в себе безліч популярних серверних програм, таких як Apache, MySQL, PHP, що робить його популярним вибором для вебсерверів [21].

Важливою особливістю Ubuntu Server є його простота управління та налаштування. Завдяки великій кількості документації та активному співтовариству, користувачі легко можуть знайти відповіді на свої запитання та

підтримку. Також, система має вбудований механізм для автоматизації та управління розгортанням, що спрощує управління ресурсами та застосунками.

Ubuntu Server відрізняється своєю безпекою. Завдяки регулярним оновленням та патчам безпеки, система забезпечує високий рівень захисту. Крім того, можливість легко інтегрувати різні мережеві інструменти безпеки забезпечує гнучкість у створенні надійних серверних рішень.

Щодо продуктивності, Ubuntu Server оптимізований для роботи на різних типах апаратного забезпечення, від невеликих приватних серверів до великих дата-центрів [21]. Система підтримує останні технології, включаючи контейнеризацію та хмарні обчислення, що робить її ідеальним рішенням для сучасних вимог до IT-інфраструктури.

Крім того, Ubuntu Server підтримує велику кількість мов програмування, що робить його універсальним вибором для розробників. Це також сприяє широкому використанню Ubuntu Server в різних галузях, від веброзробки до наукових досліджень.

2.3. Опис електричної принципової схеми

Електрична принципова схема для моніторингу параметрів серверної кімнати показна рис.2.13.

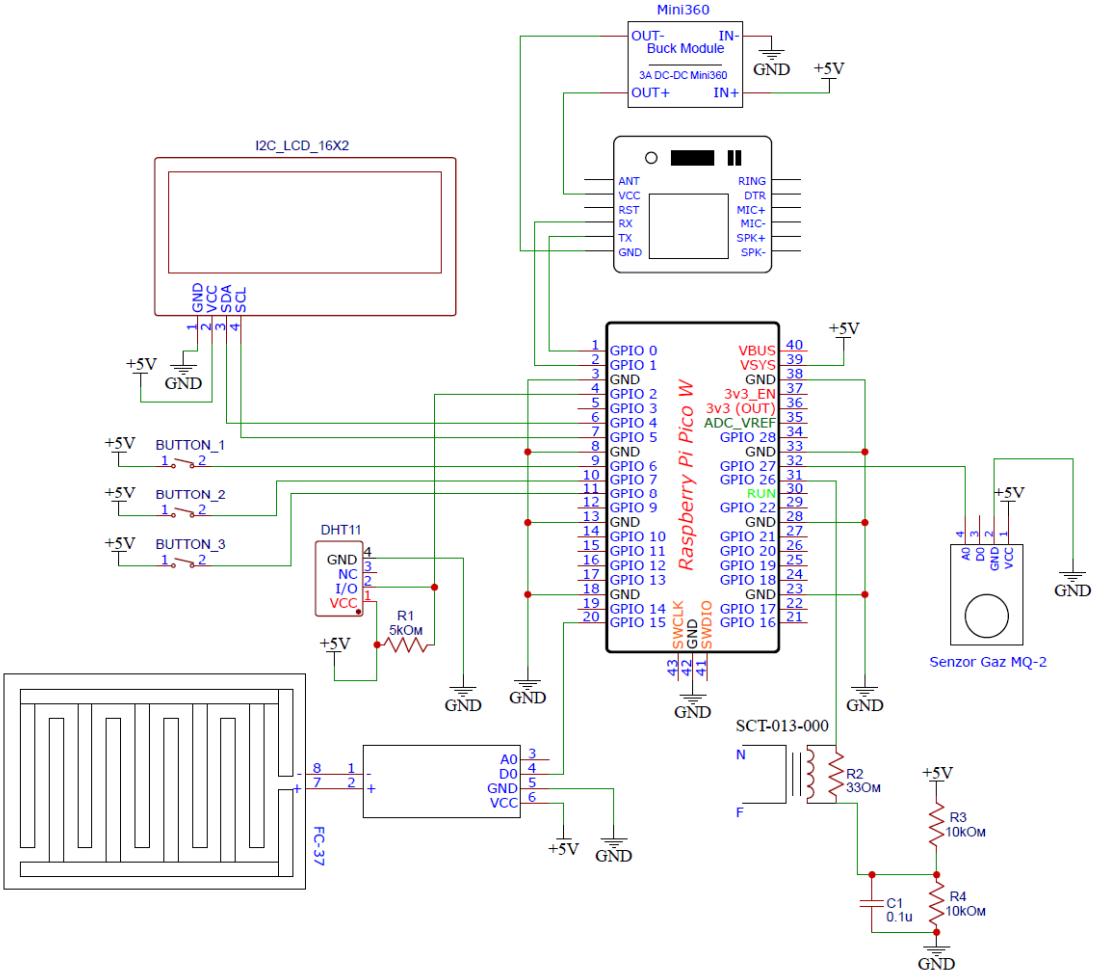


Рис. 2.13. Електрична принципова схема блоку моніторингу

У центрі схеми розташований мікроконтролер з численними виводами GPIO, які можуть бути налаштовані на прийом або відправлення сигналів. Верхня частина схеми містить модуль Mini360, який є понижувальним регулятором напруги і використовується спеціально для живлення модуля SIM800L, що підключений через інтерфейси з позначеннями RX, TX, VCC, GND та інші, які забезпечують зв'язок та живлення.

Три кнопки, позначені як BUTTON 1, BUTTON 2 та BUTTON 3, призначені для введення команд користувачем. Датчик DHT11, який включає чотири виводи, зокрема для живлення (VCC), заземлення (GND), а також вивід даних (I/O), розташований поруч з одним з резисторів.

На лівій стороні схеми розміщений I2C LCD дисплей з відповідними з'єднаннями для живлення та передачі даних. Додатково, є датчик газу MQ-2 та

датчик струму SCT-013-000, кожен з яких підключається до мікроконтролера через власні інтерфейси.

Схема також містить пасивні компоненти, такі як резистори R1, R2, R3, R4, деякі з яких утворюють дільники напруги або виконують інші функції, а також конденсатор C1, що використовується для фільтрації шумів напруги на вхідній лінії. З'єднання заземлення, позначені символом GND, є загальними для всіх компонентів, що забезпечує їхню належну роботу в одній системі.

2.4. Висновки до розділу

В даному розділі було проведено детальний опис ключових компонентів системи, таких як мікроконтролер Raspberry Pi Pico W, модулі SIM800L, DHT11, SCT-013-000, MQ-2, давач FC-37 з модулем YL-38, а також LCD-дисплей 1602 з I2C-інтерфейсом. Кожен з цих компонентів відіграє важливу роль у функціонуванні системи, забезпечуючи збір, обробку та відображення даних у різних умовах. Окрім апаратного забезпечення, було розглянуто застосування сервісу MQTT та операційної системи Ubuntu Server для системи моніторингу. Також було розроблено електричну принципову схему, що інтегрує всі ці компоненти в єдину систему моніторингу.

РОЗДІЛ 3
РОЗРОБКА СИСТЕМИ МОНІТОРИНГУ

3.1. Вибір мови програмування

Використання MicroPython для програмування Raspberry Pi Pico W пропонує унікальне поєднання легкості використання, швидкості розробки та гнучкості, що робить його ідеальним вибором для широкого спектру проектів. MicroPython, будучи спрощеною версією Python, відомої своєю читабельністю і простотою, дозволяє розробникам легко писати та тестувати код, значно скорочуючи час розробки [22]. Він підтримує багато бібліотек та модулів, забезпечуючи гнучкість для різних застосувань, а велика і активна спільнота Python надає суттєву підтримку.

Оптимізований для мінімізації споживання ресурсів мікроконтролерів, MicroPython є енергоефективним, забезпечуючи достатню продуктивність для більшості задач. Також, з урахуванням вбудованого Wi-Fi модуля Raspberry Pi Pico W, MicroPython легко інтегрується з мережевими можливостями, роблячи його відмінним вибором для Інтернету речей (IoT), де потрібна здатність до віддаленого збору даних, управління пристроями або інтеграція з хмарними сервісами [23].

Крім технічних переваг, MicroPython має великий освітній потенціал. Його простота і доступність роблять його чудовим інструментом для навчання основам програмування та роботи з електронікою. У сукупності ці фактори роблять MicroPython для Raspberry Pi Pico W вибором, який поєднує в собі гнучкість, простоту у використанні, доступність та потенціал для використання у різноманітних проектах – від освітніх до складних IoT-систем.

3.2. Програмне забезпечення для моніторингу

3.2.1. Налаштування MQTT. Проведемо налаштування MQTT broker для можливості отримання даних мікроконтролера Pico W. На рис.3.1 показано частину конфігураційного файлу MQTT.

```
/etc/mosquitto/conf.d/default.conf
allow_anonymous false
password_file /etc/mosquitto/passwd
listener 1883 0.0.0.0
```

Рис. 3.1. Конфігураційний файл MQTT

Фрагмент конфігурації містить наступні параметри:

- `allow_anonymous false` - анонімні підключення (тобто підключення без використання імені користувача та пароля) до MQTT брокера заборонені;

- `password_file /etc/mosquitto/passwd` - шлях до файлу паролів, де зберігається інформація про імена користувачів та їхні паролі;

- `listener 1883 0.0.0.0` — налаштовує MQTT брокера на прослуховування вхідних підключень на порті 1883, IP-адреса 0.0.0.0 означає, що брокер прийматиме підключення на всіх мережевих інтерфейсах сервера.

3.2.2. Програмний коду блоку моніторингу. Перед написання коду для Raspberry Pico W була розроблена блок-схема, яка описує алгоритм роботи цього коду. Вона міститься у додатку Б. Також у додатку В міститься увесь лістинг програмного коду блоку моніторингу.

На початковому етапі підключаємо усі необхідні бібліотеки до файлу програми (див.рис.3.2).

```
import math
import dht
import machine
from machine import Pin
import time
from umqtt.simple import MQTTClient
import network
from lcd_api import LcdApi
from pico_i2c_lcd import I2cLcd
```

Рис. 3.2. Лістинг коду підключених бібліотек

Цей блок імпорту в MicroPython включає в себе різноманітні бібліотеки та модулі, які необхідні для роботи з мікроконтролером, зазвичай у проектах IoT або для розробки вбудованих систем. Він містить математичні функції (`math`), інструменти для взаємодії з DHT-давачами температури та вологості (`dht`), засоби для керування апаратними компонентами мікроконтролера, як-от GPIO піни (`machine` і `Pin`). Також присутні функції для роботи з часом (`time`), підключення до мережі (`network`) та обміну повідомленнями через MQTT протокол (`umqtt.simple`). Додатково, імпортовано API та класи для управління LCD-дисплеями через I2C інтерфейс (`lcd_api` та `pico_i2c_lcd`). Всі ці імпорти потрібні для можливісті зчитування даних з давачів, відображення інформації, взаємодії з мережею та управлінням різними апаратними компонентами.

