Міністерство освіти і науки України

Луцький національний технічний університет
(повне найменування закладу вищої освіти)

Факультет комп'ютерних та інформаційних технологій
(повне найменування факультету)

Кафедра комп'ютерної інженерії та безпеки
(повне найменування кафедри)

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

ВБУДОВАНА СИСТЕМА УПРАВЛІННЯ ІОТ ДАТЧИКАМИ

EMBEDDED IOT SENSOR MANAGEMENT SYSTEM

спеціальність 123 Комп'ютерна інженерія
(шифр і назва спеціальності)

освітня програма Комп'ютерна інженерія
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІс-21
Воробйов Артем Олександрович

(підпис)

Керівник:
к.т.н., доцент
Костючко Сергій Миколайович

(підпис)

Кваліфікаційну роботу
допущено до захисту
« 10 » червня 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Лавренчук Світлана Василівна

(підпис)

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерної інженерії та безпеки*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *123 Комп'ютерна інженерія*
Освітня програма: *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ доц. Тарас ТЕРЛЕЦЬКИЙ
« 10 » 01 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Воробйову Артему Олександровичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Вбудована система управління IoT датчиками*

Керівник роботи *к.т.н., доц. Костючко Сергій Миколайович*

затверджені наказом закладу вищої освіти від «04» січня 2025 року № 11/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 10.06.2025р.

3. Вихідні дані до роботи *джерелом розробки є науково-технічна література та публікації в періодичних виданнях з даного питання, опубліковані зарубіжні та вітчизняні роботи в даній області та різні інтернет-ресурси технічного спрямування*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
Вступ
Аналіз проблем та огляд існуючих рішень
Теоретична частина
Практична частина
Висновки

5. Перелік графічного (ілюстративного) матеріалу:
Існуючі рішення
Використані технології
Архітектура системи
Схема роботи програмного продукту
Інтерфейс системи

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналітична частина	*Костючко С.М., доцент*		
Оцінювання та оптимізація вибору модулів для апаратної реалізації пристрою	*Костючко С.М., доцент*		
Розробка та дослідження запропонованого рішення	*Костючко С.М., доцент*		
Нормоконтроль	*Багнюк Н.В., доцент*		
Гарант ОП	*Лавренчук С.В., доцент*		
Показник запозичень тексту	***___%***		
Академічна доброчесність	*Міскевич О.І.,ст.викладач*		

7. Дата видачі завдання 10.01.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	*Огляд літератури із досліджуваної проблеми, аналіз предметної області та наявних рішень*	до 10.02.2025 р.	Виконано
2.	*Вибір мікроконтролера, модулів та програмного забезпечення для проєкту*	до 02.03.2025 р.	Виконано
3.	*Проєктування, підключення та налагодження коректної роботи пристрою*	до 02.04.2025 р.	Виконано
4.	*Висновки та пропозиції*	до 10.04.2025 р.	Виконано
5.	*Формування списку використаних джерел*	до 15.04.2025 р.	Виконано
6.	*Формування додатків*	до 02.05.2025 р.	Виконано
7.	*Оформлення ілюстративного матеріалу*	до 10.05.2025 р.	Виконано
8.	*Нормоконтроль*	до 15.05.2025 р.	Виконано
9.	*Інструментальна перевірка на академічний плагіат*	до 30.05.2025 р.	Виконано
10	*Представлення кваліфікаційної роботи бакалавра до захисту*	до 03.06.2025 р	Виконано
11.	*Здача кваліфікаційної роботи та всіх супровідних документів на кафедру*	до 10.06.2025 р.	Виконано

Здобувач вищої освіти _____ Воробйов А.О.
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ Костючко С.М.
(підпис) (прізвище, ініціали)

АНОТАЦІЯ

Воробйов А. О. Вбудована система управління IoT датчиками. Рукопис.

Кваліфікаційна робота бакалавра ОП «Комп'ютерна інженерія» спеціальності 123 Комп'ютерна інженерія. Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота складається з вступу, трьох розділів, висновків, списку використаних джерел, додатків.

В першому розділі здійнено огляд технологій та наявних рішень, було розглянуто поняття Інтернет речей, їх використання, захист від взлому та забезпення стабільної роботи, наявні існуючі рішення.

В другому розділі проведено аналіз наявних мікроконтролерів, модулів, датчиків, екранів та програмного забезпечення. Після здійснення вибору було обґрунтування.

Третій розділ присвячено опису та розробці апаратної та програмної частин. Візуалізовано роботу пристрою за допомогою спеціалізованої програми.

