Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління
(повна назва)

Кафедра Автоматизації проектування обчислювальної техніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)
(рівень вищої освіти)

Автоматизована система клімат-контролю на основі комірчастих самоорганізованих мереж
(тема)

Виконав:
студент _2_ курсу, групи СКСм-21-1
Малишев М. О.
(прізвище, ініціали)

Спеціальність 123 - Комп'ютерна інженерія
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Спеціалізовані комп'ютерні системи
(повна назва освітньої програми)

Керівник доцент каф. АПОТ Рахліс Д. Ю.
(посада, прізвище,ініціали)

Допускається до захисту

Зав. кафедри - Чумаченко С.В.
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління

Кафедра Автоматизації проектування обчислювальної техніки

Рівень вищої освіти другий (магістерський)

Спеціальність 123_Комп'ютерна інженерія
(шифр і назва)

Тип програми Освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Спеціалізовані комп'ютерні системи

(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
«____»_____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Малишеву Михайлу Олександровичу

(прізвище, ім'я, по батькові)

1. Тема роботи Автоматична система клімат-контролю на основі комірчастих самоорганізованих мереж.
затверджена наказом по університету від _14_ _листопада_ 2022 р. № ____1478_____
2. Термін подання студентом роботи до екзаменаційної комісії _20_ __грудня__ 2022 р.
3. Вихідні дані до роботи:
Плата WeMos D1 mini;
Середовище розробки Arduino IDE;
Датчик температури татиску BME280, датчик температури DS18B20, датчик температури та вологості DHT22, резистор 4,7 кОм, та з'єднувальні елементи;
Каомірчасті самоорганізовані мережі.
Виконувані операції:
1) Показ температури, вологості, тиску;
2) Передача даних за допомогою комірчастої самоорганізованої мережі;

4. Перелік питань, що потрібно опрацювати в роботі:
Огляд існуючих рішень;
Технологічна платформа реалізації;
Середовище розробки програмного забеспечення;
Програмування мікроконтроллеру;
Тестування розробленного приладу;
Аналіз отриманих результатів.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)_____

12 слайдів

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	Дата

7. Дата видачі завдання 14 листопада 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
	Видача теми роботи, її узгодження та затвердження.	01.09.2022– 02.09.2022	Виконав
	Аналіз проблемної області, постановка задачі, вибір засобів реалізації.	03.09.2022 – 05.09.2022	Виконав
	Придбання необхідних компонентів	10.09.2022 – 13.09.2022	Виконав
	Збірка пристрою	20.09.2022 – 3.10.2022	Виконав
	Написання коду для мікроконтролера	10.11.2022– 01.11.2022	Виконав
	Оформлення пояснювальної записки.	01.12.2022– 10.12.2022	Виконав
	Перевірка виконаного проекту, допуск до захисту.		Виконав
	Захист проекту.	20.12.2022	Виконав

Студент _____
(підпис)

Керівник роботи _____ доц. каф. АПОТ Рахліс Д.Ю.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 51 с., 21 рис., 4 розділів, 3 додатків,12 джерел.

WEMOS D1 MINI, ДАТЧИК, ТЕМПЕРАТУРА, ТИСК, МІКРОКЛІМАТ, ПРОГРАМА, ПРОШИВКА, СКЕТЧ, ВИВІД.

Метою кваліфікаційної роботи є аналіз комірчастих самоорганізованих мереж, огляд існуючих рішень на ринку, створення схематичного зображення прототипу, реалізація прототипу системи клімат-контролю на основі комірчастих самоорганізованих мереж та аналіз доцільності використання комірчастих самоорганізованих мереж для використання в системах клімат-контролю.

У ході роботи проведено аналіз існуючих рішень на ринку. Було розглянуто принцип роботи комірчастих самоорганізованих мереж, виконано підбір обладнання та середовища розробки, створено графічну схему та прототип приладу, проведено аналіз доцільності використання комірчастих самоорганізованих мереж для систем клімат-контролю.

ABSTRACT

Explanatory note of the qualification work: 51 pp., 21 figures, 4 sections, 4 appendices, 12 sources.

WEMOS D1 MINI, SENSOR, TEMPERATURE, PRESSURE, MICROCLIMATE, PROGRAM, FIRMWARE, SKETCH, OUTPUT.

The purpose of the qualification work is the analysis of cellular self-organized networks, an overview of existing solutions on the market, the creation of a schematic image of the prototype, the implementation of a climate control system prototype based on cellular self-organized networks, and the analysis of the feasibility of using cellular self-organized networks for use in climate control systems.

In the course of the work, an analysis of existing solutions on the market was carried out. The principle of operation of cellular self-organized networks was reviewed, equipment and development environment were selected, a graphic diagram and a device prototype were created, and the feasibility of using cellular self-organized networks for use in climate control systems was analyzed.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Скетч (англ. Scetch) – програма-прошивка мікроконтролера.

AVR – це назва популярного сімейства мікроконтролерів, яке випускає компанія Atmel. Крім AVR під цим брендом випускаються мікроконтролери та інших архітектур, наприклад, ARM і i8051.

CE або CS (англ. Chip Enable, Chip select) – назва лінії управління в цифровій електроніці, що використовується для вибору одного (або набору) інтегрованих мікросхем.

GND (англ. GROUND) – вузол ланцюга, потенціал якого умовно приймається за нуль, і вся напруга в системі відраховуються від потенціалу цього вузла.

IRQ (англ. Interrupt) – події, які тимчасово призупиняють роботу основної програми, передають управління зовнішнім джерелам і виконують їх завдання. Потім він передає управління головній програмі, де вона зупинилася.

MISO або MO (англ. Master In Slave Out) – вхід ведучого, вихід веденого. Служить для передачі даних від веденого пристрою до ведучого.

MOSI або MI (англ. Master Out Slave In) – вихід ведучого, вхід веденого. Служить для передачі даних від ведучого пристрою до веденого.

SCL або SCK (англ. Serial Clock) – послідовний тактовий сигнал. Служить для передачі тактового сигналу для ведених пристроїв.

SDA (англ. Serial DAta) – послідовна лінія даних.

VCC (англ. Voltage Common Collector) – вища напруга щодо заземлення (ground). VCC – це вхідна потужність пристрою.

VIN (англ. Voltage Input) – пін для входу напруги.

VOUT (англ. Voltage Output) – пін для виходу напруги.

ВСТУП

За останнє десятиліття відбулася захоплююча еволюція можливостей мікропроцесорів. Це є результатом активного використання комп’ютерів для роботи, відпочинку та розваг. Розвиток мікропроцесорів сприяв зростанню потужних персональних комп'ютерів, які використовуються у всіх сферах життя. Завдяки своїй обчислювальній потужності та швидкості мікропроцесори також знайшли свій шлях до розробки автономних продуктів, таких як електронні інструменти, які вимагають складних можливостей керування. В еволюції можливостей мікропроцесора замість зосередження на більшій ширині слова та адресному просторі. нинішній акцент робиться на надзвичайно швидкому контролі в реальному часі. Розробка мікроконтролерів була зосереджена на інтеграції засобів, необхідних для підтримки швидкого керування, в одному чіпі.

Швидкозростаючі потреби сучасного суспільства вимагають широкомасштабного, тотального використання новітніх технологій у різних відростках економіки. Розробка сучасних систем автоматизації технологічного процесу виробництва є однією з актуальних завдань розвитку економіки будь-якої держави.

Всі різноманітні засоби цифрової техніки: персональні комп'ютери, мікропроцесорні системи вимірювань і автоматизації технологічних процесів, цифровий зв'язок, телебачення, побутова техніка і т.п. побудовані на єдиній елементній базі, до складу якої входять надзвичайно різноманітні за складністю мікросхеми – від логічних елементів, що виконують найпростіші операції, до складніших програмованих кристалів, що містять мільйони логічних елементів.

З появою мікропроцесорів з програмованою структурою виникло якісна зміна підходів до методів проектування та виготовлення засобів автоматики.

Мікропроцесор здатний виконувати команди, що входять в його систему команд. Змінюючи послідовність команди (програми), можна вирішувати різні завдання на одному і тому ж мікропроцесорі. Інакше кажучи, в цьому випадку структура апаратних засобів не пов'язана з характером вирішуваної задачі. Це забезпечує мікропроцесорам масове виробництво з деякими зниженням вартості.

Дана робота присвячена аналізу комірчастих самоорганізованих мереж для використання у системах клімат-контролю для квартир, офісів та підприємств.

1 СИСТЕМА КЛІМАТ-КОНТРОЛЮ

1.1 Об’єкти аналізу в системах клімат-контролю

Клімат-контроль – це система, що складається з кондиціонера, системи опалення, системи фільтрації, спеціальних датчиків, що займаються різними місцями. Якісна система клімат-контролю враховує багато параметрів контролю різних кліматичних умов. Враховується і вулична температура та рівень сонячного випромінювання.

Система автоматичного клімат-контролю ефективно контролює параметри мікроклімату впливаючи на них за потреби. Зараз системи клімат-контролю використовуються в автомобілях, системах розумного дому, на підприємствах. Основними критеріями для аналізу в системі клімат контролю було обрано: вологість повітря, атмосферний тиск та температура повітря.

1.1.1 Вологість повітря

Вологість – це вимірювання кількості водяної пари в повітрі.

Атмосферна вологість – це міра води, яка утримується в повітрі у вигляді газу. Вода може бути твердою (лід), рідкою (вода) або газоподібною (пара). Паровий компонент становить близько 99% усієї води, що міститься в атмосфері. Повітря, яким ми дихаємо, є сумішшю газів – переважно азоту (78%) і кисню (21%) з невеликою кількістю вуглекислого газу, аргону та водяної пари, серед іншого.

Тепліше повітря може переносити більше водяної пари, ніж холодніше, якщо є велика кількість води. Це тому, що він має більше енергії для випаровування води в пару та утримання її в цьому стані. У тропіках дуже тепло і дуже вологі – повітря в тропіках містить багато водяної пари. Над дуже холодними Арктикою та Антарктикою дуже мало водяної пари. Деякі

дуже теплі регіони також є дуже сухими (наприклад, пустелі Сахара), тому що там дуже мало доступної води для випаровування в пару, а приблизно на 30 градусах на північ або південь від екватора повітря опускається зверху і вже дуже сухе.

Кількість водяної пари в повітрі можна кількісно визначити трьома способами: відносна вологість, питома вологість та точка роси.

Відносна вологість є найпоширенішим показником вологості. Він вимірює, наскільки повітря близьке до насичення, тобто скільки водяної пари міститься в повітрі порівняно з тим, скільки могло б бути за цієї температури. Тепліше повітря може утримувати більше водяної пари, оскільки є більше доступної енергії. Якщо відносна вологість повітря становить 100%, то воно повністю насичене.