Після підключення бібліотек, було ініціалізовано LCD через I2C та UART для GSM-модуля (див.рис.3.3).

```
uart = machine.UART(0, baudrate=9600, tx=Pin(0), rx=Pin(1))

i2c = machine.I2C(0, sda=machine.Pin(4), scl=machine.Pin(5), freq=400000)
lcd = I2cLcd(machine.I2C(0, sda=machine.Pin(4),
                         scl=machine.Pin(5), freq=400000), I2C_ADDR, 2, 16)
```

Рис. 3.3. Лістинг коду для налаштування UART

Після ініціалізації інтерфейсів, було здійснено підключення до мережі Wi-Fi та виведено на LCD-екран IP-адресу отриману по DHCP (див.рис.3.4).

```
ssid = '[redacted]'
password = '[redacted]'
station = network.WLAN(network.STA_IF)
station.active(True)
station.connect(ssid, password)

while not station.isconnected():
    time.sleep(1)

print('Підключено до WiFi')
lcd.clear()
lcd.putstr("WiFi Connected")

lcd.move_to(0, 0)
lcd.putstr('IP:' + station.ifconfig()[0])
```

Рис. 3.4. Лістинг коду для підключення до Wi-Fi

На hис.3.5 показано фрагмент лістингу коду налаштування параметрів для підключення до MQTT сервера, включаючи адресу сервера, ідентифікатор клієнта та інші дані для аутентифікації (див.рис.3.5).

```
mqtt_server = '192.168.0.111'
client_id = '[redacted]'
username = '[redacted]'
password = '[redacted]'
client = MQTTClient(client_id, mqtt_server, user=username, password=password)
client.connect()
```

Рис. 3.5. Лістинг коду для підключення до MQTT сервера

В подальшому було ініціалізовано кнопки, давачі та встановлено коефіцієнт калібрування для давача SCT013-000 (див.рис.3.6).

```
button_temp = Pin(6, Pin.IN, Pin.PULL_DOWN)
button_hum = Pin(7, Pin.IN, Pin.PULL_DOWN)
button_irms = Pin(8, Pin.IN, Pin.PULL_DOWN)

dht_sensor = dht.DHT11(machine.Pin(2))
adc1 = machine.ADC(27)
adc2 = machine.ADC(26)
sensor_pin = Pin(15, Pin.IN)

CALIBRATION = 86
```

Рис. 3.6. Лістинг фрагменту коду параметрів для давача SCT013-000

Функція send_command призначена для відправки команд через UART інтерфейс і отримання відповіді від підключеного пристрою. Вона приймає команду як вхідний аргумент, до якої додає символи кінця рядка (b'\r\n') і відправляє це через UART. Потім ініціалізується змінна response для зберігання відповіді. У циклі безперервно читаються дані з UART, по одному байту за раз, і додаються до response. Як тільки у відповіді з’являється підстрока b'OK' або b'ERROR', що свідчить про завершення обробки команди або про помилку, цикл завершується і повна відповідь повертається. Ця функція дозволяє відправляти команди та отримувати відповіді, що є ключовим для взаємодії з різними пристроями через UART (див.рис.3.7).

```
def send_command(command):
    uart.write(command + b'\r\n')
    response = b''
    while True:
        data = uart.read(1)
        if data is not None:
            response += data
            if b'OK' in response or b'ERROR' in response:
                break
    return response
```

Рис. 3.7. Лістинг коду функції send_command

Функція send_sms використовується для відправлення SMS-повідомлень через GSM модуль. Вона використовує команди AT (Attention Commands),

специфічні для управління GSM модулями. AT+CMGF=1 налаштовує модуль на текстовий режим, AT+CMGS вказує номер телефону для відправлення, а потім відправляє текстове повідомлення. bytes([26]) відправляє символ завершення (CTRL+Z) для ініціації відправлення SMS (див.рис.3.8).

```
def send_sms(message):
    send_command(b'AT+CMGF=1')
    send_command(b'AT+CMGS="<[illegible]>"')
    send_command(message.encode('utf-8'))
    utime.sleep(1)
    uart.write(bytes([26]))
```

Рис. 3.8. Лістинг коду функції send_sms

На hис.3.9 показано лістинг функція, яка обчислює дійсну величину струму (Irms) на основі зразків з аналогового-цифрового перетворювача. Вона читає значення струму, перетворює їх у вольти, обчислює середньоквадратичне значення, а потім множить на калібрувальний коефіцієнт для отримання Irms.

```
def calc_Irms(samples):
    sum_of_squares = 0
    for _ in range(samples):
        value = adc2.read_u16()
        voltage = (value / 65535) * 3.3
        sum_of_squares += voltage ** 2
    mean_square = sum_of_squares / samples
    Irms = math.sqrt(mean_square) * CALIBRATION
    return Irms
```

Рис. 3.9. Лістинг коду функції cacl_Irms

Наступна функція перевіряє наявність інтернет-з'єднання, роблячи HTTP запит до Google.com (див.рис.3.10). Якщо отримується відповідь зі статус-кодом 200, це означає, що інтернет-з'єднання є активним. У разі помилки або відсутності відповіді протягом вказаного часу (timeout), функція повертає False, ідентифікуючи відсутність інтернету.

```
def is_internet():
    try:
        response = urequests.get('http://www.google.com', timeout=10)
        if response.status_code == 200:
            return True
    except:
        return False
```

Рис. 3.10. – Лістинг коду для перевірки з’єднання

Після підключення усіх необхідних бібліотек та створення функцій було створено основний цикл скрипта.

У сегменті коду представленому на Рис.3.11 виконується зчитування даних з різних давачів: давача води, температури та вологості, давача диму, та давача електричного струму. Функція calc_Irms обчислює середньоквадратичне значення струму, а потім розраховується споживану електричну потужність.

```
water_detected = sensor_pin.value()
dht_sensor.measure()
temp = dht_sensor.temperature()
hum = dht_sensor.humidity()
smoke = adc1.read_u16()
irms = calc_Irms(1480)
power = (irms * 220)/1000
```

Рис. 3.11. Лістинг коду для зчитування даних

В подальшому програма використовує умовні оператори для перевірки, чи була натиснута певна кнопка (див.рис.3.12). В залежності від натиснутої кнопки, на LCD-дисплей виводиться відповідна інформація: IP-адреса, температура, вологість або струм.

```
if button_temp.value() == 1:
    lcd.clear()
    lcd.move_to(0, 0)
    lcd.putstr('IP:' + station.ifconfig()[0])
    lcd.move_to(0, 1)
    lcd.putstr('Temperature:{}C'.format(temp))

if button_hum.value() == 1:
    lcd.clear()
    lcd.move_to(0, 0)
    lcd.putstr('IP:' + station.ifconfig()[0])
    lcd.move_to(0, 1)
    lcd.putstr('Humidity:{}%'.format(hum))

if button_irms.value() == 1:
    lcd.clear()
    lcd.move_to(0, 0)
    lcd.putstr('IP:' + station.ifconfig()[0])
    lcd.move_to(0, 1)
    lcd.putstr('I=: {:.2f} A'.format(Irms))
```

Рис. 3.12. Лістинг фрагменту коду використання операторів перевірки

Наступний блок коду відправляє дані з давачів на MQTT сервер та реагує на різні ситуації, такі як виявлення води, високий рівень диму (див.рис.3.13). Це дозволяє віддалено відстежувати стан сенсорів через MQTT-клієнт.

```
client.publish('dht11/temperature', str(temp))
client.publish('SCT/irms', str(irms))
client.publish('SCT/power', str(power))
client.publish('dht11/humidity', str(hum))

if water_detected == 0:
    client.publish('fc37/flooding', '1')
    print("Water detected: 1")
else:
    client.publish('fc37/flooding', '0')
    print("No water detected: 0")

if smoke <= 52000:
    client.publish('mq2/smoke', '0')
else:
    client.publish('mq2/smoke', '1')
```

Рис. 3.13. Лістинг коду для відправки даних з датчиків на сервер

У даному фрагменті програми виконується моніторинг стану мережевих з'єднань мікроконтролера: перевіряється підключення до WiFi та наявність інтернет-з'єднання (див.рис.3.14). У разі відсутності WiFi з'єднання, доступу до мережі Інтернет, електропостачання та при відсутності даних у буфері UART (модуль GSM готовий для відправлення даних), програма відправляє SMS-повідомлення про критичні ситуації. Це забезпечує можливість своєчасного сповіщення про відсутність необхідних мережевих з'єднань та електропостачання.

```
if not station.isconnected() and uart.any() == 0:
    send_sms('Немає з\'єднання з WiFi')

if not is_internet() and uart.any() == 0:
    send_sms('Немає з\'єднання з інтернетом')

if not irms < 0.5 and uart.any() == 0:
    send_sms('Струм відсутній')
```

Рис. 3.14. Лістинг коду для перевірки мережевих з’єднань

3.2.3. Програмний код системи сповіщення та зберігання. Перед написання програмного коду для системи сповіщення в Telegram-bot та системи зберігання статистики в базу даних MySQL була розроблена блок-схема, яка описує алгоритм роботи програмного коду, яка міститься у додатку Г. Сам програмний код міститься в додатку Д.