Ключові слова: Інтернет речей, мікроконтролер, вбудовані системи, пристрій, модуль, датчик, енергоефективність.

ANNOTATION

Vorobiov A. Embedded IoT sensor management system. Manuscript.

Qualifying work of a bachelor of EP «Computer Engineering» specialty 123 Computer Engineering. Lutsk National Technical University. Lutsk, 2025.

Qualification work consists of introduction, three chapters, conclusions, a list of references, and appendices.

The first section provides an overview of technologies and existing solutions. It examines the concept of the Internet of Things, its applications, protection against hacking, ensuring stable operation, and available existing solutions.

The second chapter presents an analysis of available microcontrollers, modules, sensors, displays, and software. After the selection process, the choices are substantiated.

The third section dedicated to the description and development of the hardware and software components. The operation of the device is visualized using a specialized program.

Keywords: Internet of Things, microcontroller, embedded systems, device, module, sensor, energy efficiency.

ЗМІСТ

ВСТУП

Актуальність теми. Пристрій який містить декілька функцій для різних сценаріїв повсякденного життя стане корисним помічником в побуті. Годинник з показниками навколишнього середовища в звичайному режимі, що допоможе зробити комфортнішим перебування вдома корегуючи певні параметри довкілля. Або ж використати як систему оповіщення при виявленні диму за допомогою звукового сигналу встановився в місцях потенційної небезпеки.

Вбудовані системи управління IoT датчиками дозволяють реалізувати енергоефективні рішення, за допомогою яких можна не лише збирати дані, а й реагувати на зміни в навколишньому середовищі. Розробка такого пристрою є доречною враховуючи тенденції сучасного світу, такі як мобільність, зручність, інформативність.

Особливо корисною така система буде в умовах житлового приміщення, де рівень комфортного життя є найважливішою умовою та напряму впливає на якість життя. Відстеження таких даних як температура, вологість та якість повітря в реальному часі дозволяє своєчасно вжити заходів для покращення умов або систематизувати певну дію, наприклад вмикати зранку зволожувач повітря.

Метою роботи є розробка пристрою для контролю параметрів повітря в приміщенні.

Об’єкт дослідження – проектування пристроїв на базі мікроконтролера.

Предмет дослідження – пристрій який керує IoT-датчиками.

Завдання, які необхідно виконати:

– аналіз існуючих рішень;

– дослідити доступні модулі та їхню роботу;

– візуалізувати роботу пристрою;

– спроектувати підключення модулів та їх розміщення у корпусі;

– реалізувати годинник з показниками повітря та системою оповіщення при виявленні диму;

– запропонувати можливість розширення функціоналу пристрою та керування ним.

«Матеріали дослідження були опубліковані в збірнику тез міжнародної науково-практичної конференції молодих вчених та студентів «Програмне та апаратне забезпечення в інформаційних технологіях » 6 травня 2025 року» [1].

РОЗДІЛ 1
АНАЛІТИЧНА ЧАСТИНА

1.1 Огляд теоретичного матеріалу про Інтернет речей

У сучасному світі цифрових технологій у багатьох людей відкривається доступ до речей, які полегшують життя виконуючи певні фукнції, наприклад розумні розетки з функцією ввімкнення в заданий час. Завдяки розвитку пристроїв Інтернет речей стали доступні такі пристрої за допомогою яких тепер створюються екосистеми розумного будинку. «IoT (Internet of Things) або Інтернет речей – система фізичних об’єктів, взаємопов’язаних між собою за допомогою вбудованих датчиків, програмного забезпечення та/або інших технологій» [2]. Такий зв’язок пристроїв дозволяє керувати як окремим вибраним пристроєм, так і сукупністю пристроїв, також це дає змогу одному датчику керувати іншим об’єктом чи навіть пристроєм за певних умов.

Поняття Інтернет речей існує понад 10 років, але розвиток технологій та глобальна тенденція надали можливість використовувати пристрої IoT не лише в спеціалізованих комп’ютерах розроблених під конкретні задачі, а впровадити в повсякденне життя безліч пристроїв без яких сучасне життя важко уявити. Це зумовлено зниженням ціни на виробництво та зменшенням необхідного для роботи розміру мікросхем, що дозволяє створювати системи різні по розміру, від декількох сантиметрів з низьким енергоспоживанням до великих стаціонарних систем.