У період високих температур повітря з дуже високою відносною вологістю є дуже некомфортним, оскільки насичене повітря впливає на механізм охолодження нашого тіла. Повітря не може легко містити більше води у вигляді пари і тому не може ефективно випаровувати піт із нашої шкіри.

За низьких температур повітря з дуже високою відносною вологістю може викликати у нас відчуття прохолоди. Це пов’язано з тим, що біля нашої шкіри більше водяної пари, а оскільки вода є набагато кращим провідником, ніж сухе повітря, холодне повітря передається до нашої шкіри, змушуючи нас відчувати себе холодніше.

Питома вологість і коефіцієнт змішування вимірюють кількість водяної пари в повітрі. Вони дуже схожі, але питома вологість – це маса водяної пари в масі повітря (включаючи сухе повітря та водяну пару), тоді як коефіцієнт змішування – це відношення маси водяної пари до маси сухого повітря.

Як питома вологість, так і коефіцієнт змішування є найвищими навколо екватора, близько 20 г кг-1, де повітря тепле і може містити більше водяної пари, і найнижчими (близько нуля) у холодних полярних регіонах і високо в атмосфері.

Температура точки роси або температура за вологим термометром також є показниками вологості. Це обидва показники того, наскільки близько повітря до насичення. Якщо вони дорівнюють фактичній температурі повітря, то повітря є насиченим і відносна вологість становить 100 %.

Температура за мокрим термометром є традиційним способом вимірювання вологості. Його вимірюють, дозволивши повітрю охолодити термометр, підданий впливу води шляхом випаровування. Зовнішній вигляд вологого термометра наведено на рисунку 1.1.

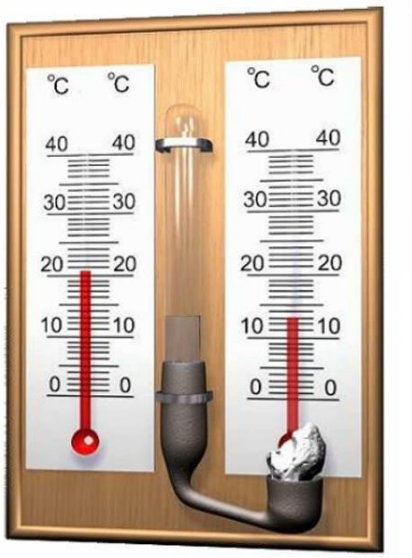

Рисунок 1.1 − Зовнішній вологого термометра

Температура точки роси вимірюється шляхом охолодження поверхні до точки, коли повітря конденсує деяку кількість водяної пари – це температура, при якій повітря стає насиченим і схоже на росу, яку можна побачити на траві рано вранці, коли температура досягає впав за ніч.

1.1.2 Температура повітря

Для перетворення води в газоподібну форму (водяну пару) потрібна енергія у вигляді тепла. Теплова енергія зазвичай вимірюється в одиницях калорій. Очевидно, що потенційна енергія води більша, ніж у льоду, а пара — у води. Ця прихована кількість тепла у водяній парі називається прихованою

теплотою. Іншими словами, кількість тепла, витраченого під час процесу випаровування, ніколи не втрачається, воно завжди пов’язане з водяною парою. Є кілька доказів, які демонструють використання теплової енергії під час випаровування.

Наприклад:

− людина відчуває охолоджуючий ефект, коли сидите перед вентилятором або в тіні на відкритому повітрі після потовиділення в літні місяці, оскільки піт випаровується;

− виникає охолоджуючий ефект, коли краплі спирту тримаються на долоні людини, оскільки спирт випаровується тощо;

− теплова енергія виділяється під час конденсації (перетворення пари в рідку або тверду форму). Ця енергія, що виділяється після конденсації, називається прихованою теплотою конденсації.

1.1.3 Атмосферний тиск

Атмосферний тиск дуже важливий для прогнозування погоди в певному місці, але спочатку нам потрібно його визначити. Атмосферний тиск означає силу, з якою частинки повітря діють на поверхню. Метеорологи зазвичай вимірюють атмосферний тиск у певній місцевості; тобто вони вимірюють тиск, який чинить вага молекул повітря в цій конкретній точці. Частинки повітря невидимі, але вони мають вагу, і зі збільшенням ваги повітря тиск також зростає, і навпаки. Атмосферний тиск зазвичай вимірюється приладом, який називається барометром, тому атмосферний тиск іноді називають барометричним. Зовнішній вигляд барометра наведено на рисунку 1.2.

Рисунок 1.2 – Зовнішній вигляд барометра

Коли повітря нагрівається, воно розширюється, а його маса зменшується, маючи на увазі, що воно стає менш щільним. Це створює систему високого тиску, яка в реальному житті виникає, коли повітря поблизу земної поверхні нагрівається нерівномірно. Розширюючись, молекули повітря прагнуть рухатися вгору, утворюючи порожній простір, який необхідно зайняти. Зазвичай це пов’язано з їхньою меншою щільністю, що прокладає шлях для більш прохолодного повітря. Прохолодніше повітря, яке має тенденцію рухатися, щоб заповнити порожнечу, нагрівається, коли воно занурюється. Це призводить до безхмарної та сухої погоди.

Навпаки, коли повітря над землею нагрівається, воно стає менш щільним, тому піднімається. Коли він піднімається, він прагне охолонути. Результатом є конденсація повітря, утворення хмар і, зрештою, опади. Це те, що характеризує систему низького тиску. Коли синоптики прогнозують, що система низького тиску прямує до вас, це означає, що буде хмарно з високою ймовірністю дощу.

Підсумовуючи, атмосферний тиск є найкращим індикатором поточних і мінливих погодних умов, якщо не найважливішим. Синоптикам достатньо знати атмосферний тиск у певному місці, і погоду можна передбачити. У більшості випадків прогноз погоди характеризується передбаченням того, яка система тиску буде відчуватися в певній місцевості.

1.2 Аналіз предметної області

Мікроконтролер (MCU) – невеликий комп’ютер на мікросхемі з інтегрованою схемою (IC). Один або кілька центральних процесорів (процесорних ядер), пам’ять і програмовані периферійні пристрої вводу/виводу є компонентами мікроконтролера. Пам'ять програм у вигляді сегнетоелектричної оперативної пам'яті, NOR-флеш-пам’яті або OTP-ПЗУ також часто включається в мікросхему, а також невелика кількість оперативної пам'яті. Мікроконтролери призначені для вбудованих програм, на відміну від мікропроцесорів, що використовуються в персональних комп'ютерах або інших додатках загального призначення, що складаються з різних дискретних мікросхем[10]. Зовнішній вигляд мікроконтролерів наведено на рисунку 1.3.

Рисунок 1.3 − Зовнішній вигляд мікроконтролерів

Сучасною мовою мікроконтролер — це система на кристалі (SoC), але вона менш складна. Мікроконтролер може бути одним із компонентів SoC, але зазвичай він поєднується з більш складними периферійними пристроями, такими як графічний процесор (GPU), модуль Wi-Fi або один чи більше співпроцесорів. Системи керування автомобільними двигунами, імплантовані

медичні пристрої, пульти дистанційного керування, офісне обладнання, побутова техніка, електроінструменти, іграшки та інші вбудовані системи – це лише деякі приклади автоматично керованих товарів і гаджетів, які використовують мікроконтролери.

Мікроконтролери роблять доступним цифрове керування ще більшою кількістю пристроїв і процесів, оскільки вони менші та дешевші, ніж конструкції, які вимагають окремих мікропроцесорів, пам'яті та пристроїв введення/виведення. Зазвичай можна знайти мікроконтролери зі змішаними сигналами, які об'єднують аналогові частини, необхідні для керування нецифровими електричними системами.

За розрядністю також поділяють мікроконтролери. Вартість, продуктивність ядра процесора, функціональність і енергоспоживання в статичному режимі є ключовими відмінностями між 8-бітними та 32-бітними мікроконтролерами. Перед початком нового проекту розробники повинні ретельно оцінити потреби в мікроконтролері з точки зору потужності ядра процесора, доступності інтерфейсів і енергоспоживання. Хоча 32-розрядні мікроконтролери працюють краще, ніж 8-розрядні, інженерам все одно доводиться вирішувати між тим, що доступно, і тим, що насправді потрібно для їхніх проектів. Важливо вибрати інструмент, який найбільше підходить для проекту, а не хвилюватися про те, яке ядро процесора 8-розрядне чи 32-розрядне краще.

Звичайно, таке рішення суттєво позначиться на кінцевій ціні продукту.

1.2.1 Платформа розробки та середовище програмування

Апаратне та програмне забезпечення з відкритим кодом є основою електронної платформи, відомої як (Ардуіно)Arduino. Плати Arduino можуть зчитувати вхідні дані, такі як світло датчика, палець користувача на кнопці або повідомлення в соціальній мережі Twitter, і перетворювати їх у вихідні дані, такі як увімкнення двигуна чи світлодіода, або надсилання повідомлення в мережу. Надаючи мікроконтролеру плати набір інструкцій,

ви можете керувати діями плати. Ви можете досягти цього за допомогою програмного забезпечення Arduino на основі обробки та мови програмування Arduino (IDE) на основі підключення[1].

Протягом багатьох років незліченна кількість проектів, починаючи від простих предметів домашнього вжитку і закінчуючи складними науковими приладами, реалізувалася на базі Arduino. Студенти, любителі, художники, програмісти та професіонали з усього світу зібралися навколо цієї платформи з відкритим кодом, вносячи внесок у величезну кількість знань, які зараз доступні та можуть бути корисними як для новачків, так і для спеціалістів.

В Інституті дизайну взаємодії Ivrea, Arduino було створено як простий інструмент для швидкого створення прототипів, орієнтований на студентів, які не мають попередніх знань з електроніки чи програмування. Платформа Arduino почала розвиватися в міру того, як її сприйняла більша аудиторія, вирізняючи пропозицію продуктів від звичайних 8-розрядних плат до елементів для додатків Інтернету речей (IoT), переносних пристроїв, 3D-друку та вбудованих налаштувань. Усі плати Arduino повністю відкриті, що дозволяє користувачам створювати їх самостійно та згодом налаштовувати відповідно до власних потреб. Окрім відкритого вихідного коду, програма розширюється завдяки внескам користувачів з усього світу.

Текстовий редактор для написання коду, область повідомлень, текстова консоль, панель інструментів із кнопками для звичайних операцій і низка меню — усе це включено до Arduino IDE, часто відомого як Arduino Software (IDE). Щоб завантажувати програми та спілкуватися з ними, він підключається до апаратного забезпечення Arduino та Genuino.