На рис. 3.15 показано фрагмент лістингу блок імпорту в Python бібліотек, які використовуються для різних аспектів роботи з мережевими та базами даних у програмі. Імпорт `mysql.connector` дозволяє підключатися та взаємодіяти з базами даних MySQL, що є корисним для зберігання та обробки даних. `paho.mqtt.client` використовується для роботи з MQTT протоколом. `datetime` дозволяє робити операції з датою та часом, що може бути необхідним для маркування даних часовими відмітками. `telebot` - це бібліотека для створення ботів в Telegram, що дозволяє використовувати бота як інтерфейс для взаємодії

з користувачами. `threading` використовується для створення паралельних потоків виконання, що дозволяє програмі виконувати кілька задач одночасно.

```
import mysql.connector
import paho.mqtt.client as mqtt
from datetime import datetime
import telebot
import threading
```

Рис. 3.15. Лістинг коду імпорту бібліотек

У наступному фрагменті коду визначаються конфігураційні параметри та змінні для програми, яка займається моніторингом та сповіщенням (див.рис.3.16). Конфігурація для підключення до бази даних MySQL включає інформацію про користувача, пароль, хост, назву бази даних. Налаштування для MQTT-брокера містять IP-адресу брокера, порт, ім'я користувача та пароль, а також список тем, які будуть використовуватися для підписки на повідомлення від давачів. Ініціалізація Telegram бота виконується за допомогою токена, що дозволяє програмі взаємодіяти з користувачами через Telegram.

```
config = {
    'user': '[redacted]',
    'password': '[redacted]',
    'host': '127.0.0.1',
    'database': 'server_room',
    'raise_on_warnings': True
}

mqtt_broker = "192.168.0.111"
mqtt_port = 1883
mqtt_username = "[redacted]"
mqtt_password = "[redacted]"
mqtt_topics = ["dht11/temperature",
               "dht11/humidity",
               "mq2/smoke",
               "SCT/irms",
               "SCT/power",
               "fc37/flooding"]

telegram_token = '[redacted]'
bot = telebot.TeleBot(telegram_token)
```

Рис. 3.16. Лістинг коду параметрів програми моніторингу

Цей фрагмент програмного коду представляє собою комплексну систему обробки даних і сповіщення (див.рис.3.17).

Ключовим елементом є обробка повідомлень MQTT, які включають інформацію від давачів, таких як задимлення (mq2/smoke), затоплення (fc37/flooding), вимірювання електричної потужності (SCT/power) та струму (SCT/irms), а також параметри температури та вологості (dht11). Для кожного типу повідомлення програма визначає відповідну логіку обробки.

```
topic = message.topic
msg_payload = str(message.payload.decode("utf-8"))

conn = mysql.connector.connect(**config)
cursor = conn.cursor()

now = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

if topic == 'mq2/smoke':
    smoke = float(msg_payload)
    if smoke == 0:
        smoke_alerts_sent = 0
    elif smoke_alerts_sent < 2:
        bot.send_message(chat_id='[illegible]', text=f'Увага: Задимлення')
        smoke_alerts_sent += 1
```

Рис. 3.17. Лістинг коду для системи обробки даних та сповіщень

У разі виявлення аномалій, таких як вищезгадане задимлення чи затоплення, система ініціює процедуру сповіщення через Telegram, використовуючи API Telegram бота. Це дозволяє оперативно інформувати відповідальних осіб про потенційні небезпеки. Ліміт сповіщень (в даному випадку, два) служить для запобігання надмірного повторення сповіщень (див.рис.3.18).

```
if 19 <= temperature <= 21:
    temperature_alerts_sent = 0
elif temperature_alerts_sent < 2:
    bot.send_message(chat_id='[illegible]',
                     text=f'Увага: Температура виходить за межі! Значення: {temperature}°C')
    temperature_alerts_sent += 1
```

Рис. 3.18. Лістинг коду для виявлення відхилень показників моніторингу

Для даних, що стосуються електричної потужності, струму, температури та вологості, програма виконує запис у базу даних з використанням SQL-запитів (див.рис.3.19). Це забезпечує зберігання статистичних даних для подальшого аналізу та відстеження тенденції. Часові мітки (зафіксовані за допомогою модуля datetime) відіграють ключову роль у забезпеченні точності та відстеження хронології подій.

```
elif topic == 'SCT/power':
    power = float(msg_payload)
    insert_query = "INSERT INTO power (power, recorded_at) VALUES (%s, %s)"
    cursor.execute(insert_query, (msg_payload, now))
    conn.commit()
```

Рис. 3.19. Лістинг коду для запису даних в базу даних.

Також у коді включено розгалужену систему обробки винятків для управління помилками при взаємодії з базою даних MySQL, що є важливим для стабільної роботи системи (див.рис.3.20). При виявленні помилок у процесі роботи з базою даних або при виникненні інших виняткових ситуацій, програма здійснює запис помилок для подальшого аналізу та вирішення проблем.

```
except mysql.connector.Error as err:
    print(f"Database error: {err}")
except Exception as e:
    print(f"Error: {e}")
```

Рис. 3.20. Лістинг коду для виявлення помилок

У наступному фрагменті коду реалізується встановлення з'єднання з MQTT брокером для обміну повідомленнями у системах Інтернету речей (IoT) (див.рис.3.21).

Ініціалізація клієнта MQTT включає в себе встановлення параметрів аутентифікації, що є важливим для забезпечення безпеки з'єднання. Подальша прив'язка обробників подій до клієнта MQTT дозволяє визначити специфічні реакції на події підключення до брокера та отримання повідомлень. Це забезпечує гнучкість у реагуванні на зміни у мережі та обробці вхідних даних.

Підключення до MQTT брокера відбувається шляхом вказання IP-адреси та порту, з установленням часу очікування відповіді. Останнім кроком є підписка на визначені теми, які відображають категорії повідомлень, на які система має реагувати. Це включає в себе теми, що пов'язані з вимірюваннями давачів та іншими значущими подіями в системі.

```
client = mqtt.Client()
client.username_pw_set(mqtt_username, mqtt_password)

client.on_connect = on_connect
client.on_message = on_message

client.connect(mqtt_broker, mqtt_port, 60)

for topic in mqtt_topics:
    client.subscribe(topic)
```

Рис. 3.21. Лістинг коду для встановлення з'єднання з MQTT

На рис.3.22 показано фрагмент лістингу коду, який реалізовує інтерфейс користувача через Telegram бота для системи моніторингу параметрів у серверній кімнаті. Використання бота в Telegram дозволяє користувачам інтерактивно запитувати статус системи та отримувати актуальну інформацію про умови у серверній, такі як температура, вологість, потужність і струм.

Функція send_welcome, активована командою /start, надає користувачам вступне повідомлення з інструкціями щодо використання бота. Це забезпечує простоту взаємодії з системою для користувачів, які можуть не бути знайомі з деталями її роботи.

Функція send_stats, активована командою /stats, забирає останні доступні дані про стан серверної кімнати, формує з цих даних повідомлення і відправляє його користувачу. Це дозволяє користувачам швидко отримувати оновлену інформацію про критичні параметри, що спостерігаються у серверній кімнаті, забезпечуючи своєчасне реагування на можливі проблеми або зміни умов.

Використання окремого потоку для опитування бота забезпечує безперервну роботу бота без блокування основного потоку програми. Це важливо для забезпечення стабільності та відповідності роботи системи, оскільки запобігає можливим затримкам у обробці інших завдань, пов'язаних з моніторингом.

```
@bot.message_handler(commands=['start'])
def send_welcome(message):
    bot.reply_to(message, "Привіт! Я бот для моніторингу умов у серверній. Щоб переглянути

@bot.message_handler(commands=['stats'])
def send_stats(message):
    stats_message = "Останні параметри:\n"
    if temperature is not None:
        stats_message += f"Температура: {temperature}°C\n"
    if humidity is not None:
        stats_message += f"Вологість: {humidity}%\n"
    if power is not None:
        stats_message += f"Напруга: {power:.2f} kW*h\n"
    if irms is not None:
        stats_message += f"Струм: {irms:.2f} A"
    if temperature is None and humidity is None and irms is None and power is None:
        stats_message = "Параметри зараз недоступна."
    bot.reply_to(message, stats_message)

threading.Thread(target=bot.polling).start()
```

Рис. 3.22. Лістинг коду для виведення даних в інтерфейс користувача

3.3. Налаштування додаткових сервісів

У системі дані за допомогою скрипта зберігаються у базі даних MySQL (див.рис.3.23). Кожні нові дані зберігаються з параметром часу, для накопичення статистика.

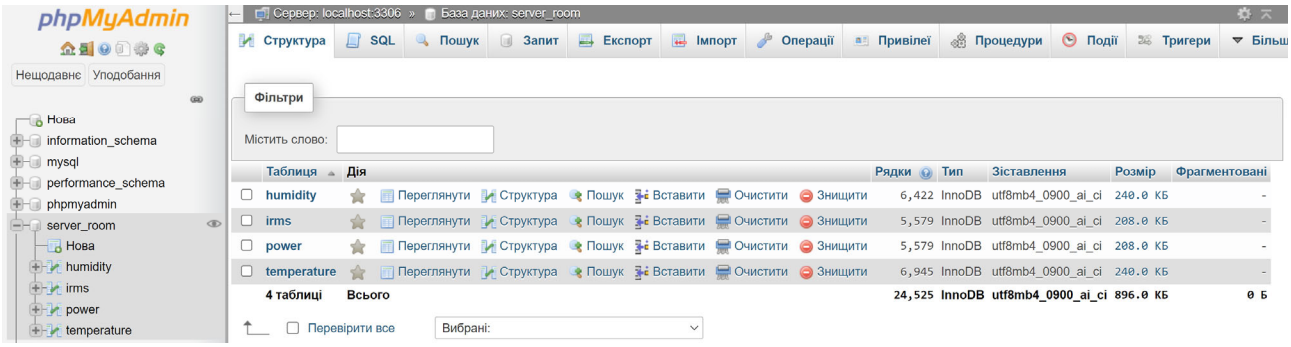


Рис. 3.23. База даних MySQL

Рисунок демонструє інтерфейс phpMyAdmin, який використовується для адміністрування баз даних MySQL. На ньому відображено базу даних під назвою server_room, що містить кілька таблиць, орієнтованих на зберігання моніторингових даних для середовища серверної кімнати.

В структурі бази даних таблиці humidity та temperature призначені для реєстрації показників вологості та температури відповідно. Ці таблиці містять часові мітки та числові значення показників, зібрані з датчиків у серверній кімнаті.

Таблиця power містить статистичні дані системи моніторингу живлення, що відіграє критичну роль у запобіганні відмовам у роботі через проблеми з електропостачанням.