Основною ідеєю Інтернет речей – це підключення фізичних пристроїв та датчиків до мережі для збирання та обміну інформацією. Для цього використовується спеціальне обладнання – сенсори, які фіксують різноманітні дані (температуру, вологість, рух, стан здоров’я, якість повітря тощо). Ці сенсори передають інформацію на центральні системи або інші пристрої, які аналізують дані та приймають рішення на їх основі.

У розумному будинку датчики можуть вимірювати температуру і вологість та на основі цієї інформації автоматично вмикати або вимикати систему

опалення чи вентиляцію. Такі системи дозволяють зробити наше життя комфортнішим та більш ефективним, економлячи енергію та ресурси.

Після збільшення різноманітності постала проблема доступу в інтернет. Інтернет речей використує багато відомих на даний час мережевих протоколів для максимальної сумісності як з новими мережами, так і для коректного функціонування в застарілих мережах. Використання будь-якого протоколу забепечує стабільне з'єднання з мережею Інтернет надаючи користувачеві стабільну роботу пристрою. «IoT – просте і плавне з'єднання фізичного світу бізнесу зі світом цифрових технологій» [2].

Основною причиною розвитку та популярності Інтернету речей є позитивний практичний результат його роботи. IoT додатки дозволяють:

– оптимізувати процеси постачання, обслуговування клієнтів, раціоналізація використання людського ресурсу та фінансових процесів;

– покращувати ефективність роботи;

– знаходити правильний підхід до клієнта, завдяки покращенню умов обслуговування;

– прискорювати процес прийняття рішення;

– аналізувати критичні показники, стан техніки та багато іншого

– збільшувати цінність бізнесу;

– знижувати кількість відходів;

– економити більше часу, передивитись використання коштів;

– знаходити вигідні пропозиції для дохідності;

– створювати нові бізнес-моделі;

«Мережа Інтернету речей складається з інтелектуальних пристроїв, які мають доступ до Інтернету» [2]. Зазвичай, це спроектована система, яка використовує для роботи різні пристрої: процесори, модулі, мережеве обладнання, датчики. Якщо така система підключена до мережі Інтернет або локальної мережі, вона відправляє зібрані дані з пристроїв на заданий хмарний сервер чи вказаний пристрій. Або ж інший сценарій, щоб спрацював потрібний механізм, IoT-пристрій налаштовується таким чином, що при спрацюванні

заданих користувачем тригерів буде приводитись в дію механізм завдяки обміну даними між цими пристроями.

Наступним кроком є обробка інформації в хмарному середовищі. Такі дії не потребують прямої участі користувача в даному процесі, але при потребі можна скоригувати потрібну інформацію або загалом переглядати забрану інформацію.

Інтернету речей спочатку був розроблений для сильно неоднорідних середовищ, де інформацію можна було збирати з різних джерел і обробляти різними технологіями. Тому схожі підходи, функції та сервіси можна згрупувати в одному шарі кожної запропонованої моделі IoT. Це спрощує розробку та вдосконалення кожного шару в майбутньому. Хоча трирівнева архітектура достатньо добре передає загальну концепцію IoT, вона є недостатньою для досліджень.

Стек IoT складається з п’яти рівнів:

– фізичний рівень, відповідає за виявлення фізичних характеристик об’єктів довкола. Цей рівень також перетворює інформацію в цифрові сигнали, зручні для передавання мережею. Важливими тут є нанотехнології, це дозволяє створювати мікроскопічні чипи, наприклад, для носимих пристроїв;

– канальний рівень відповідає за розмежування пакетів, синхронізацію, управління адресами відправника й отримувача, виявлення помилок у фізичному каналі та запобігання конфліктам;

– мережевий рівень забезпечує маршрутизацію даних у вигляді пакетів по всій мережі. До нього входить все мережеве обладнання;

– транспортний рівень співпрацює з прикладним для надійної передачі даних, забезпечуючи упорядкування пакетів, уникнення перевантаження, контроль цілісності та надійності даних;

– прикладний рівень надає інтерфейс та інструменти розробникам для створення застосунків.

Однією з ключових особливостей IoT-систем є можливість обміну даними між пристроями та серверними системами, для цього використовуються різноманітні протоколи зв’язку. «За останні роки з’явилося багато протоколів у зв’язку з розширенням застосувань та операцій в Інтернеті речей. Кожен із цих протоколів створений для задоволення потреб певного прикладного випадку або для забезпечення ефективної взаємодії в усьому екосистемному середовищі IoT. Така еволюція протоколів породжує численні набори правил і керівних положень. Багато з цих протоколів є складними для розуміння; деякі з них навіть є закритими та недоступними для широкої публіки. Через це вибір оптимального протоколу зв’язку для нового проєкту є складним завданням, а неправильне рішення може призвести до затримки запуску у виробництво та збільшення витрат на розробку»[3].