Скетчі — це комп'ютерні програми, створені за допомогою Arduino IDE. Ці програми створено за допомогою текстового редактора та збережено у вигляді файлів із закінченням .ino. Редактор пропонує інструменти для заміни тексту та пошуку тексту. Під час збереження та експорту розділ повідомлень відтворює звуковий сигнал та відображає помилки. На консолі відображається текст, згенерований програмним забезпеченням Arduino

(IDE), а також повні повідомлення про помилки та інші деталі. Послідовний порт і налаштована плата видно в нижньому правому куті вікна. Ви можете переглядати та завантажувати програми, створювати, відкривати та зберігати ескізи, а також запускати серійний монітор за допомогою кнопок на панелі інструментів. Наступні п’ять меню містять додаткові команди: Файл, Правка, Ескіз, Інструменти та Довідка. Меню є контекстно-залежним, тому представлені лише ті варіанти, які стосуються поточної частини роботи.

Крім того, цей інструмент дозволяє керувати скетчами з численними файлами (кожен відображається на окремій вкладці). Це можуть бути файли простого коду Arduino, файли C із розширенням .c, файли C++ із розширенням .cpp або файли заголовків (.h).

Зовнішній вигляд середовища програмування Arduino IDE наведено на рисунку 1.4.

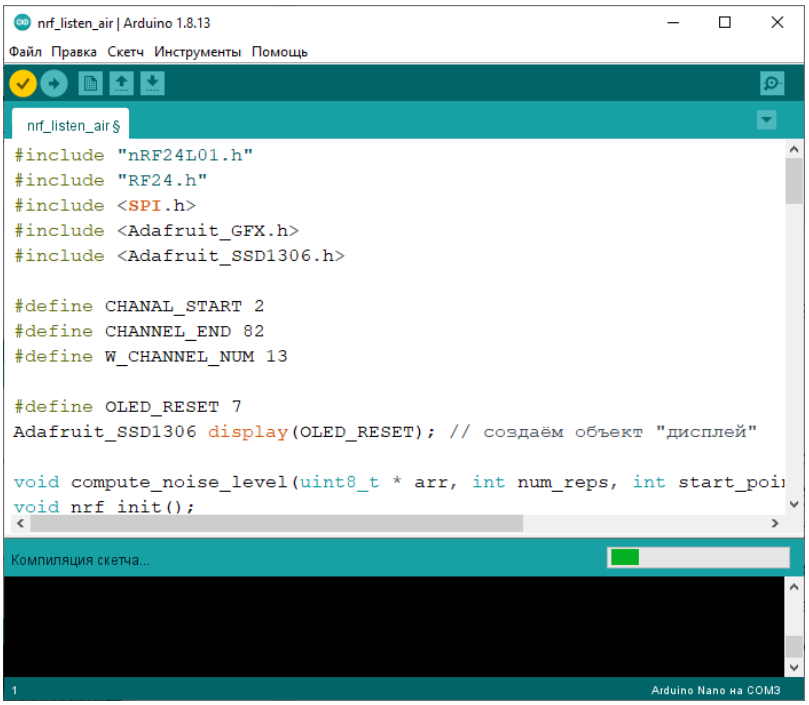

Рисунок 1.4 – Вікно Arduino IDE

1.3.3 Опис комірчастих самоорганізованих мереж

Комірчасті самоорганізовані мережі також відомі як самоорганізовані Mesh мережі.

За останні десять років інформаційні мережі з топологією Mesh набувають все більшої популярності. Кількість користувачів і точок доступу на проект зросла до десятків тисяч.

Mesh-мережі – це найбільш інтригуюча інтеграція мережевих і радіотехнологій, яка повністю задовольняє зростаючі очікування абонентів (мобільність, QoS, безпека).

Більшість Mesh-мереж, які зараз використовуються, побудовані з використанням бездротового протоколу Wi-Fi. Переваги такого рішення очевидні: комерційна рентабельність проектів визначається широким асортиментом доступних стандартних абонентських пристроїв.

Mesh — це топологія мережі, у якій бездротові пристрої з’єднані великою кількістю з’єднань, які зазвичай є надлишковими та додаються для стратегічних цілей. Це визначення добре описує, як працюють ці розгорнуті мережі.

У Mesh-мережах можлива як повністю зв’язана топологія, в якій кожен вузол з’єднаний з кожним іншим вузлом мережі, так і частково зв’язана топологія, в якій кожен вузол з’єднаний лише з деякими вузлами мережі.

Mesh-мережі мають потенціал до самоорганізації та модифікації. Mesh-мережі, які самоорганізуються, зараз найбільш поширені.

Топологія комірчастої самоорганізованої мережі приведена на рисунку 1.5.

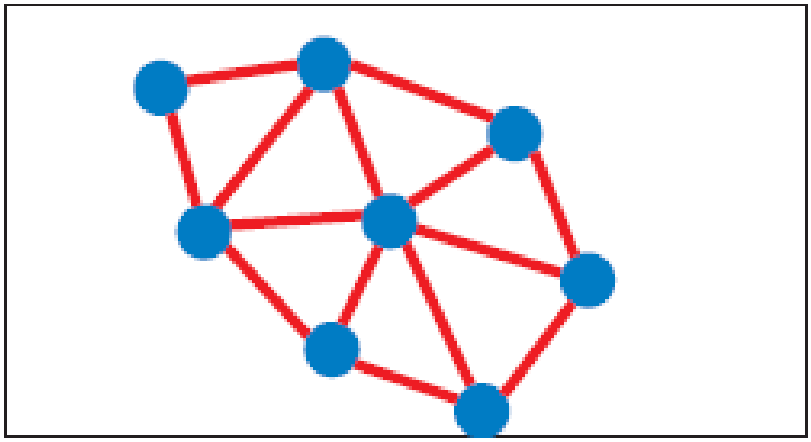

Рисунок 1.5 − Топологія комірчастої самоорганізованої мережі

Структура Mesh-мереж заснована на децентралізованому методі управління зв'язком між активними вузлами мережі. У сітчастих мережах вузли доступу служать маршрутизаторами (ретрансляторами) для інших вузлів у тій самій мережі для надання послуг доступу. Це робить можливим побудову розширених мереж, які можна масштабувати, мають взаємозамінні активні вузли та дуже стійкі до збоїв мережі (у цьому випадку нові вузли додаються до мережі автоматично)[12].

Переваги Mesh-мереж наведено нижче.

Швидке автоматичне розгортання та нарощування мережі. У сучасних Mesh-мережах вузли можуть автоматично приєднуватися та залишати мережу у будь-який час. Подібні мережі можуть бути розгорнуті будь-де навіть за відсутності інфраструктури фіксованої мережі.

Самоорганізація, самовідновлення та саме балансування (навантаження) мережі. Mesh-мережі мають значно більшу стійкість ніж традиційні бездротові мережі. Автоматична конфігурація та маршрутизація дозволяє мережі бути самоорганізованою та самовідновлюваною. Мережа продовжує функціонувати навіть якщо один або кілька вузлів вийдуть із ладу.

Збільшена зона покриття. Високі швидкості передачі даних вимагають великих співвідношень сигнал/шум. Однак, оскільки рівень сигналу зменшується експоненційно зі збільшенням відстані, то традиційних мережах

не завжди вдається забезпечити високі швидкості передачі даних для віддалених вузлів. При цьому в Mesh-мережах рівень сигналу відновлюється з кожним стрибком, тому зона покриття такої мережі теоретично може збільшуватися необмежено при збереженні показників якості функціонування для всіх вузлів.

Визначення розташування вузлів без використання технології GPS.

Низькі інфраструктурні та операційні витрати. У більшості випадків Mesh-мережі вимагають для свого функціонування значного числа транспортних (частотних) каналів, ніж традиційні бездротові мережі, що значно зменшує витрати на розгортання мережі. Оскільки Mesh-мережі є самоорганізуючими та самовідновлюваними, це означає зменшення витрат на адміністрування, обслуговування та технічну підтримку подібних мереж – значно менших, ніж у стільникових чи інших мережах з централізованою архітектурою.

Стійке функціонування в умовах багатопроменевого поширення радіохвиль, радіоперешкод і відсутності прямої видимості. За рахунок технології безлічі стрибків Mesh-мережі дозволяють забезпечити зв'язок за перешкодами (немає прямої видимості) і стійкі до радіоперешкод та багатопроменевого поширення радіохвиль.

Недолік Mesh-мереж полягає в тому, що вони використовують проміжні пункти передачі даних. Це може спричинити затримку під час пересилання інформації та як наслідок знизити якість трафіку реального часу (наприклад, мови або відео) [11]. Внаслідок цього часто вводяться обмеження на кількість точок доступу в одному кластері.

1.4 Дослідження існуючих рішень

Підчас дослідження ринку було знайдено лише одну компанію(литовська компанія “Mesh”) яка займається системами клімат-контролю на онові комірчастих самоорганізованих мереж. Компанія продає

системи комерційного призначення, а оплатою є відсоток збережених фінансових ресурсів від опалення, що унеможливлює аналіз цін. Тому можна сказати що цей ринок досить новий.

1.5 Постановка мети та задач роботи

Аналіз предметної області та існуючих комерційних рішень в галузі систем клімат-контролю показав актуальність обраної теми. Тому метою кваліфікаційної роботи є створення моделі системи клімат-контролю на базі комірчастих самоорганізованих мереж. Практичним результатом має стати прототип приладу, який може отримувати значення температури, вологості та атмосферного тиску. Для досягнення поставленої мети необхідно вирішити наступні задачі:

- аналіз предметної області
- дослідження можливості використання комірчастих самоорганізованих мереж в системах клімат-контролю;
- створення моделі системи клімат-контролю;
- вибір компонентів;
- апаратна реалізація моделі;
- створення скетчу для мікроконтролерів;
- тестування приладу;
- аналіз отриманих результатів.

Об’єктом кваліфікаційної роботи є система клімат-контролю для моніторингу показників клімату.

Предметом дослідження є комірчаста самоорганізована мережа.

2 МОДЕЛЬ СИСТЕМИ КЛІМАТ-КОНТРОЛЮ

В даному розділі проводиться аналіз, створення моделі прототипу пристрою та підбір комплектуючих для створення прототипу пристрою.

2.1 Створення моделі системи клімат-контролю

Було створено модель, тобто структурну схему по якій обирались компоненти системи. Структурна схема наведена на рисунку 2.1.