Для створення сервісу для автоматичного запуску скрипта моніторингу на Ubuntu Server було використано систему ініціалізації systemd (див.рис.3.24).

```
/etc/systemd/system/mqtt_db.service   [-M--]  0 L:[  1+17  18/ 18] *(207 / 207b) <EOF>        [*][X]
[Unit]

Description=MQTT_DB_telebot_script

After=network.target

[Service]

ExecStart=/usr/bin/python3 /home/administrator/python_mqtt_db/mqtt_db.py

Restart=always

[Install]

WantedBy=multi-user.target
```

Рис. 3.24. Сервіс моніторингу

Блок Unit містить опис служби та вказує, що служба повинна бути запущена після старту мережі. Service визначає команду, яка виконуватиметься при старті служби. Зокрема, вона вказує на скрипт Python. Також встановлено, що служба повинна автоматично перезапускатися у разі збою. Блок Install вказує, що службу слід автоматично встановлювати як частину системи systemd, що означає, що вона буде запущена при загрузці системи для різних користувачів.

3.4. Практична реалізація системи моніторингу

Для можливості монтажу блоку моніторингу в серверній було створено модель корпусу у програмі SolidWorks з подальшим друком на 3D-принтері

(див.рис.3.30). Ця модель була створена з урахуванням всіх технічних вимог і специфікацій, необхідних для забезпечення ефективного та безпроблемного встановлення.

SolidWorks є передовим програмним забезпеченням для CAD та CAE, розробленим компанією Dassault Systèmes. Це інструмент високої точності, призначений для професіоналів у галузі інженерії та дизайну, який забезпечує інтегроване рішення для 3D моделювання, аналізу, симуляції та управління даними про продукти. SolidWorks відомий своїм інтуїтивним інтерфейсом, ефективністю у вирішенні складних інженерних задач та широким функціоналом, який дозволяє користувачам створювати деталізовані 3D-моделі.

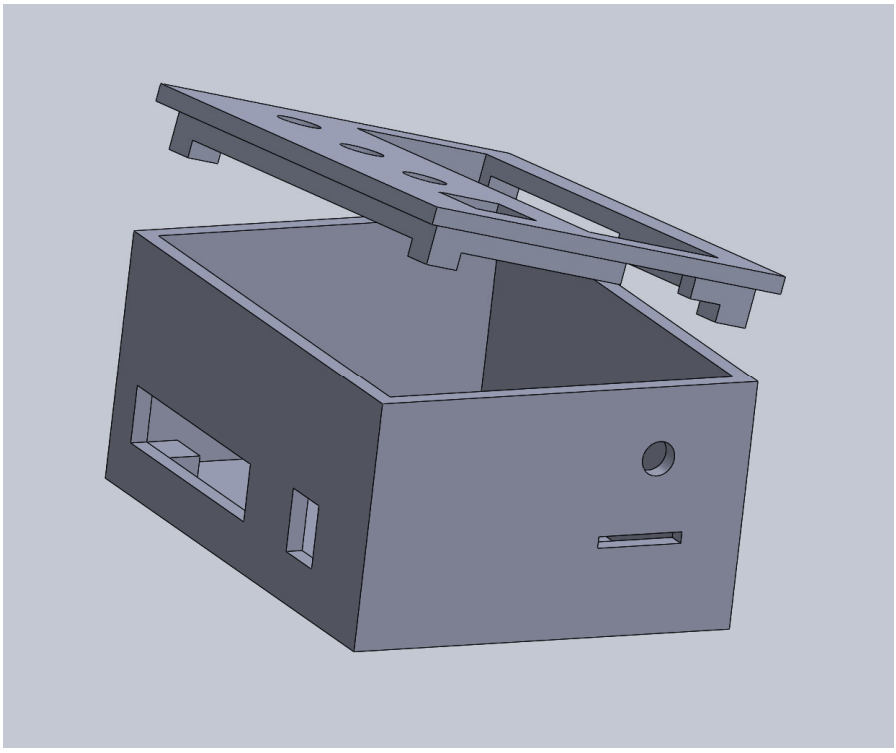


Рис. 3.25. Модель корпусу в SolidWorks

На рис.3.26 зображено блок моніторингу з усіма компонентами змонтованими в корпусі та кришці.

Рис. 3.26. Блок моніторингу в корпусу

На рис.3.27 показано змонтовану систему моніторингу в серверній кімнаті. Система введена в експлуатацію та здійснює моніторинг запрограмованих параметрів мікроклімату серверної кімнати.

Рис. 3.27. Змонтований блок моніторингу в сверній

3.5. Тестування системи моніторингу

Для ефективної перевірки системи оповіщень було земульовано різні критичні ситуації, починаючи з втрати зв'язку із мережею Інтернет, щоб переконатися, що система належно реагує. Також проведено тест на втрату WiFi-сигналу, щоб перевірити надійність оповіщень у цій ситуації. Окрім цього, земульовано відсутність електричної мережі у серверній кімнаті. Тести показали, що система оповіщень через GSM модуль працює надійно, надсилає SMS-повідомлення у випадках критичних збоїв, тим самим забезпечуючи надійний моніторинг стану мережі та електропостачання.

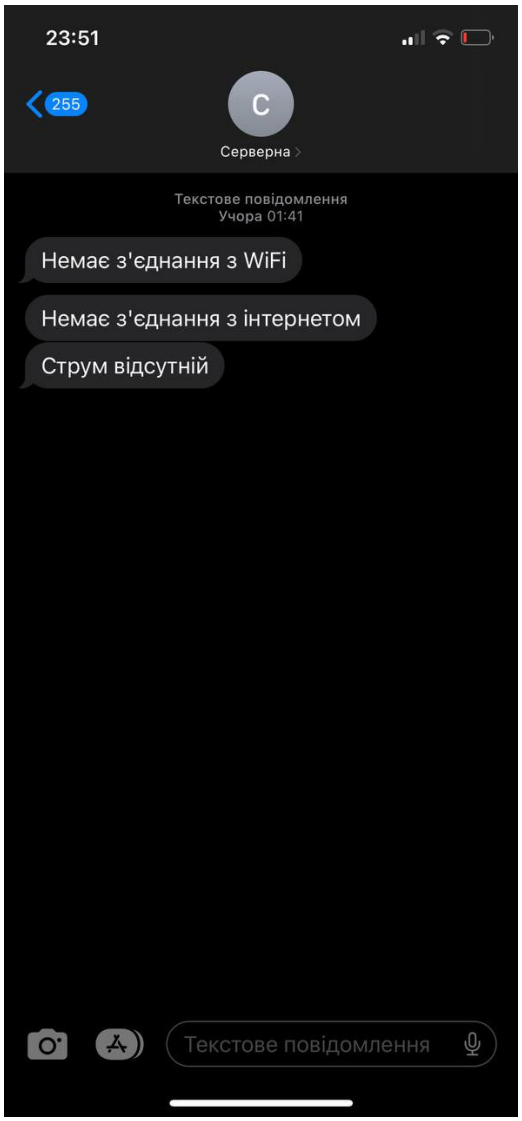


Рис. 3.28. Приклад SMS сповіщення

Для перевірки системи оповіщення через Telegram-бот, зімітувано критичні ситуації, які відображені на рис. 3.29. Перш за все, було зімітувано зростання температури та вологості в серверній кімнаті, щоб перевірити, як система реагує на такі зміни умов довкілля і як це відображається в повідомленнях Telegram-bot. Далі було проведено симуляцію виявлення диму та затоплення у серверній кімнаті. Це дозволило оцінити ефективність системи оповіщення у випадках потенційно небезпечних ситуацій, а також переконатися, що в Telegram-bot надсилються відповідні повідомлення про екстрені події.

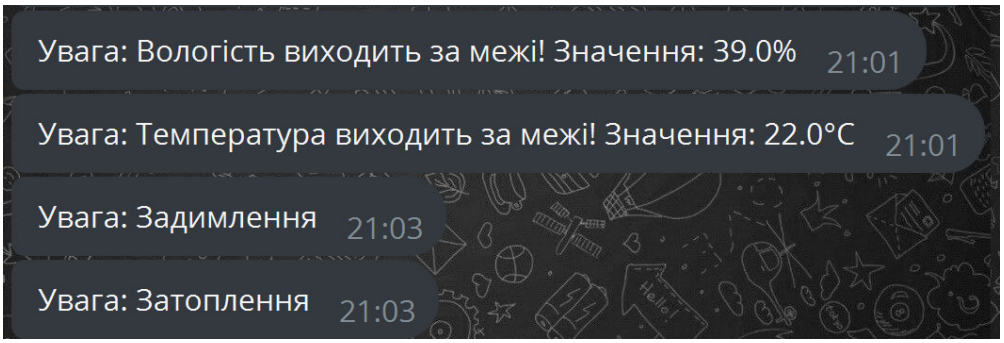


Рис. 3.29. Оповіщення через telegram-бот

Система моніторингу мікроклімату серверної кімнати успішно пройшла всі етапи тестування, підтверджуючи свою надійну роботу.

3.6. Висновки до розділу

Під час розробки комп'ютерної системи моніторингу мікроклімату серверної кімнати було розроблено програмне забезпечення на мові програмування MicroPython для мікроконтролера Raspberry Pi Pico W. Встановлено та налаштовано операційну систему Ubuntu Server та сервіс MQTT. Розроблено програмний коду для системи сповіщення в Telegram-bot та системи зберігання статистики в базу даних MySQL на мові програмування Python. Для можливості монтажу блоку моніторингу в серверній було створено модель корпусу у програмі SolidWorks з подальшим друком на 3D-принтері.

Проведено тестування системи, яке підтвердило що всі елементи працюють надійно. Реалізована система ефективно відстежує параметри мікроклімату, такі як температура, вологість, наявність диму чи затоплення, а також параметри електричного струму, що надає комплексне розуміння стану серверної кімнати.

РОЗДІЛ 4
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Охорона праці

У кваліфікаційній роботі магістра спроектовано комп’ютерну систему моніторингу мікроклімату серверної кімнати. Під час розв’язання задач дослідження, особливо практичної реалізації системи, потрібно врахувати вимоги з охорони праці і техніки безпеки, пожежної та електробезпеки.

Виконання як теоретичної частини роботи, так і практичної, передбачає використання комп’ютерної техніки та обладнання з низькими напругами і силою струму. Зокрема в блоці живлення плати використовувалась напруга живлення 5В. На платі використовуються можливі номінали напруги на рівні 5 В і 3,3 В, що не становить небезпеки для користувачів та розробника системи.

Охорона праці - це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності [24].