Wi-Fi є одним із найпопулярніших протоколів для підключення пристроїв до локальної мережі та Інтернету. Він забезпечує високу швидкість передачі даних доволі просту інтеграцію. Проте для автономних IoT-пристроїв використання Wi-Fi не завжди є оптимальним рішенням через підвищене енергоспоживання. Таке підключення ідеально підходить для пристроїв, які підключені до постійного джерела живлення, наприклад, розумних камер або стаціонарних сенсорних станцій.

Bluetooth та його модифікація Bluetooth Low Energy (BLE) застосовуються для передачі невеликих обсягів даних на короткі відстані. BLE відрізняється дуже низьким споживанням енергії, що робить цей стандарт необхідним для переносних пристроїв, таких як фітнес-трекери чи бездротові датчики в розумних будинках.

Zigbee – це протокол, спеціально розроблений для побудови розгалужених мереж IoT-пристроїв із низьким енергоспоживанням. Завдяки здатності створювати невеликі мережі зі схожими пристроями, в такій мережі можна передавати дані через проміжні вузли, збільшуючи таким чином дальність передачі без потреби в потужному сигналі. Це робить Zigbee хорошим вибором для автоматизації будівель та промислових об’єктів.

Таким чином, вибір протоколу зв’язку в IoT-системах залежить від багатьох факторів: енергоспоживання, відстані передачі даних, швидкості, кількості пристроїв у мережі та особливостей середовища експлуатації. Саме правильне рішення на цьому етапі значною мірою визначає стабільність і надійність роботи всього пристрою.

1.2 Вивчення теоретичних відомостей про вбудовані системи

Вбудовані системи – це апаратно-програмні комплекси, спеціально розроблені для виконання однієї або кількох функцій у складі більшої системи. Вони працюють у реальному часі та мають обмежені ресурси. «Архітектура вбудованих систем стосується каркасу та структури вбудованої системи, що охоплює її апаратні та програмні компоненти. На відміну від обчислювальних систем загального призначення, вбудовані системи призначені для виконання конкретних завдань, часто з обмеженнями реального часу. Архітектура визначає, як ці завдання керуються та виконуються, забезпечуючи ефективність, надійність і продуктивність системи» [4]. Типова структура вбудованої системи включає:

– центральний обчислювальний модуль – основний елемент, який керує обчислювальними процесами;

– оперативну та постійну пам’ять для зберігання даних і програмного коду;

– інтерфейси введення/виведення – для з’єднання з датчиками, приводами, індикаторами тощо;

– систему живлення – батарея або мережа, часто з енергозберігаючими компонентами;

– периферійні модулі це модулі зв’язку, таймери, аналого-цифрові перетворювачі тощо.

Ці системи можуть бути вбудовані у побутову техніку, автомобілі, медичне обладнання, промислові установки або сучасні IoT-пристрої. Зважаючи на різноманітність можливих сценаріїв використання вбудованих систем, важливу

роль у їх реалізації відіграють мікроконтролери. Саме вони забезпечують управління апаратними модулями, обробку сигналів з датчиків та реалізацію програмної логіки системи. Правильний вибір мікроконтролера має суттєвий вплив на енергоефективність, продуктивність, швидкість розробки й загальну стабільність IoT-рішень.

Для того, щоб вбудована система працювала правильно, необхідне спеціальне програмне забезпечення. За допомогою нього виконується програмування мікроконтролерів та налаштування різних сенсорів для збору та обробки даних. Крім того, для складних вбудованих систем можуть використовуватися операційні системи реального часу, які дозволяють організувати роботу пристроїв так, щоб вони реагували на зміни навколишнього середовища без затримок. Програмне забезпечення може також включати інтерфейси для управління пристроями через смартфони або комп’ютери.

Сучасні IoT-проєкти висувають до мікроконтролерів специфічні вимоги, які суттєво відрізняються від традиційних вбудованих систем. Передусім, мікроконтролери для IoT повинні мати підтримку енергоефективних режимів, вбудовані радіомодулі та засоби безпеки, гнучкі та певною мірою універсальні інтерфейси підключення, такі як I2C, SPI, UART.