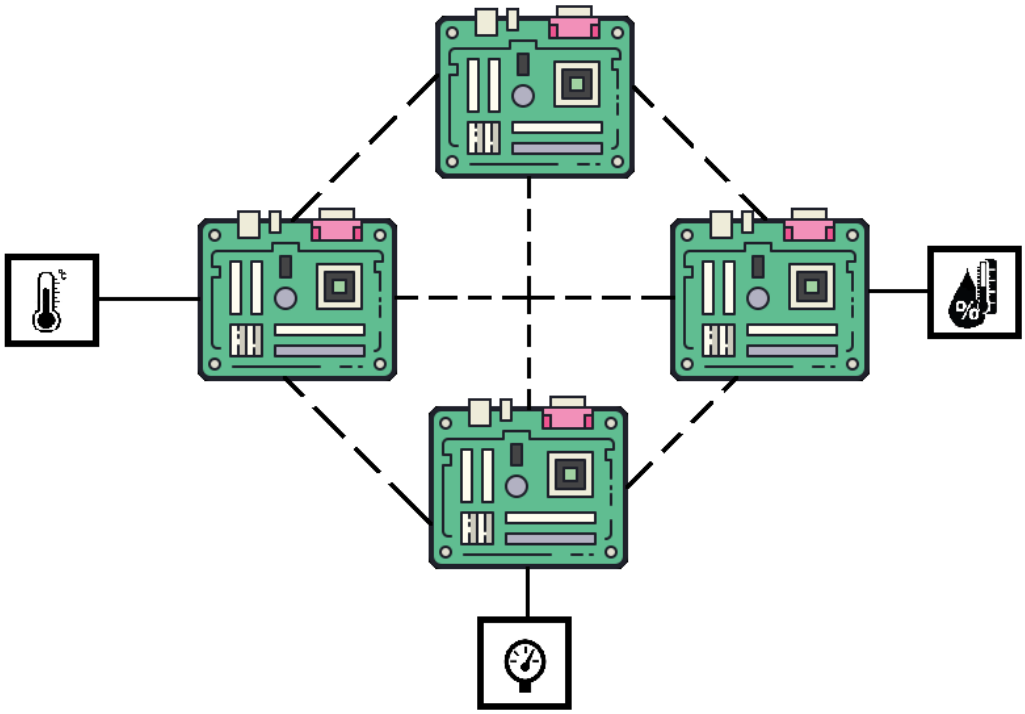

Рисунок 2.1 – Модель запропонованої системи

З рисунку 2.1 запропоновано взяти чотири мікроконтролери, які будуть з’єднані комірчастою самоорганізованою мережею, що позначено на моделі штрихованою лінією; датчик температури; датчик атмосферного тиску; датчик вологості повітря.

2.2 Вибір мікроконтролера для проекту

Для створення проекту було відібрано чотири мікроконтролери ESP8266, ESP32, STM32 та Arduino MKR1000. Для проекту було обрано мікроконтролер ESP8266 (WeMos D1 mini) на основі мікрокопроцесора Tensilica 32-bit RISC CPU Xtensa LX106, основними факторами були та функціональні можливості, ціна та розміри.

WeMos D1 mini – середовище для розробки програмного та апаратного забезпечення з відкритим кодом. ESP8266, розроблений та виготовлений компанією WeMos, містить найважливіші елементи комп'ютера: процесор, оперативну пам'ять, мережу (WiFi) і сучасну операційну систему та SDK, що може бути використано для для проектів Інтернету речей (IoT)[2].

WeMos – це китайська компанія, яка виробляє Arduino-сумісні мікроконтролери зручного розміру з великими можливостями. Власник бренду WeMos, виробник однойменних мікросхем з Китаю. Фактично за своїми функціональними можливостями мікроконтролер є міні-комп'ютером, що виконує керування та контроль над роботою електронних модулів, інтегрованих між собою для виконання різних завдань.До найважливіших технічних характеристик їх пристроїв відносять: тип інтерфейсу, наявність у схемі плати модуля Wi-Fi та портів microUSB, тип та обсяг флеш-пам'яті.

WeMos D1 mini – сумісна з Arduino плата з вбудованим модулем Wi-Fi ESP8266. Плата розробки, сумісна з Arduino, WeMos D1 mini – це найдешевша плата з підтримкою Wi-Fi на сьогодні. Система на чіпі (SoC) ESP8266 дає можливість бездротового з’єднання з іншими пристроями за допомогою протоколу передачі даних Wi-Fi. Ця плата сумісна з усіма існуючими щитами для Arduino. На платах замість чіпа ATMEGA, який використовують стандартні плати Arduino, використовується система на чіпі (SoC) під назвою ESP8266.

2.3 Опис апаратного забезпечення

2.3.1 WeMos D1 mini

WeMos D1 mini побудована на базі 32-розрядного мікроконтролера ESP8266 (мікроконтролер входить у збірку ESP12-E встановлену на платі) з інтегрованим WiFi модулем (802.11 b/g/n 2.4 ГГц). Також на платі присутні стабілізатор напруги на 3,3 В, роз'єм USB типу Micro-B і USB-UART перетворювач на базі чіпа CH340G. Мікроконтролер ESP8266 працює на тактовій частоті 80 МГц і має оперативну пам'ять RAM даних на 80 КБ (для зберігання значень змінних), і пам'ять RAM інструкцій на 32 КБ[2].

Технічні характеристики WeMos D1 mini:

- мікроконтролер: ESP8266;
- розрядність: 32 біти;
- напруга живлення плати: 3,3/5,0 В;
- бездротовий інтерфейс: Wi-Fi 802.11 b/g/n 2,4 ГГц (STA/AP/STA+AP, WEP/TKIP/AES, WPA/WPA2);
- шини, що підтримуються: SPI, I2C, I2S, 1-wire, UART, UART1, IR Remote Control;
- цифрові виводи I/O: 11 (RX, TX, D0...D8) всі виводи крім D0 підтримують INT (зовнішнє переривання), ШІМ, I2C, 1-wire;
- аналогові входи: 1 (A0) 10-біт АЦП;
- логічні рівні висновків I/O: 3,3 В;
- максимальний струм на виведенні I/O: 12 мА (кожний висновок);
- максимальна напруга на вході A0: 3,2 В (між виводом A0 та GND);
- Flash-пам'ять: 4 МБ.RAM-пам'ять даних: 80 КБ.RAM-пам'ять інструкцій: 32 КБ;
- тактова частота мікроконтролера: 80 МГц;
- чіп USB-UART перетворювача: CH340G;
- робоча температура: -40...+85°C;
- габарити: 34,2 х25, 6 мм;

– вага: 10 г.

Програми зберігаються у flash пам'яті об'ємом 4 МБ. Усі цифрові виводи крім D0 можна використовувати для роботи із зовнішніми перериваннями, ШІМ, шиною I2C або 1-wire (виводи шини I2C за замовчуванням D1 та D2, але їх можна перепризначити). Логічні рівні всіх цифрових виводів 3,3 В. На аналоговий вхід A0 можна подавати напругу до 3,2 В. Висновки D3, D4 та D8 підтягнуті до 3V3 через резистори 10 кОм (це пов'язано з особливістю завантаження скетчів у плату).

Плата розроблена для створення проектів Інтернет речей, мікроконтролер здатний зберігати з'єднання з точкою доступу WiFi при зниженому енергоспоживання, всього 1 мА. Це дозволяє створювати пристрої, що працюють від акумуляторів або батарейок.

Зовнішній вигляд та опис пінів WeMos D1 mini приведено на рисунку 2.2.

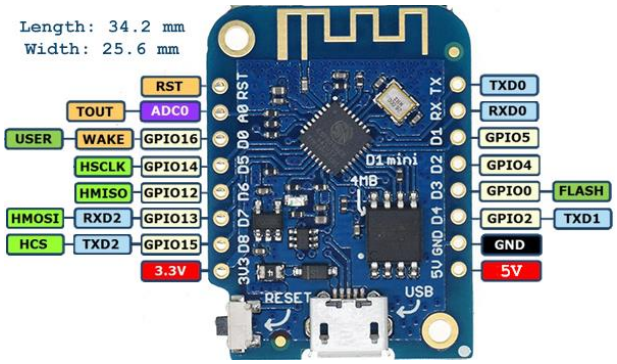

Рисунок 2.2 – Зовнішній вигляд та опис пінів WeMos D1 mini

Мікроконтролер споживає до:

– 200 мА в режимі передачі даних WiFi;

– 60 мА в режимі прийому даних WiFi;

– 40 мА у режимі очікування;

– 1 мА в режимі зниженого енергоспоживання зі збереженням з'єднання WiFi;

– 20 мкА у режимі глибокого сну.

2.3.2 ESP8266

Модуль ESP8266 – це модуль Wi-Fi, який належить до сімейства ESP, його можна використовувати для управління проектами в галузі електроніки в будь-якій точці світу. Він має вбудований мікроконтролер та Flash пам'ять розміром 1 Мб, що дозволяє йому підключатися до Wi-Fi. Стек протоколів TCP / IP дозволяє модулю взаємодіяти з сигналами Wi-Fi. Максимальна робоча напруга модуля становить 3.3 В[3].

Технічні характеристики мікроконтролеру ESP8266 (ESP-12E) наведено на рисунку 2.3.

Categories	Items	Values
WiFi Paramters	WiFi Protocles	802.11 b/g/n
	Frequency Range	2.4GHz-2.5GHz (2400M-2483.5M)
Hardware Paramaters	Peripheral Bus	UART/HSPI/I2C/I2S/Ir Remote Contorl
		GPIO/PWM
	Operating Voltage	3.0~3.6V
	Operating Current	Average value: 80mA
	Operating Temperature Range	-40°~125°
	Ambient Temperature Range	Normal temperature
	Package Size	16mm*24mm*3mm
	External Interface	N/A
Software Parameters	Wi-Fi mode	station/softAP/SoftAP+station
	Security	WPA/WPA2
	Encryption	WEP/TKIP/AES
	Firmware Upgrade	UART Download / OTA (via network) / download and write firmware via host
	Ssoftware Development	Supports Cloud Server Development / SDK for custom firmware development
	Network Protocols	IPv4, TCP/UDP/HTTP/FTP
	User Configuration	AT Instruction Set, Cloud Server, Android/iOS App

Рисунок 2.3 – Технічні характеристики мікроконтролеру ESP-12E

Зовнішній вигляд та опис пінів ESP8266 (ESP-12E) приведено на рисунку 2.4.