Нормативними документами, що забезпечують охорону праці при роботі з ЕОМ є:

– НПАОП 0.00-7.15-18 “Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями” [25];

– ДСанПіН 3.3.2.007-98 “Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин” [26];

– Закон України “Про охорону праці”.

Середовище праці передбачає створення комфортних умов у серверній кімнаті, яка має бути сухою, добре освітленою, із світлими стінами та стелею та покриттям підлоги лінолеумом. Мікроклімат повинен відповідати встановленим параметрам, які включають температуру в межах +22-24°С, відносну вологість

між 40-60%, та швидкість руху повітря приблизно 0.1 м/с, що забезпечується ефективною вентиляційною системою.

Розміщення та площа кімнати мають відповідати нормам, забезпечуючи достатній простір для обладнання та комфорт працівників. Це включає мінімальну площу 6.0 м² та об'єм 20.0 м³ на робоче місце, при цьому забороняється використання підвальних та цокольних приміщень. Шум та вібрація мають бути контрольованими, щоб не перевищувати допустимі норми, а серверна кімната не повинна межувати з приміщеннями, де ці показники вищі. Електричне обладнання має включати електророзетки з контактами для підключення нульового захисного провідника, а підключення обладнання до двопровідної мережі є неприпустимим. Пожежна безпека передбачає наявність системи автоматичної пожежної сигналізації, включаючи димові та теплові пожежні сповіщувачі, а також вуглекислотні вогнегасники з розрахунку 1 на 20 м², але не менше двох на приміщення [27].

В даному розділі роботи було розглянуто основні нормативні документи та положення з охорони праці, які регулюють умови праці, використання комп’ютерних систем.

4.2. Безпека в надзвичайних ситуаціях

4.2.1. Фактори ризику і можливі порушення здоров’я користувачів комп’ютерної мережі.

Велика кількість людей проводить багато часу за комп’ютером, що збільшує ризики та можливі негативні наслідки для здоров’я. Довге сидіння за комп’ютером призводить до різних проблем із здоров’ям. Найпоширенішими причин погіршення стану здоров’я є порушення постави та біль у спині [28]. Неправильна постава, яка зумовлена внаслідок погано організованого робочого місця або неправильно підібраного комп’ютерного стільця, це може спричинити хронічні болі у спині, шиї та плечах. Також поширена проблема із зап’ястям та кистями, наприклад синдром зап’ястного каналу, який виникає через

повторюванні рухи, наприклад використання миші чи клавіатури [28]. Ще одним серйозним ризиком, пов’язаним із тривалим використанням комп’ютера, є проблеми зі здоров’ям очей. Проведення надто тривалого часу перед екраном може призвести до таких симптомів, як погіршенням зору або напруження очей, яке характеризується сухістю, свербінням, подразненням і втомою очей. Робота за комп'ютером часто пов'язана з високим рівнем стресу, особливо в умовах коли терміни стислі, а вимоги високі. Це може призвести до вигоряння, депресії та інших психологічних проблем. Крім того, ізоляція, яка часто супроводжує роботу вдома або в невеликих кабінетах, може вплинути на соціальне та емоційне самопочуття.

Також важливо розглянути вплив на якість сну та загальне фізичне здоров'я. Багато користувачів комп'ютерів скаржаться на порушення сну, що може бути пов'язано з надмірним впливом на світлові екрани перед сном. Це світло пригнічує вироблення мелатоніну, гормону сну, заважаючи людям засинати та підтримувати здоровий цикл сну. Використання спеціальних програм або окулярів для блокування синього світла є важливим для тих, хто регулярно працює з комп'ютерами ввечері. Фізична неактивність, яка часто супроводжує довготривалу роботу за комп'ютером, також є фактором ризику для ряду хронічних захворювань, серцево-судинні захворювання та діабет другого типу. Недостатня фізична активність призводить до уповільнення метаболізму, що може спричинити набір ваги і зниження загальної фізичної форми. Це підвищує ризик розвитку серцевих захворювань.

Залежність від Інтернету та соціальних медіа є ще однією проблемою, з якою стикаються користувачі комп'ютерних мереж. Ця залежність може призвести до зниження продуктивності, соціальної ізоляції та погіршення психічного здоров'я. Важливо встановлювати межі та регулярно проводити час без використання електронних пристроїв.

Для зменшення цих ризиків існують різні стратегії. По-перше, важливо правильно організувати робоче місце. Ергономічні крісла, належно розташовані монітори та клавіатури, підставка для ніг, регулярні перерви для руху та

розтяжки можуть значно знизити ризик мускул скелетних проблем. Крім того, використання спеціальних окулярів або застосування програм, що знижують синє світло від екранів, можуть допомогти зменшити втому очей [29].

Що стосується психологічного здоров'я, важливо забезпечити баланс між роботою та особистим життям, включаючи регулярні перерви, хобі, соціальну взаємодію та вправи на свіжому повітрі. Також корисною може бути практика медитації для зниження рівня стресу.

4.2.2. Джерела, зони дії та рівні забруднення навколишнього середовища у разі аварій на хімічно і радіаційно небезпечних об’єктах. Джерела та причини аварій на хімічно та радіаційно небезпечних об'єктах різноманітні та часто пов'язані з комплексом чинників, включаючи людський фактор, технічні несправності, недостатність безпекових заходів та навіть природні катастрофи. Розглянемо приклади, які демонструють ці аварії.

Аварія на Чорнобильській АЕС (1986 рік) – одна з найбільш катастрофічних ядерних аварій у світовій історії. Ця аварія сталася через людські помилки та недоліків у справності реактора. Під час проведення експерименту з безпеки, який включав вимкнення деяких систем безпеки, стався вибух, що призвів до викиду радіаційних матеріалів у атмосферу [30].

Витік метилізоціанату в Бхопалі, Індія (1984 рік) – цей інцидент став однією з найбільших хімічних катастроф у світі. Витік стався внаслідок води, яка потрапила у резервуар з метилізоціанатом, що призвело до хімічної реакції та викиду токсичного газу. Причиною інциденту стали недоліки в управлінні безпекою та технічному обслуговуванні на заводі [31].

Аварія на АЕС Фукусіма-Даїчі (2011 рік) – приклад аварії, викликаної природним катаклізмом. Масштабне землетрус та наступний цунамі призвели до втрати зовнішнього живлення та відмови систем охолодження реакторів, що спричинило серйозні пошкодження та радіоактивні викиди [32].

Вибух на хімічному заводі в Тяньцзіні, Китай (2015 рік) – стався внаслідок неправильного зберігання великої кількості небезпечних хімічних речовин.

Вибух спричинив значні руйнування та забруднення навколишнього середовища [33].

При аваріях на атомних електростанціях, таких як Чорнобиль або Фукусіма, радіаційне забруднення може поширюватися на сотні та навіть тисячі кілометрів від місця інциденту. Радіоактивні частинки, підняті у повітря, можуть бути рознесені вітром на великі відстані, що призводить до широкомасштабного забруднення атмосфери, ґрунтів, водойм та живих організмів. Характер такого розповсюдження залежить від багатьох факторів, включаючи інтенсивність викидів, погодні умови та топографію місцевості [34].

Випадок хімічного забруднення, як у Бхопалі, Індії, показує, як витік токсичних речовин може мати негайні та руйнівні наслідки для навколишнього середовища та здоров'я людей. В таких випадках, забруднювачі можуть швидко розповсюджуватися повітрям та водою, впливаючи на великі території. Хімічні речовини можуть контамінувати ґрунт, підземні та поверхневі води, що призводить до довгострокових екологічних наслідків. Зона дії забруднення також залежить від швидкості реагування на аварію. Ефективні заходи реагування, такі як ізоляція вогнища аварії, можуть обмежити розповсюдження забруднювачів. Проте, в багатьох випадках, особливо при великих аваріях або недостатньому реагуванні, забруднення може поширитися на значні території, ускладнюючи процес відновлення та очищення.

Рівні забруднення на небезпечних об'єктах класифікуються за серйозністю впливу на довкілля та здоров'я людей. Ця класифікація допомагає у визначенні необхідних заходів реагування та стратегій відновлення.

Низький рівень забруднення:

– характеризується мінімальними концентраціями забруднюючих речовин, які не перевищують встановлені норми безпеки;

– вплив на довкілля та здоров'я людей є мінімальним або відсутнім;

– зазвичай не вимагає спеціальних заходів реагування, але потребує моніторингу для виявлення можливих змін.

Середній рівень забруднення:

– рівень забруднення перевищує норми безпеки, але не досягає критично небезпечного рівня;

– може мати короткотермінові негативні наслідки для довкілля, такі як зниження якості води, забруднення ґрунтів, шкоду для диких тварин;

– для людей можуть виникнути тимчасові проблеми зі здоров'ям, особливо в разі прямого контакту з забрудненими речовинами;

– вимагає заходів по локалізації забруднення та відновленню забруднених територій.

Високий рівень забруднення:

– характеризується значним перевищенням норм безпеки;

– має серйозні негативні наслідки для довкілля, такі як масова загибель тварин та рослин, забруднення водойм, руйнування екосистем;

– для людей високий ризик розвитку серйозних захворювань, включаючи отруєння, хронічні захворювання, збільшення випадків раку в разі радіаційного забруднення;

– потребує негайних заходів по евакуації населення, локалізації забруднення та тривалого відновлення екосистем.

Критичний рівень забруднення:

– найбільш небезпечний рівень, що включає екстремально високі концентрації токсичних або радіоактивних речовин;

– зона забруднення може бути оголошена непридатною для життя на тривалий час;

– довкілля може зазнати незворотних змін, з надзвичайно довгим періодом відновлення;

– для людей критичний рівень забруднення несе безпосередню загрозу життю та здоров'ю;

– вимагає довгострокових стратегій очищення та реабілітації, а також масштабних екологічних та охоронних проектів [35].

У даному розділі було проведено аналіз факторів ризику, зон впливу та рівнів забруднення, які виникають у випадку аварій на хімічно і радіаційно небезпечних об’єктах. Розглянуто різні сценарії аварій, які включають як людські помилки, так і природні катастрофи, та їх вплив на довкілля та здоров’я людей. Забруднення, викликане такими аваріями, може масштабні наслідки, поширюючись на значні території та призводячи до серйозних екологічних та проблем зі здоров’ям.