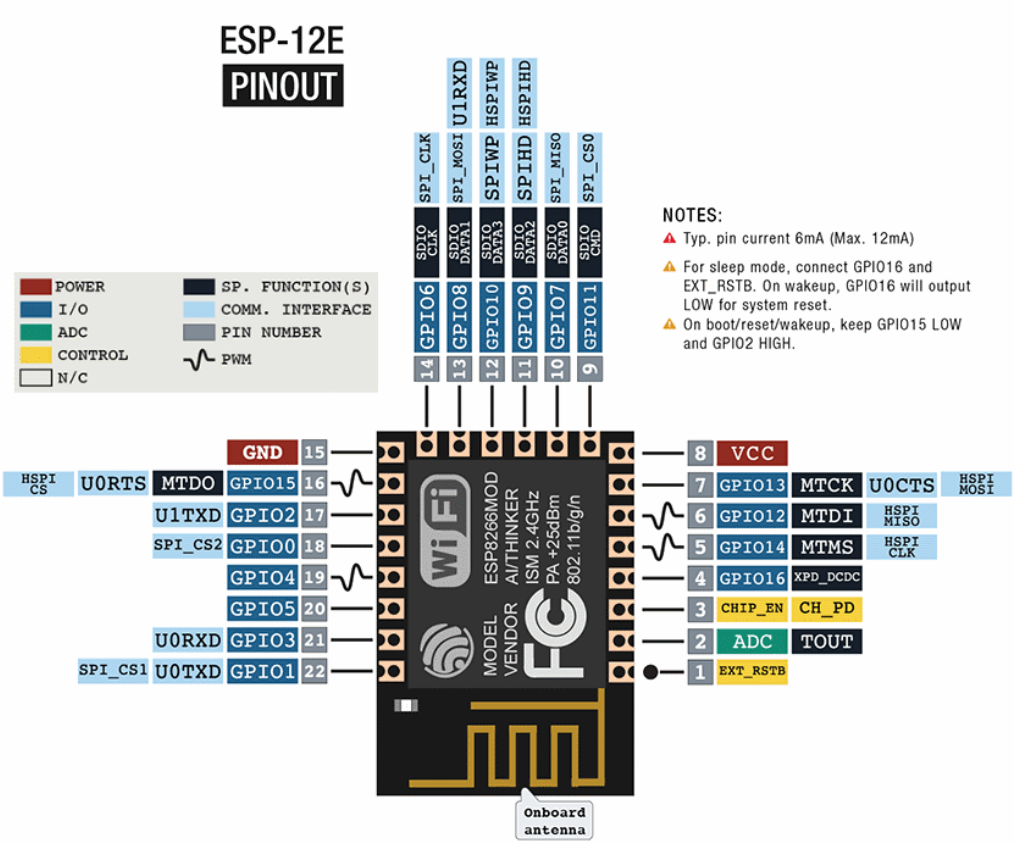

Рисунок 2.4 – Зовнішній вигляд та опис пінів ESP8266 (ESP-12E)

2.2.3 Модуль KY-015 датчиком температури та вологості на DHT22

DHT22 – це цифровий датчик вологості та температури, що складається з термістора та ємнісного датчика вологості. Також датчик містить у собі АЦП для перетворення аналогових значень вологості та температури. Датчик DHT22 не має високу швидкодію і точність, зате простий, недорогий і відмінно підходять для навчання і контролю вологості в приміщенні[6]. Зовнішній вигляд та опис виводів модулю KY-015 наведено на рисунку 2.5.

Рисунок 2.5 – Зовнішній вигляд та опис виводів модулю KY-015

2.2.4 Модуль BME280

Датчик атмосферного тиску вологості та температури повітря. BME280 може вимірювати температуру в діапазоні від -40°C до 85°C У діапазоні температур від 0 до 65°C точність становить ±1,0°C; за межами цього діапазону точність падає до ±1,5°C. Модуль також може вимірювати відносну вологість у діапазоні від 0 до 100% з точністю ±3%, вимірювати тиск від 300 Па до 1100 гПа з абсолютною точністю ±1 гПа. При вимірюванні атмосферного тиску в діапазоні температур від 0 до 65°C досягається повна точність. За межами цього діапазону точність падає до 1,7 гПа[7].

Зовнішній вигляд та опис виводів зображено на рисунку 2.6.

Рисунок 2.6 – Зовнішній вигляд та опис виводів модулю BME280

2.2.5 Датчик DS18B20

DS18B20 це датчик температури виробництва “Dallas Semiconductor”. Використовується лише один цифровий контакт для зв’язку з мікроконтролером. Датчик температури DS18B20 досить точний і не потребує зовнішніх компонентів для роботи. Він має температурний діапазон

від -55°C до +125°C і точність ±0,5°C.Роздільна здатність датчика температури може бути встановлена на 9, 10, 11 або 12 біт. Однак роздільна здатність за замовчуванням під час увімкнення живлення становить 12 біт (тобто з точністю 0,0625°C).Датчик працює від джерела живлення від 3 В до 5,5 В і споживає лише 1 мА під час активного перетворення температури[5]. Зовнішній вигляд та опис виводів зображено на рисунку 2.7.

Рисунок 2.7 – Зовнішній вигляд та опис виводів датчику DS18B20

2.2.6 Резистор

Резистор з вуглецевим провідним шаром, призначений для роботи в ланцюгах постійного, змінного та імпульсного струму. Опір – 4.7 кОм. Можлива розбіжність опору – 5%. Діапазон робочих температур від -55 до +125 ° С.

Зовнішній вигляд зображено на рисунку 2.8.

Рисунок 2.8 – Зовнішній вигляд резистору

2.2.7 Набір дротів

Провід сполучний для збірки схем і елементів з роз'ємом - штекер (Female to Male), який використовується для з'єднання штирьового роз'єму будь-якої плати розробки (наприклад, Arduino, макетної платою) з іншого платою розробки, датчикам або сенсором. Також ви можете комбінувати її з перемичкою типу роз'єм «Male» для створення перемички «Female-Male». Провід поставляються у вигляді 40-контактного стрічкового шлейфу, який можна або об'єднати, щоб акуратно організувати, або від'єднати дроти для створення окремих перемичок.

Зовнішній вигляд дротів зображено на рисунку 2.9.

Технічні характеристики:

– тип проводу для Arduino роз'єм - штекер (Male - Female);
– набір різнокольорових дротів 10 кольорів;
– загальна кількість проводів 40 шт;
– довжина кожного з'єднувального проводу 20 см.

Рисунок 2.9 – Зовнішній вигляд дротів

3 АПАРАТНА РЕАЛІЗАЦІЯ МОДЕЛІ

В даному розділі розглянуто розроблений прототип автоматичної системи клімат-контролю на основі комірчастих самоорганізованих мереж, яка містить обрані компоненти які описані у пункті 2.2. Приведено структурну та функціональну схеми взаємодії компонентів пристрою та результати його тестування.

3.1 Функціональна схема з’єднання

Під час розробки даного проекту було створено функціональну схему, які зображено на рисунку 3.1 відповідно.

Функціональна схема описує фізичне з’єднання. Для з’єднання мікроконтролерів та датчиків використовуються такі середовища передачі даних як радіохвилі та дроти.

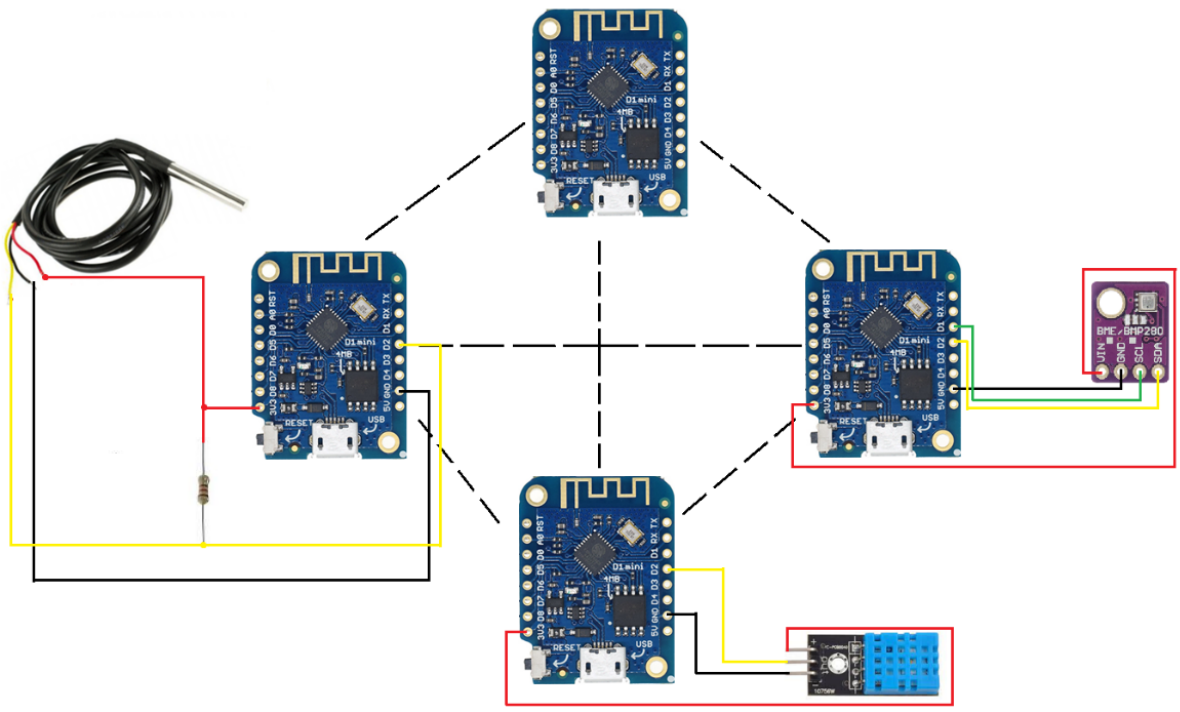

Рисунок 3.1 – Функціональна схема проекту

Як зображено на рисунку 3.1, основою розробленої схеми є чотири плати WeMos D1 mini із під'єднаними до них периферійними модулями. Живлення схеми відбувається через mini-USB порт або ж через блок живлення шляхом перепайки самого mini-USB.

На схемі розташовані:

– чотири плати WeMos D1 mini на основі мікроконтролеру ESP8266;

– датчик BME280 3.3B I2C;

– датчик DS18B20;

– плата KY-015 на основі датчику DHT22;

– резистор з опором 4.7кОм;

– з'єднувальні дроти.

Розглянемо з`єднання компонентів системи детальніше. На функціональній схемі, червоним кольором позначено VCC (напругу 3,3В), чорним кольором GND (заземлення), жовтим кольором SDA (послідовну лінію даних), зеленим кольором позначено синхросигнал (Clock).

Поточна схема зв'язку між мікроконтролерами та периферійними пристроями представлена функціональною схемою на рисунку 3.2. Для з'єднання компонентів один з одним використовуються дроти (Female to Male) і електромагнітні хвилі. Хоча для вдосконалення пристрою бажано створити друковану плату та з'єднати всі компоненти за допомогою пайки, але для демонстраційних цілей було обрано поточний варіант.

Розглянемо під'єднання додаткових компонентів до основних мікроконтролерів.

3.1.1 Підключення модуля BME280 до WeMos D1 mini

Для підключення було взято 4 дроти, пункт 2.2.7. Також для підключення використовується програмна шина I2C, через це модуль використовує лише один вивід для передачі даних. Для коректної роботи

модуля необхідно під’єднати чотири контакти GND, VCC, SCL, SDA (рис. 3.2):

- GND – до виводу GND на WeMos D1 mini, GND-GND;
- VIN(3,3V) − до виводу WeMos D1 mini, 3V3(3.3V) – VIN(3,3V);
- SCL – до виводу D1 на WeMos D1 mini, SCK – D1;
- SDA − до виводу D2 на WeMos D1 mini, SDA – D2.

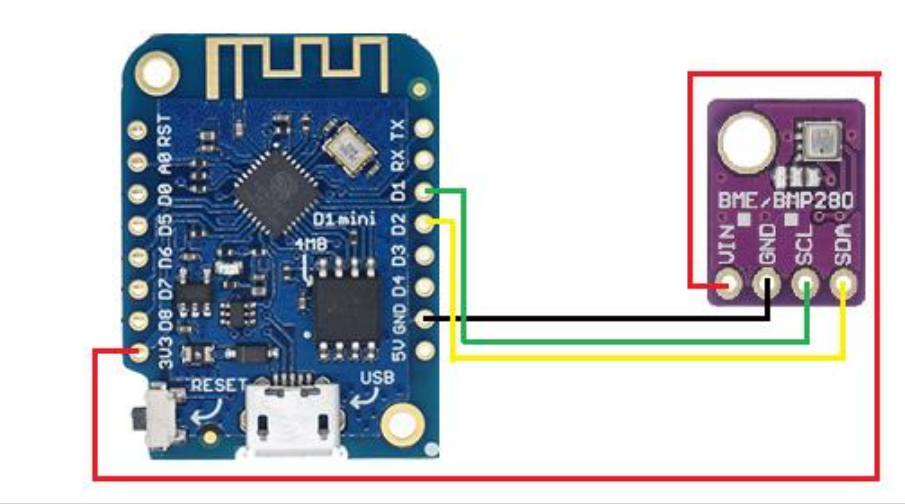

Рисунок 3.2 − Підключення модуля BME680 до WeMos D1 mini

3.1.2 Підключення модуля KY-015 до WeMos D1 mini

Підключення модуля KY-015 на основі DHT22 до плати WeMos D1 mini трьома контактами GND, VCC, SDA (рис. 3.3):

- GND – до виводу GND на WeMos D1 mini, GND-GND;
- VIN(3,3V) − до виводу WeMos D1 mini, 3V3(3.3V) – VIN(3,3V);
- SDA − до виводу D2 на WeMos D1 mini, SDA – D2.

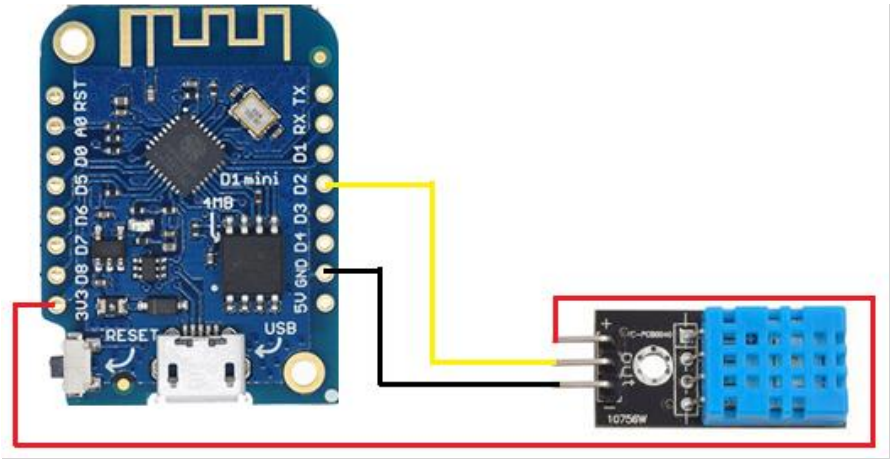

Рисунок 3.3 – Підключення модуля KY-015 до WeMos D1 mini

3.1.3 Підключення датчику DS12B20 до WeMos D1 mini

Підключення датчику DS12B20 до плати WeMos D1 mini, для цього було використано підтягуючий резистор номіналом 4.7кОм.

Червоний дрід датчику одночасно з'єднано з виводом 3V3 плати WeMos D1 mini та одним з контактів резистора. Інший контакт резистора одночасно під'єднано до виводу D2 плати WeMos D1 mini та жовтого дроту датчику. Чорний дрід датчику DS12B20 під'єднано до виводу GND плати WeMos D1 mini. Приклад підключення наведено на рисунку 3.4.

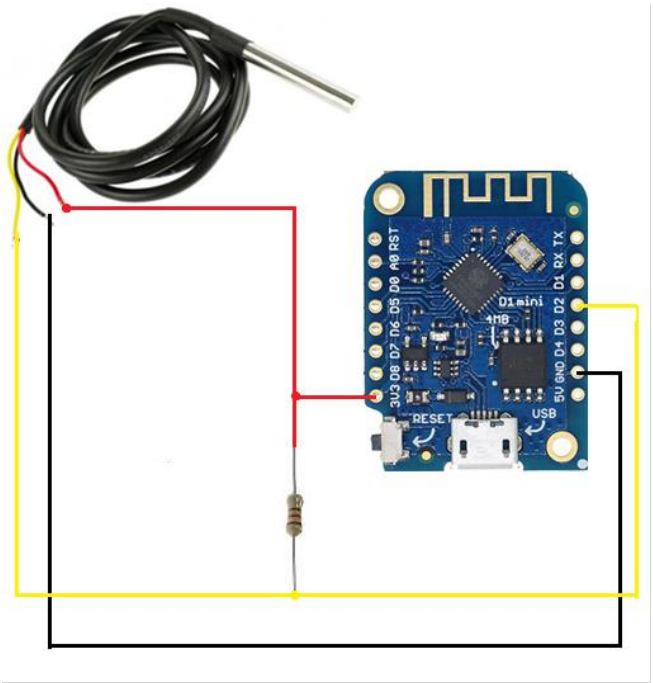

Рисунок 3.4 – Підключення модуля модуля DS12B20 до WeMos D1 mini

3.2 Тестування схеми розробленого прототипу

Прототип було зібрано основуючись на функціональній схемі з пункту 3.1. Також було перевірена справність усіх компонентів прикладами з бібліотек, які використовуються в даному проекті. Збірка системи зображена на рисунку 3.5.

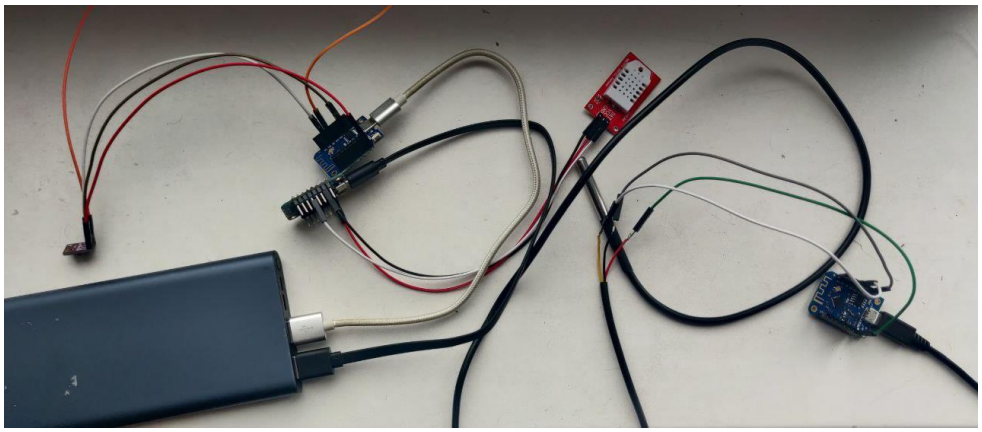

Рисунок 3.5 – Збірка системи

Тестування системи проводилось в декілька кроків. Тестування з'єднання за допомогою комірчастої самоорганізованої мережі проводилося додаванням нових комірців мережі та від'єднання їх. Тестування всього прототипу пристрою відбувалось нагріванням та охолодженням різних датчиків та перевіркою повідомлень.

Для тестування роботи прототипу приладу було подано живлення на всі плати WeMos D1 mini кожен з яких було послідовно від'єднано від живлення крім того до якого не підключені датчики, бо він відображає результати. Результати роботи проекту ви зможете побачити у вікні послідовного монітора. На представленому рисунку 3.6 видно, що ми отримуємо дані від трьох датчиків, що свідчить про коректне перепідключення та правильне функціонування мережі. На рисунку також можна побачити унікальний номер комірки в полі "Received from", ім'я комірки – "Node Name", ім'я датчику "Sensor Type" та показники.

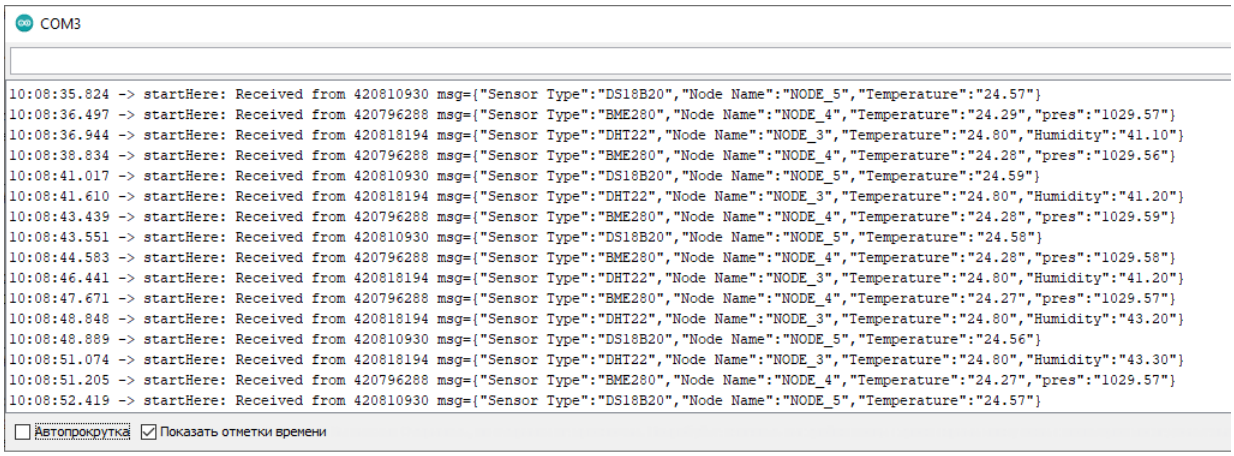

Рисунок 3.6 – Тестування системи

Результати тестування підтверджують працездатність прототипу пристрою.