4.3. Висновки до розділу

У цьому розділі було детально досліджено важливі аспекти охорони праці та безпеки при монтажі та експлуатації комп'ютерних систем. Крім того, розглянуто питання безпеки в надзвичайних ситуаціях, а саме ризики та негативні наслідки для здоров’я, пов'язані з тривалим використанням комп'ютерів, та методи їх зменшення. Також проаналізовано приклади аварій на хімічно та радіаційно небезпечних об'єктах, їх причини, наслідки та методи реагування на такі ситуації.

ВИСНОВКИ

У підсумку виконаних теоретичних та практичних досліджень були отримані такі результати:

- проведено огляд та порівняння методів існуючих систем моніторингу серверних кімнат;

- досліджено можливості мікроконтролерів ESP8266, Raspberry Pi Pico та Arduino для забезпечення ефективності в розробленні системи моніторингу;

- розроблено схему роботи системи моніторингу для подальшої реалізації;

- розглянуто компоненти, які входять до системи моніторингу;

- створена та описана архітектуру системи моніторингу для сервених кімнат;

- реалізовано систему моніторингу, що базується на модулі Raspberry Pi Pico W та давачах струму, температури та вологості, диму, затоплення;

- розроблено telegram-бот для оповіщення про критичні ситуації в параметрах моніторингу;

- розроблено та налаштовано систему SMS сповіщення про критичні ситуації зі з доступом до мережі Інтернет та електропостачанням;

- підтверджено що система здатна ефективно відстежувати стан мікроклімату серверних кімнат та в випадках відхилень параметрів від норми проводити оповіщення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Developing Low Cost Server Room Monitoring and Control Device. URL: https://www.researchgate.net/publication/319838725_Developing_Low_Cost_Server_Room_Monitoring_and_Control_Device (дата звернення: 16.12.2023).

2. How tempCube Works?. URL: https://tempcube.io/pages/how-tempcube-works (дата звернення: 16.12.2023).

3. Requirements for a flexible and scalable Automatic Server Room Environmental Conditions Monitoring and Control system. URL: https://zenodo.org/records/1284899 (дата звернення: 16.12.2023).

4. IoT Remote Monitoring Solutions with Wireless Sensors. URL: https://www.monnit.com/applications/remote-monitoring/ (дата звернення: 16.12.2023).

5. Vertiv Geist™ Environmental Monitoring. URL: https://www.vertiv.com/en-us/products/brands/geist/vertiv-geist-environmental-monitoring/ (дата звернення: 16.12.2023).

6. AVTECH - Monitor Temperature and Environment Conditions with Room Alert. URL: https://avtech.com/ (дата звернення: 16.12.2023).

7. ESP8266 Wi-Fi SoC | Espressif Systems. URL: https://www.espressif.com/en/products/socs/esp8266 (дата звернення: 16.12.2023).

8. Raspberry Pi Documentation - Raspberry Pi Pico and Pico W. URL: https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html (дата звернення: 16.12.2023).

9. Raspberry Pi Pico microcontroller: specifications, features and RP2040 — The MagPi magazine. URL: https://magpi.raspberrypi.com/articles/raspberry-pi-pico-microcontroller-specifications-features-and-rp2040 (дата звернення: 16.12.2023).

10. Arduino Uno – Wikipedia. URL: https://en.wikipedia.org/wiki/Arduino_Uno (дата звернення: 16.12.2023).

11. UNO R3 | Arduino Documentation. URL: https://docs.arduino.cc/hardware/uno-rev3 (дата звернення: 16.12.2023).

12. Тимощук В.Д, Козуб М.В, Тимощук Д.І. Аналогові та цифрові ситеми: аналіз та практичне застосування. Теорія модернізації в контексті сучасної світової науки: матеріали І Міжнародної наукової конференції, (Полтава, 23 червня 2023 р.), Вінниця: Європейська наукова платформа, 2023. С 168–169.

13. Василишин В. В., Тимощук В. Д., Кітчак Н. Ю, Луцик Н.С. Аналіз характеристик та застосування мікроконтролерів ATTINY85, ATMEGA8, RP2040. Актуальні задачі сучасних технологій: збірник тез доповідей XII міжнародної науково-практичної конференції молодих учених та студентів (Тернопіль, 6–7 грудня 2023 року), Тернопіль: ТНТУ, 2023. С. 420.

14. RP2040 Datasheet. URL: https://datasheets.raspberrypi.com /rp2040/rp2040-datasheet.pdf (дата звернення: 16.12.2023).

15. Datasheet_SIM800L. URL: https://arduino.ua/files/ Datasheet_SIM800L.pdf (дата звернення: 16.12.2023).

16. Digital-output relative humidity & temperature sensor/module. URL: https://arduino.ua/docs/DHT11.pdf (дата звернення: 16.12.2023).

17. SCT-013-030-XiDiTechnology. URL: https://arduino.ua/files/SCT-013-030-XiDiTechnology.pdf (дата звернення: 16.12.2023).

18. TECHNICAL DATA MQ-2 GAS SENSOR. URL: https://arduino.ua/docs/MQ-2.pdf (дата звернення: 16.12.2023).

19. Documentation | Eclipse Mosquitto. URL: https://mosquitto.org/documentation/ (дата звернення: 16.12.2023).

20. Тимощук В.Д., Василишин В.В., Мудрий І.В., Луцик Н.С. Огляд та порівняння протоколів передачі інформації в IoT. Матеріали XI науково-технічної конференції "Інформаційні моделі, системи та технології" Тернопільського національного технічного університету імені Івана Пулюя (Тернопіль, 13-14 грудня 2023 року). Тернопіль: ТНТУ. 2023. С. 188.

21. Ubuntu Server documentation. URL: https://ubuntu.com/server/docs (дата звернення: 16.12.2023).

22. Python Documentation. URL: https://docs.python.org/uk/3/ (дата звернення: 16.12.2023).

23. MicroPython latest documentation. URL: https://docs.micropython.org/en/latest/ (дата звернення: 16.12.2023).

24. Закон України Про охорону праці Відомості Верховної Ради України (ВВР), № 49, 1992. 668 с.

25. Наказ міністерство соціальної політики України № 207 Про затвердження вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями від 14.02.2018.

26. ДСАНПІН 3.3.2.007-98: Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин №7 від 10.12.98.

27. Лупенко С.А., Луцик Н.С., Луцків А.М., Осухівська Г.М., Тиш Є.В. Методичні вказівки до виконання кваліфікаційної роботи магістра для студентів спеціальності 123 «Комп'ютерна інженерія» другого (магістерського) рівня вищої освіти усіх форм навчання. Тернопіль, ТНТУ. 2021. 34 с..

28. Шкідливий вплив персонального комп'ютера на здоров'я людини. URL: https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/21370/5363.pdf (дата звернення: 16.12.2023).

29. Як захиститися від синього світла на телефоні та ПК: простий алгоритм. URL: https://my.ua/news/cluster/2023-05-03-iak-zakhistitisia-vid-sinogo-svitla-na-telefoni-ta-pk-prostii-algoritm (дата звернення: 16.12.2023).

30. Барановська Н. П. Соціальні та економічні наслідки Чорнобильської катастрофи. К. 2001. 20 с.

31. Bhopal 1984 Disaster: A Gas Leak Tragedy and its Effects on the People. URL: https://www.jetir.org/papers/JETIREW06090.pdf (дата звернення: 16.12.2023).

32. Fukushima disaster: What happened at the nuclear plant? - BBC News. URL: https://www.bbc.com/news/world-asia-56252695 (дата звернення: 16.12.2023).

33. Case study of the Tianjin accident: Application of barrier and systems analysis to understand challenges to industry loss prevention in emerging economies. URL: https://minerva.jrc.ec.europa.eu/en/shorturl/minerva/acceptedmanuscriptcase_ study _of_the_tianjin_accident_20190817pdf (дата звернення: 16.12.2023).

34. Стручок В.С. Техноекологія та цивільна безпека. Частина «Цивільна безпека». Навчальний посібник. Тернопіль: ТНТУ. 2022. 150 с.

35. Exposure to outdoor air pollution and its human health outcomes: A scoping review. URL: https://journals.plos.org/plosone/article?id=10.1371/ journal.pone. 0216550 (дата звернення: 16.12.2023).

ДОДАТКИ

Додаток А Тези конференцій

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

XI НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

13-14 грудня 2023 року

ТЕРНОПІЛЬ
2023

УДК 004.738.5:004.7
В. Тимощук, В. Василишин, І. Мудрий, Н. Луцик доктор філософії, доцент
(Тернопільський національний технічний університет імені Івана Пулюя)

ОГЛЯД ТА ПОРІВНЯННЯ ПРОТОКОЛІВ ПЕРЕДАЧІ ІНФОРМАЦІЇ В IOT

V. Tymoshchuk, V. Vasylyshyn, I. Mudryi , N. Lutsyk Ph.D, Assoc. Prof.

OVERVIEW AND COMPARISON OF INFORMATION TRANSFER PROTOCOLS IN IOT

В епоху Індустріального Інтернету речей (IIoT), вибір відповідного протоколу передачі даних відіграє критичну роль у забезпеченні ефективності, безпеки та надійності систем. Сучасний вибір протоколів IIoT є різноманітним, кожен з яких пропонує унікальні переваги та має власні обмеження, залежно від конкретних вимог застосування.

MQTT протокол, що використовує модель "publisher-subscriber", оптимізований для мінімізації мережевого трафіку та енергоспоживання, що робить його ідеальним для пристроїв з обмеженими ресурсами. На відміну від MQTT, CoAP, спеціально розроблений для мереж з обмеженими ресурсами, підтримує швидкий обмін даними, використовуючи UDP, хоча й має меншу надійність.

В контексті промислових застосувань, OPC UA виступає як надійний стандарт для обміну даними, забезпечуючи високу безпеку та надійність передачі даних. Хоча цей протокол вимагає значних ресурсів для реалізації, його можливості в області безпеки та інтеграції роблять його важливим вибором для промислових систем.

Modbus, будучи одним з найстаріших протоколів автоматизації, зберігає свою популярність завдяки простоті та надійності. Він підходить для застосувань, де необхідна базова функціональність без складних вимог до безпеки.

AMQP, що зосереджений на обміні повідомленнями в корпоративних системах, пропонує високу надійність та гнучкість. Хоча він вважається більш складним і ресурсномістким, його можливості в області масштабованості та надійності роблять його оптимальним рішенням для великих та складних систем.

DDS, з іншого боку, надає швидкий та гнучкий механізм розподілу даних, ідеально підходить для систем, де вимоги до продуктивності та масштабованості є пріоритетними. Він забезпечує високий рівень налаштування якості обслуговування та є надійним рішенням для складних промислових застосувань.

Отже, вибір протоколу в IIoT залежить від балансу між потребами в енергоефективності, швидкості передачі даних, безпеці та інтеграції з іншими системами. Ефективне використання цих протоколів вимагає глибокого розуміння їх характеристик та вимог до системи, щоб оптимізувати обмін даними та підвищити загальну продуктивність IIoT-систем.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний технічний університет імені Івана Пулюя (Україна)
Університет імені П'єра і Марії Кюрі (Франція)
Маріборський університет (Словенія)
Технічний університет у Кошице (Словаччина)
Вільнюський технічний університет ім. Гедимінаса (Литва)
Міжнародний університет цивільної авіації (Марокко)
Наукове товариство ім. Т.Шевченка

АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ

Збірник

тез доповідей

XII Міжнародної науково-практичної конференції молодих учених та студентів

6-7 грудня 2023 року

УКРАЇНА
ТЕРНОПІЛЬ – 2023

Матеріали XII Міжнародної науково-практичної конференції молодих учених та студентів «АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ» – Тернопіль, 6-7 грудня 2023 року

УДК 621.382

В. В. Василишин; В. Д. Тимощук; Н. Ю. Кітчак;
Н. С. Луцик, доктор філософії, доцент
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

АНАЛІЗ ХАРАКТЕРИСТИК ТА ЗАСТОСУВАННЯ МІКРОКОНТРОЛЕРІВ ATTINY85, ATMEGA8, RP2040

V. Vasylyshyn, V. Tymoshchuk, N. Kitchak, N. Lutsyk Ph.D, Assoc. Prof.

ANALYSIS OF CHARACTERISTICS AND APPLICATION OF MICROCONTROLLERS ATTINY85, ATMEGA8, RP2040

Для оцінки мікроконтролерів Attiny85, Atmega8, і RP2040, розглянемо їх архітектурні особливості, ресурси, функціональність та енергоспоживання, визначаючи їх застосування у різноманітних проектах.

Attiny85, розроблений Microchip Technology, базується на AVR архітектурі і характеризується низьким споживанням енергії. Його основні обмеження — це кількість GPIO-виводів і 8 КБ пам'яті Flash. Працюючи в діапазоні напруги від 2.7 до 5.5 В, він досягає максимальної частоти 8 МГц. Attiny85 ідеально підходить для простих завдань, де ключовими є компактність і енергоефективність.

Atmega8, також від Microchip Technology, є 8-бітним мікроконтролером на базі AVR. Цей мікроконтролер має схожі робочі параметри з Attiny85, але з більшою частотою до 16 МГц. Це розширює його функціональні можливості, дозволяючи використовувати його у більш складних проектах, які потребують додаткових периферійних пристроїв, таких як UART або аналогово-цифрові конвертери.

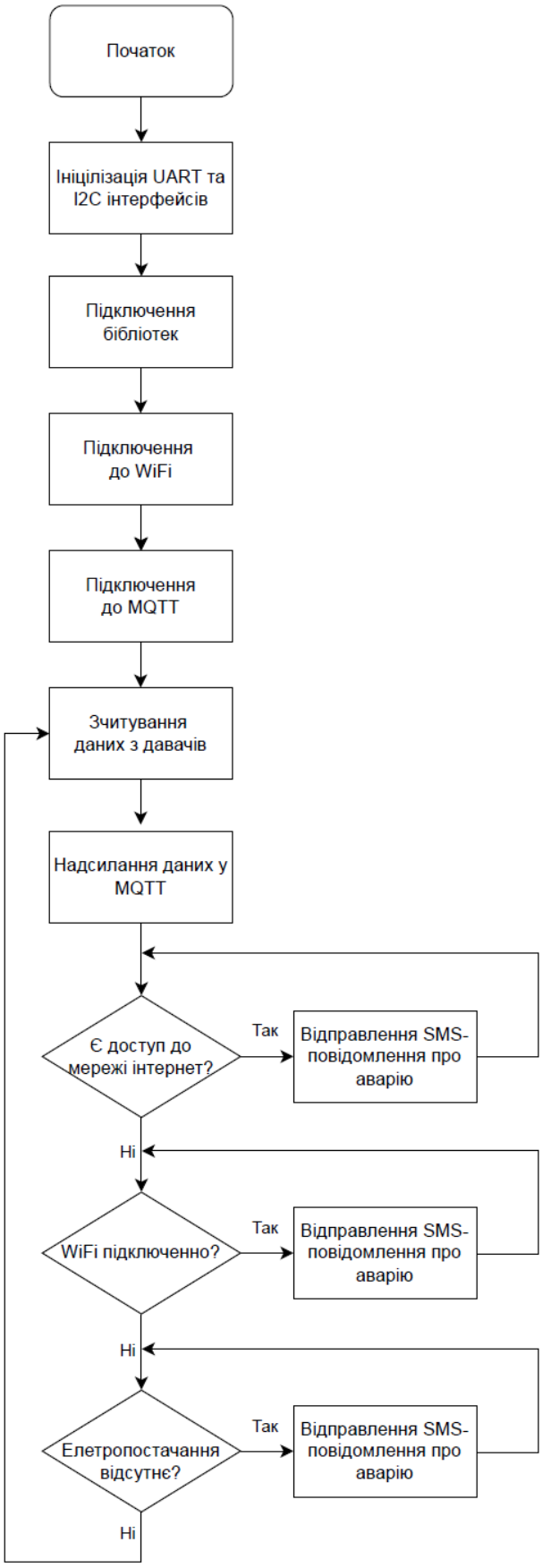
У порівнянні, RP2040 від компанії Raspberry Pi, працюючий на ARM Cortex-M0+ архітектурі, є значно потужнішим. З 16 МБ пам'яті Flash і 264 КБ SRAM, цей мікроконтролер досягає частоти до 133 МГц і підтримує різноманітні інтерфейси, у тому числі USB. Це робить його відмінним вибором для складних проектів у сфері IoT або робототехніки, де потрібні висока обчислювальна потужність і розширені комунікаційні можливості.

Висновок. Отже Attiny85, Atmega8 та RP2040 є важливими гравцями у світі мікроконтролерів, кожен з яких служить різним потребам в інженерії та дизайні. Attiny85, розроблений Microchip Technology, є особливо ефективним у сценаріях, де потрібні малі габарити та енергоефективність, в той час як Atmega8, що також належить до лінійки Microchip, вирізняється своєю здатністю впоратися з трохи більш складними завданнями, завдяки вищій частоті роботи. Натомість, RP2040 від Raspberry Pi, працює на потужнішій ARM Cortex-M0+ архітектурі, забезпечуючи значно більшу обчислювальну потужність та підтримку розширених комунікаційних інтерфейсів, що робить його ідеальним для складних проектів в сферах, таких як IoT або робототехніка. Ці три мікроконтролери, хоча і різняться за характеристиками та потенційними сферами застосування, кожен вносить унікальний вклад у розробку електроніки та автоматизації.

Література

1. ATTINY85 [Електронний ресурс]. — URL: https://www.microchip.com/en-us/product/ATtiny85#document-table (дата звернення: 27.11.2023).
2. ATmega8 [Електронний ресурс]. — URL: https://www.microchip.com/en-us/product/ATmega8#document-table (дата звернення: 27.11.2023).
3. RP2040 [Електронний ресурс]. — URL: https://www.raspberrypi.com/products/rp2040/specifications/ (дата звернення: 27.11.2023).

Додаток Б Алгоритм роботи скрипта на Raspberry Pi Pico W



Додаток В Лістинг скрипта для Raspberry Pi Pico W

```
import math
import dht
import machine
from machine import Pin
import time
from umqtt.simple import MQTTClient
import network
from lcd_api import LcdApi
from pico_i2c_lcd import I2cLcd

I2C_ADDR = 0x27
I2C_NUM_ROWS = 2
I2C_NUM_COLS = 16

uart = machine.UART(0, baudrate=9600, tx=Pin(0), rx=Pin(1))

i2c = machine.I2C(0, sda=machine.Pin(4), scl=machine.Pin(5),
freq=400000)
lcd = I2cLcd(machine.I2C(0, sda=machine.Pin(4), scl=machine.Pin(5),
freq=400000), I2C_ADDR, 2, 16)

ssid = ''
password = ''
station = network.WLAN(network.STA_IF)
station.active(True)
station.connect(ssid, password)

while not station.isconnected():
    time.sleep(1)

print('Підключено до WiFi')
lcd.clear()
lcd.putstr("WiFi Connected")
```