3.3 Технічні умови та інструкція користувача

Користування прототипом приладу не передбачено в місцях де можливе потрапляння води на нього, в приміщеннях з високим забрудненням повітря.

Для того, щоб використовувати даний прилад, потрібно:

− під`єднати кожну плату WeMos D1 mini до джерела живлення за допомогою Mini USB (під`єднати можна до ноутбука, комп`ютера чи павербанку, де є вхід USB);

− підключити WeMos D1 mini який не підєднано до жодного датчику до комп’ютера за допомогою USB шини та відкрити монітор послідовного порта до якого підключений WeMos D1 mini.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОЕКТУ

4.1 Налаштування скетчу

Для створення програми/скетчу в середовищі програмування Arduino IDE була використана мова програмування C++ та операційна система Windows 10.

Перед початком роботи необхідно встановити драйвери плати Arduino. Ви повинні підключити плату Arduino до локального комп'ютера за допомогою USB і запустити «Диспетчер пристроїв» на локальному комп'ютері, щоб виконати це. Якщо в назві пристрою є «CH340», драйвер уже встановлено; однак, якщо пристрій відображається як «Невідомий пристрій», ви повинні інсталювати драйвер, який можна знайти в папці файлів Arduino IDE у «\drivers\dpinst-x86.exe».

Ви повинні запустити середовище розробки, перейти до «Файл / Параметри», ввести URL-адресу (http://arduino.esp8266.com/stable/package esp8266com_index.json) у область «Додаткові посилання для Менеджера плат:», а потім клацнути "ok" для роботи з мікроконтролером ESP8266 в Arduino IDE.

Далі виберіть найновішу версію та клацніть «Встановити» в меню «Інструменти / Плата / Менеджер». Тепер на панелі інструментів має відображатися ESP8266[4].

4.2 Підключення бібліотек

В проекті було використано 7 бібліотек:

- Arduino_JSON;
- Arduino-Temperature-Control-Library;
- BME280;

- DHT sensor library;
- OneWire;
- painlessMesh;
- Wire.

Arduino_JSON – бібліотека для платформи Arduino, IoT і будь-якого вбудованого проекту C++. JSON файли можна серіалізувати та десеріалізувати за допомогою цієї бібліотеки.

Найпопулярнішим форматом для обміну та організації даних є JSON (JavaScript Object Notation). Як альтернатива XML, він переважно використовується для передачі даних між сервером і веб-програмою.

Arduino-Temperature-Control-Library – бібліотека для зчитування температурних показників. Бібліотека підтримує такі пристрої: DS18B20 DS18S20.

BME280 це бібліотека для програмно-апаратної пратформ Arduino для зчитування та інтерпретації даних з плати BME280 через програмні шини I2C, SPI або Sw SPI.

DHT sensor library – це бібліотека для зчитування температурних показників повітря та показників вологості повітря для DHT сенсорів.

Бібліотека OneWire надає процедури для зв’язку. OneWire – це протокол Master/Slave, і всі необхідні комунікаційні кабелі – це один дріт. Slave пристрої на шині OneWire можуть навіть отримувати живлення від лінії передачі даних.

PainlessMesh – це бібліотека, яка реалізує комірчасту мережу за допомогою обладнання esp8266 і esp32. Бібліотека створює комірчасту мережу та спрощує технічні нюанси для користувача[8].

Бібліотека Wire використовується для зв'язку мікроконтролера з пристроями та модулями через інтерфейс I2C. Лінія даних (SDA) і лінія тактового сигналу (SCL) є єдиними двома контактами, які використовуються для зв’язку за допомогою шини I2C. До відповідних роз'ємів мікроконтролера можна підключити до 120 пристроїв, що підтримують

інтерфейс I2C. Бібліотека Wire необхідна для зв'язку з такими пристроями та обміну даними.

4.3 Створення скетчу мовою C++

Спрощений алгоритм роботи прототипу пристрою реалізований за допомогою автомата Мілі, який зображено на рисунку 4.1, де a1 – отримання показників з датчиків, a2 – формування JSON рядка для передачі показників датчиків, a3 – відправлення JSON рядка іншим коміркам мережі, a4 – отримання повідомлення від іншої комірки мережі, a5 – обробка повідомлення, x – наявність вхідного повідомлення.

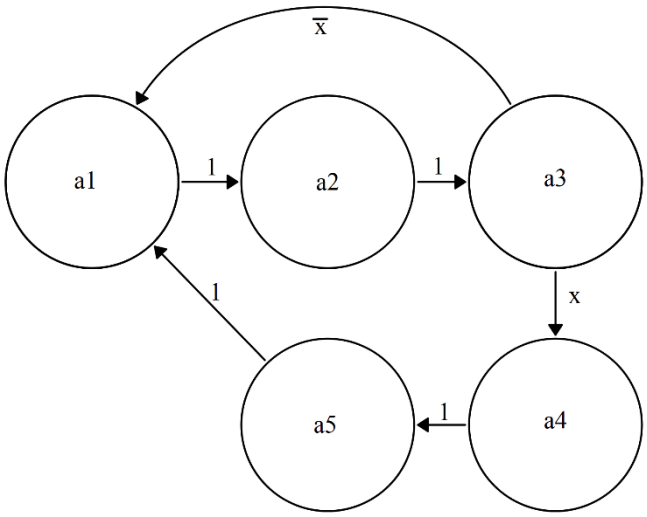

Рисунок 4.1 – Алгоритм у вигляді автомату Мілі

Повний код знаходиться в лістингу програмного коду (додаток А).

Підключення бібліотек. Вибір сенсору за допомогою розкоментування препроцесорної директиви "define". Реалізація описана в лістингу 4.1.

Лістинг 4.1 – Підключення бібліотек та вибір сенсору

```
#include <painlessMesh.h>
#include <Arduino_JSON.h>
#define BME_280
```

```
//#define DHT22
//#define DS18B20
//#define ENABLE_LOG
```

Далі у програмі використовуються макроси #ifdef і #endif щоб увімкнути чи вимкнути певні частини коду. Перша частина слугує для пристрою з датчиком BME280. Підключимо бібліотеки, необхідних для роботи з даним датчиком. Потім створимо об'єкт BME280I2C. Після цього поставимо змінні, з якими працюватиме об'єкт нашого класу. Реалізація описана в лістингу 4.2.

Лістинг 4.2 – Ініціалізація об’єктів для BME280

```
#ifdef BME_280
#include <BME280I2C.h>
#include <Wire.h>
BME280I2C bme;
BME280::TempUnit tempUnit(BME280::TempUnit_Celsius);
BME280::PresUnit presUnit(BME280::PresUnit_Pa);
#endif
```

Підключення бібліотек та ініціалізація об’єктів для пристрою з датчиком DHT22. Реалізація описана в лістингу 4.3.

Лістинг 4.3 – Ініціалізація об’єктів для DHT22

```
#ifdef DHT22
#include "DHT.h"
#define DHTPIN 4
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);
#endif
```

Підключення бібліотек та ініціалізація об’єктів для пристрою з датчиком DS18B20. Реалізація описана в лістингу 4.4.

Лістинг 4.4 – Ініціалізація об’єктів для DS18B20

```
#ifdef DS18B20
#include <OneWire.h>
#include <DallasTemperature.h>
const int oneWireBus = 4;
OneWire oneWire(oneWireBus);
DallasTemperature ds18b20(&oneWire);
#endif
```

Ініціалізуэмо рядкову змінну nodeName. Ця змінна слугує для ідентифікації вузлів в мережі, обов’язково повинна бути унікальною для кожного пристрою. Також ініціалізуємо змінні речовинного типу для зберігання даних температури, вологості та тиску.

```
String nodeName = "NODE_1";
float temp(NAN), hum(NAN), pres(NAN);
```

Встановлення параметрів доступу до комірчастої мережі – ідентифікатор мережі, пароль і номер порту. Ці параметри мають бути однаковими для всіх вузлів усередині нашої мережі.

```
#define   MESH_PREFIX     "someMesh"
#define   MESH_PASSWORD   "someMeshPass"
#define   MESH_PORT       5555
```

Створення об'єктів: Scheduler, painlessMesh та JSONVar. Створення прототипу функції sendMessage для подальшої передачі функції у задачу. Створення задачі у вигляді потоку, який завжди виконуватиметься і викликатиме функцію через деякий час. Задача використовуватиметься для

передачі широкомовних повідомлень (broadcast message) всім вузлам у мережі. Задача має три параметри. Перший визначатиме як часто задача викликатиме функцію, другий задаватиме час життя задачі (у нас це нескінченність), а третій містить покажчик на функцію, що викликається. Реалізація описана в лістингу 4.5.

Лістинг 4.5 – Ініціалізація об’єктів для комірчастої мережі та JSON формату

```
Scheduler userScheduler;
painlessMesh  mesh;
JSONVar myVar;
void sendMessage();
Task taskSendMessage( TASK_SECOND * 1 , TASK_FOREVER, &sendMessage );
```

Функція sendMessage передає повідомлення всім вузлам у мережі. Ця функція викликає іншу функцію, яка повертає рядок JSON. Реалізація описана в лістингу 4.6.

Лістинг 4.6 – Створення функції для роботи з JSON форматом

```
void sendMessage() {
String msg = construnct_json();
mesh.sendBroadcast( msg );
taskSendMessage.setInterval( random( TASK_SECOND * 1, TASK_SECOND * 5 ));
}
```

Функція receivedCallback виконує дії при прийомі повідомлення. У функції два параметри: ідентифікатор вузла (node id) та повідомлення.

```
void receivedCallback( uint32_t from, String &msg ) {
Serial.printf("startHere: Received from %u msg=%s\n", from, msg.c_str());
}
```

Функція newConnectionCallback викликається щоразу, коли до комірчастої мережі підключається новий пристрій. Після виклику вона передає відповідне повідомлення у вікно монітора послідовного зв'язку.

```
void newConnectionCallback(uint32_t nodeId) {
Serial.printf("--> startHere: New Connection, nodeId = %u\n", nodeId);
```

Функція nodeTimeAdjustedCallback викликається для завдання тимчасових зрушень (затримок), необхідних для відображення роботи прототипу пристрою.

```
void nodeTimeAdjustedCallback(int32_t offset) {
 Serial.printf("Adjusted time %u. Offset = %d\n", mesh.getNodeTime(), offset);
}
```

У функції setup ініціалізується послідовний зв'язок зі швидкістю 115200 бод і друкується ім'я вузла. Таким чином у вікні монітора послідовного зв'язку відображається робота нашої мережі. Також ми задамо клас setDebugMsgTypes для mesh об'єкта, який вестиме логи повідомлень ERROR | STARTUP. Функція init ініціалізує комірчасту мережу за допомогою відповідних параметрів (SSID, пароля та номера порту) . Реалізація описана в лістингу 4.7.