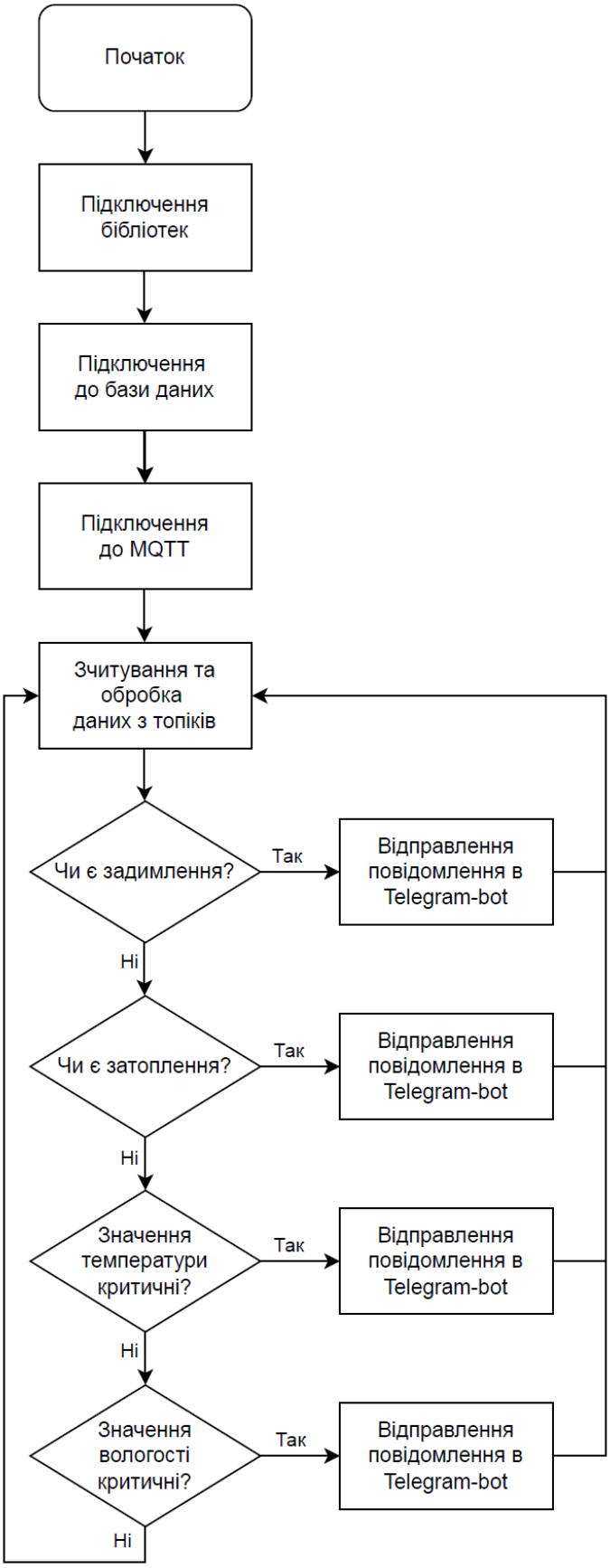
```
lcd.move_to(0, 0)
lcd.putstr('IP:' + station.ifconfig()[0])

mqtt_server = '192.168.0.111'
client_id = '****'
username = '****'
password = '****'
client = MQTTClient(client_id, mqtt_server, user=username,
password=password)
client.connect()
print('Підключено до MQTT сервера')

button_temp = Pin(6, Pin.IN, Pin.PULL_DOWN)
button_hum = Pin(7, Pin.IN, Pin.PULL_DOWN)
button_irms = Pin(8, Pin.IN, Pin.PULL_DOWN)

dht_sensor = dht.DHT11(machine.Pin(2))
adc1 = machine.ADC(27)
adc2 = machine.ADC(26)
sensor_pin = Pin(15, Pin.IN)

CALIBRATION = 86

def send_command(command):
    uart.write(command + b'\r\n')
    response = b''
    while True:
        data = uart.read(1)
        if data is not None:
            response += data
            if b'OK' in response or b'ERROR' in response:
                break
    return response
```

```
def send_sms(message):
    send_command(b'AT+CMGF=1')
    send_command(b'AT+CMGS="<>"')
    send_command(message.encode('utf-8'))
    utime.sleep(1)
    uart.write(bytes([26]))

def calc_Irms(samples):
    sum_of_squares = 0
    for _ in range(samples):
        value = adc2.read_u16()
        voltage = (value / 65535) * 3.3
        sum_of_squares += voltage ** 2
    mean_square = sum_of_squares / samples
    Irms = math.sqrt(mean_square) * CALIBRATION
    return Irms

def is_internet():
    try:
        response      =      urequests.get('http://www.google.com',
timeout=10)
        if response.status_code == 200:
            return True
    except:
        return False


while True:
    try:
        water_detected = sensor_pin.value()
        dht_sensor.measure()
        temp = dht_sensor.temperature()
        hum = dht_sensor.humidity()
        smoke = adc1.read_u16()
        irms = calc_Irms(1480)
```

```
power = (irms * 220)/1000


if button_temp.value() == 1:
    lcd.clear()
    lcd.move_to(0, 0)
    lcd.putstr('IP:' + station.ifconfig()[0])
    lcd.move_to(0, 1)
    lcd.putstr('Temperature:{}C'.format(temp))

if button_hum.value() == 1:
    lcd.clear()
    lcd.move_to(0, 0)
    lcd.putstr('IP:' + station.ifconfig()[0])
    lcd.move_to(0, 1)
    lcd.putstr('Humidity:{}%'.format(hum))

if button_irms.value() == 1:
    lcd.clear()
    lcd.move_to(0, 0)
    lcd.putstr('IP:' + station.ifconfig()[0])
    lcd.move_to(0, 1)
    lcd.putstr('I=: {:.2f} A'.format(Irms))


client.publish('dht11/temperature', str(temp))
client.publish('SCT/irms', str(irms))
client.publish('SCT/power', str(power))
client.publish('dht11/humidity', str(hum))

if water_detected == 0:
    client.publish('fc37/flooding', '1')
    print("Water detected: 1")
else:
    client.publish('fc37/flooding', '0')
```

```
            print("No water detected: 0")

        if smoke <= 52000:
            client.publish('mq2/smoke', '0')
        else:
            client.publish('mq2/smoke', '1')

        if not station.isconnected() and uart.any() == 0:
            send_sms('Немає з\'єднання з WiFi')

        if not is_internet() and uart.any() == 0:
            send_sms('Немає з\'єднання з інтернетом')

        if not irms < 0.5 and uart.any() == 0:
            send_sms('Струм відсутній')

        print('Temperature:', temp, 'C')
        print('Humidity:', hum, '%')
        print('Smoke Level:', smoke)
        print(f"W= {power:.2f} kW*h, I= {irms:.2f} A")
    except OSError as e:
        print('Не вдалося прочитати з DHT11')

    time.sleep(1)
```

Додаток Г Алгоритм системи сповіщення в Telegram-bot та системи зберігання статистики в базу даних



Додаток Д Лістинг скрипта для системи сповіщення в Telegram-bot та системи зберігання статистики в базу даних

```
import mysql.connector
import paho.mqtt.client as mqtt
from datetime import datetime
import telebot
import threading

config = {
    'user': '****',
    'password': '****',
    'host': '127.0.0.1',
    'database': 'server_room',
    'raise_on_warnings': True
}

mqtt_broker = "192.168.0.111"
mqtt_port = 1883
mqtt_username = "****"
mqtt_password = "****"
mqtt_topics = ["dht11/temperature", "dht11/humidity", "mq2/smoke",
"SCT/irms", "SCT/power", "fc37/flooding"]

telegram_token = '****'
bot = telebot.TeleBot(telegram_token)

temperature_alerts_sent = 0
humidity_alerts_sent = 0
smoke_alerts_sent = 0
flooding_alerts_sent = 0
irms = None
power= None
```

```
temperature = None
humidity = None
smoke = None


def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Підключено до MQTT брокера")
    else:
        print(f"Помилка підключення до MQTT, код: {rc}")


def on_message(client, userdata, message):
    global temperature_alerts_sent, humidity_alerts_sent,
smoke_alerts_sent, flooding_alerts_sent, smoke, humidity,
temperature, irms, power


    try:
        topic = message.topic
        msg_payload = str(message.payload.decode("utf-8"))


        conn = mysql.connector.connect(**config)
        cursor = conn.cursor()


        now = datetime.now().strftime('%Y-%m-%d %H:%M:%S')


        if topic == 'mq2/smoke':
            smoke = float(msg_payload)
            if smoke == 0:
                smoke_alerts_sent = 0
            elif smoke_alerts_sent < 2:
                bot.send_message(chat_id='****', text=f'Увага:
Задимлення')
                smoke_alerts_sent += 1
```

```
        if topic == 'fc37/flooding':
            flooding = int(msg_payload)
            print(flooding)
            if flooding == 0:
                flooding_alerts_sent = 0
            elif flooding_alerts_sent < 2:
                bot.send_message(chat_id='****', text=f'Увага:
Затоплення')
                flooding_alerts_sent += 1

        elif topic == 'SCT/power':
            power = float(msg_payload)
            insert_query = "INSERT INTO power (power, recorded_at)
VALUES (%s, %s)"
            cursor.execute(insert_query, (msg_payload, now))
            conn.commit()

        elif topic == 'SCT/irms':
            irms = float(msg_payload)
            insert_query = "INSERT INTO irms (irms, recorded_at)
VALUES (%s, %s)"
            cursor.execute(insert_query, (msg_payload, now))
            conn.commit()

        elif topic == 'dht11/temperature':
            temperature = float(msg_payload)
            insert_query = "INSERT INTO temperature (temperature,
recorded_at) VALUES (%s, %s)"
            cursor.execute(insert_query, (msg_payload, now))
            conn.commit()

            if 19 <= temperature <= 21:
                temperature_alerts_sent = 0
```

```
            elif temperature_alerts_sent < 2:
                bot.send_message(chat_id='****', text=f'Увага:
Температура виходить за межі! Значення: {temperature}°C')
                temperature_alerts_sent += 1

        elif topic == 'dht11/humidity':
            humidity = float(msg_payload)
            insert_query = "INSERT INTO humidity (humidity,
recorded_at) VALUES (%s, %s)"
            cursor.execute(insert_query, (msg_payload, now))
            conn.commit()

            if 40 <= humidity <= 60:
                humidity_alerts_sent = 0
            elif humidity_alerts_sent < 2:
                bot.send_message(chat_id='****', text=f'Увага:
Вологість виходить за межі! Значення: {humidity}%')
                humidity_alerts_sent += 1

        cursor.close()
        conn.close()

    except mysql.connector.Error as err:
        print(f"Database error: {err}")
    except Exception as e:
        print(f"Error: {e}")

client = mqtt.Client()
client.username_pw_set(mqtt_username, mqtt_password)

client.on_connect = on_connect
client.on_message = on_message
```

```
client.connect(mqtt_broker, mqtt_port, 60)


for topic in mqtt_topics:
    client.subscribe(topic)


@bot.message_handler(commands=['start'])
def send_welcome(message):
    bot.reply_to(message, "Привіт! Я бот для моніторингу умов у серверній. Щоб переглянути параметри в даний момент введіть /stats ")


@bot.message_handler(commands=['stats'])
def send_stats(message):
    stats_message = "Останні параметри:\n"
    if temperature is not None:
        stats_message += f"Температура: {temperature}°C\n"
    if humidity is not None:
        stats_message += f"Вологість: {humidity}%\n"
    if power is not None:
        stats_message += f"Напруга: {power:.2f} kW*h\n"
    if irms is not None:
        stats_message += f"Струм: {irms:.2f} A"
    if temperature is None and humidity is None and irms is None and power is None:
        stats_message = "Параметри зараз недоступна."
    bot.reply_to(message, stats_message)


threading.Thread(target=bot.polling).start()
try:
    client.loop_forever()
except KeyboardInterrupt:
    print("Скрипт зупинено")
```