Лістинг 4.7 – Ініціалізація послідовного порту та об’єктів для роботи з комірчастою мережею

```
Serial.begin(115200);
Serial.println(nodeName);
mesh.setDebugMsgTypes( ERROR | STARTUP );
mesh.init( MESH_PREFIX, MESH_PASSWORD, &userScheduler, MESH_PORT );
```

Ініціалізація всіх функцій, що викликаються в процесі роботи комірчастої мережі. Ці функції будуть викликатись коли необхідно буде виконати якусь задачу. Реалізація описана в лістингу 4.8.

Лістинг 4.8 – Додавання функцій до планувальника завдань

```
mesh.onReceive(&receivedCallback);
mesh.onNewConnection(&newConnectionCallback);
mesh.onChangedConnections(&changedConnectionCallback);
mesh.onNodeTimeAdjusted(&nodeTimeAdjustedCallback);
```

Задача було додано до планувальник завдань (task scheduler) і планувальник завдань був увімкнений за допомогою методу taskSendMessage.enable. Після виконання цієї команди, різні завдання виконуватимуться одночасно у фоновому режимі.

```
userScheduler.addTask( taskSendMessage );
taskSendMessage.enable();
```

Також функція setup містить усі необхідні макроси ifdef and endif, які використовуються для ініціалізації різних датчиків у проекті. Ініціалізація для подальшої роботи з датчиком BME280. Реалізація описана в лістингу 4.9.

Лістинг 4.9 – Програмний код для роботи з датчиком BME280

```
#ifdef BME_280
  Wire.begin();
  while (!bme.begin())
  {
    Serial.println("Could not find BME280 sensor!");
    delay(1000);
  }
  // bme.chipID();  // Deprecated. See chipModel().
```

```
  switch (bme.chipModel())
  {
    case BME280::ChipModel_BME280:
      Serial.println("Found BME280 sensor! Success.");
      break;
    case BME280::ChipModel_BMP280:
      Serial.println("Found BMP280 sensor! No Humidity available.");
      break;
    default:
      Serial.println("Found UNKNOWN sensor! Error!");
  }
#endif
```

Ініціалізація для подальшої роботи з датчиками DHT22 та DS18B20 за допомогою відповідних макросів ifdef та #endif. Реалізація описана в лістингу 4.10.

Лістинг 4.10 – Програмний код для роботи з датчиками DHT22 та DS18B20

```
#ifdef DHT22
  Serial.println(F("DHTxx test!"));
  dht.begin();
#endif
#ifdef DS18B20
  ds18b20.begin();
#endif
```

В основній функції програми loop оновлення нашої комірчастої мережі за допомогою методу mesh.update, який працює з усіма запрограмованими нами задачами. Задачі не будуть виконуватись, якщо не викликати метод update.

```
void loop()
{
  mesh.update();
}
```

Функція construnct_json формує (конструює) JSON рядок і повертає його функції. Реалізація функції починається з виклику методу bme.read, яка зчитує з датчика BME280 необхідні параметри. Далі розділимо лічені значення тиску на 100 щоб отримати значення тиску в мілібарах. Потім ініціалізуємо масив JSON і введемо в нього ім'я датчика, ім'я вузла, дані температури і тиску і потім витягнемо ці дані з масиву за допомогою функції JSON.stringify для виведення їх на екран і повернення значення нашої функції construnct_json. Також було створено макроси ifdef і endif для того, щоб увімкнути або вимкнути логування наших параметрів. Реалізація описана в лістингу 4.11.

Лістинг 4.11 – Програмний код для роботи з JSON форматом

```
String construnct_json()
{
#ifdef BME_280
  bme.read(pres, temp, hum, tempUnit, presUnit);
  pres = pres / 100;
  myVar["Sensor Type"] = "BME280";
  myVar["Node Name"] = nodeName;
  myVar["Temperature"] = serialized(String(temp, 2));
  myVar["pres"] = serialized(String(pres, 2));
#ifdef ENABLE_LOG
  Serial.println(JSON.stringify(myVar));
#endif
  return JSON.stringify(myVar);
#endif
```

Таким же чином реалізовано програмний код для датчиків DHT22 та DS18B20 (додаток А).

4.4 Аналіз використання комірчастих самоорганізованих мереж

Було проаналізовано статтю «Performance Assessment of ESP8266 Wireless Mesh Networks». Отримані авторами цієї статті [9] результати можуть бути накладені і на розроблену модель.

Дослідження показало, що немає статистично значущої переваги у використанні повідомлень з великими розмірами (536 або 1460 байт); це вимагатиме великих обчислювальних потужностей. Крім того, як і очікувалося, на результати продуктивності вплинуло збільшення кількості сенсорних вузлів, швидкості надсилання повідомлень та розміру корисного навантаження повідомлення. Додавання другого та третього сенсорних вузлів до мережі зробило більший вплив на продуктивність, ніж додавання четвертого та п'ятого вузлів. Також було помічено, що, незалежно від кількості сенсорних вузлів у мережі, найкраща продуктивність досягається з повідомленнями розміром 250 байт. Що стосується затримки доставки, можна зробити висновок про високу волатильність, можливо, виправдану управлінням топологією мережі, що виконується бібліотекою painlessMesh.

Таким чином, можна зробити висновки, на даному етапі розвитку комірчасті мережі не підходять для передачі великої кількості інформації. Доказом цього є, наприклад, роботоспроможність чату на основі комірчастої самоорганізованої мережі під час масових мітингів в Гонконзі, Китай. Тому можна зробити логічний висновок, що і для приладів, які передають не велику кількість інформації (наприклад, системи клімат-контролю) доцільно використовувати комірчасті самоорганізовані мережі.

ВИСНОВКИ

У даній кваліфікаційній роботі було розглянуто та проаналізовано поняття клімат-контролю та розглянуто його основні складові. У ході роботи були створені модель та прототип приладу. Проаналізовано комірчасті самоорганізовані мережі та виявлено, що прилади з використанням комірчастих самоорганізованих мереж найкраще працюють для використання їх на великій площі, в місцях з невеликою кількістю точок доступу до мережі інтернет та в місцях з великою вірогідністю пошкодити прилад. Також було виявлено, що для комірчастих самоорганізованих мереж, побудованих на такому ж принципі як і в кваліфікаційній роботі, не важливий розмір мережевих пакетів, а важливі лише кількість переданих даних, кількість пристроїв та якість з'єднання.

Було проведено аналіз доступних компонентів та відібрано найкращі компоненти, за критерієм ціна-якість, які відповідають технічним вимогам.

Під час дослідження при використання комірчастих самоорганізованих мереж для систем клімат-контролю було виявлено такі переваги:

- збільшення дальності передачі даних завдяки збільшенню кількості пристроїв;
- невисока ціна;
- простота додавання нових сенсорів;
- самоорганізація мережі за розташуванням та навантаженням;
- невелике споживання енергії;
- надійність.

До недоліків використання комірчастих самоорганізованих мереж в комп'ютерних системах можна віднести: зменшення швидкості та збільшення затримок при передачі даних відносно мереж на базі протоколів TCP/IP.

Розроблений прототип можна вдосконалити наступним чином:

– додати LCD або OLED LCD дисплей;

– додати можливість моніторингу через мережу інтернет за допомогою WEB-сайту, мобільного застосунку або Telegram чи Discord боту;

– спроектувати друковані плати та розпаяти на них мікроконтролери та інші комплектуючі;

– реалізувати можливість автономного живлення за допомогою Li-ion або LiFePO4 акумуляторів;

– спроектувати корпуси для всіх компонентів для захисту від механічних пошкоджень, вологи та пилу;

– створення можливості підключення пропрієтарних датчиків;

– збільшення антени для збільшення дальності зв’язку.

Було створено та протестовано прототип приладу, побудованого на базі запропонованої моделі клімат-контролю. Те, що система клімат-контролю не використовує великі об'єми даних та не є критичною до затримок їх передачі через високу інертність показників клімату, обумовлює доречність використання комірчастих самоорганізованих мереж.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Arduino [Електронний ресурс]. – Режим доступу: https://www.arduino.cc/. – Дата доступу: 15.08.2022 р.

2. WeMos D1 mini Lite [Електронний ресурс]. Режим доступу: https://www.wemos.cc/en/latest/d1/d1_mini_lite.html. – Дата доступу: 15.08.2022 р.

3. ESP8266 (ESP-12E) [Електронний ресурс]. – Режим доступу: https://www.alldatasheet.com/datasheet-pdf/pdf/1179099/ETC2/ESP12E.html. – Дата доступу: 16.08.2022 р.

4. Arduino on ESP8266 [Електронний ресурс]. – Режим доступу: https://github.com/esp8266/Arduino#readme. – Дата доступу: 17.02.2021 р.

5. DS18B20 [Електронний ресурс]. – Режим доступу: https://html.alldatasheet.com/html-pdf/58557/DALLAS/DS18B20/181/1/DS18B20.html. – Дата доступу: 20.08.2022 р.

6. DHT22 [Електронний ресурс]. – Режим доступу: https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf. – Дата доступу: 23.08.2022 р.

7. BME280 [Електронний ресурс]. – Режим доступу: https://www.mouser.com/datasheet/2/783/BST-BME280-DS002-1509607.pdf. – Дата доступу: 03.09.2022 р.

8. PainlessMesh [Електронний ресурс]. – Режим доступу: https://gitlab.com/painlessMesh/painlessMesh. – Дата доступу: 03.09.2022 р.

9. Santos L. Soares Performance Assessment of ESP8266 Wireless Mesh Networks / Luís Santos, Tiago Costa, João M. L. P. Caldeira, Vasco N. G. J. // Information. – 2022. – № 13 (5). – P. 1-15.

10. Фрунзе А.В. Микроконтроллеры? Это же просто! / А.В. Фрунзе. – М.: ДДЭКА, 2015. – 312 с.

11. Таненбаум Э. С. Компьютерные сети. / Э. С. Таненбаум, Д. Уэзеролл. – 5-е изд. – СПб.: Питер, 2012. – 960 с.

12. Cilfone A. Wireless Mesh Networking: An IoT-Oriented Perspective Survey on Relevant Technologies / A. Cilfone, L. Davoli, L. Belli, G. Ferrari // Future Internet. – 2019. – № 11 (4). – P. 1-35.