Однією з головних переваг використання мікроконтролерів є їхня здатність працювати в умовах обмежених ресурсів – як з боку енергоспоживання, так і з боку обчислювальної потужності. Наприклад, більшість сучасних мікроконтролерів підтримують різні режими енергозбереження, що дає змогу зменшити споживання енергії до мінімального рівня у фазах очікування. Це дуже важливо для пристроїв, які працюють від акумуляторыв чи батарейок і повинні функціонувати протягом місяців або ще більш тривалий час без підзарядки.

При виборі мікроконтролера для IoT-пристроїв слід також враховувати архітектуру процесора. Однокристальні MCU забезпечують простоту та ефективність для базових завдань, тоді як багатоядерні архітектури пропонують гнучкість та оптимізацію для складніших застосувань. Наприклад, багатоядерні рішення можуть мати окремі ядра для обробки додатків, радіозв’язку та безпеки,

що дозволяє ефективніше розподіляти ресурси та покращувати продуктивність системи.

«Мікроконтролери (MCU) знаходять застосування в найрізноманітніших системах – від керування двигунами автомобілів та імплантованих медичних пристроїв до пультів дистанційного керування, офісної техніки, побутових приладів, електроінструментів, іграшок та інших вбудованих систем. Вони виконують роль посередника, забезпечуючи взаємодію між протоколами зв’язку та шарами абстракції апаратного забезпечення, а також підтримуючи запуск вибраної операційної системи реального часу (RTOS) або ОС, яка керує пристроєм»[5]. Це дозволяє значно скоротити час розробки готових продуктів, оскільки багато ключових функцій реалізовані на апаратному рівні й не потребують зовнішніх компонентів.

Завдяки великій різноманітності моделей і архітектур, користувач може підібрати мікроконтролер, який найкраще підійде вимогам конкретного проєкту. Деякі мікроконтролери орієнтовані на мінімальне енергоспоживання, інші – на високу продуктивність або на підтримку сучасних стандартів зв’язку. Наприклад, у задачах із обробкою мультимедійного контенту краще використовувати більш потужніші з апаратною підтримкою цифрової обробки сигналів, тоді як у пристроях для розумного будинку достатньо компактного та енергоощадного рішення.

Не менш важливим є розвиток спеціального програмного забезпення через популярність таких пристроїв, що значною мірою спрощує програмування мікроконтролерів. Завдяки зручному середовищу розробки, бібліотекам, які можуть писати інші користувачі та ділитись ними на форумах та прикладам коду, навіть початківці можуть створювати стабільно працюючі пристрої за доволі короткий проміжок часу. Більшість компаній, які займаються серійним виробництвом мікроконтролерів, надають безкоштовне програмне забезпечення, детальну технічну документацію, а також відкриті спільноти розробників.

Усі вище згадані пункти роблять мікроконтролери основною частиною сучасних вбудованих систем. Вони дозволяють створювати компактні, розумні,

автономні пристрої, що можуть збирати, обробляти й передавати інформацію в реальному часі. Без їх використання було б неможливо розробити більшість інноваційних рішень пов’язаних із автоматизацією, моніторингом або дистанційним керуванням.

1.3 Безпека в системах Інтернету речей

Інтернет речей (IoT) став важливою частиною нашого життя. Розумні будинки, пристрої для здоров’я, системи моніторингу довкілля працюють, збираючи та передаючи дані через мережу. Однак із широким розповсюдженням таких пристроїв виникла й нова проблема з безпекою.

З одного боку, IoT-пристрої створюють зручність і автоматизують багато процесів, проте з іншого – вони відкривають нові можливості для кіберзагроз. Багато пристроїв мають обмежені обчислювальні ресурси, через що їм важко підтримувати складні захисні механізми, які використовуються, наприклад, у комп’ютерах чи серверах. Саме тому атаки на IoT-системи іноді є набагато простішими для зловмисників.

Одним із головних ризиків є передача даних без належного захисту. Наприклад, розумний термостат або сигналізація можуть передавати інформацію про стан будинку без шифрування. Уявімо, що хтось перехоплює ці дані – він може дізнатися, коли нікого немає вдома. Саме тому застосування методів шифрування даних є обов’язковим кроком для захисту користувачів і їхнього майна.

«Обмеженість ресурсів та повсюдність IoT-мереж роблять їх вразливими до різноманітних атак. IoT-мережі зазвичай проєктуються з обмеженими обчислювальними та пам’яттєвими можливостями, що потенційно робить їх чутливими до маніпуляцій або порушень роботи. Крім того, велика кількість пристроїв, підключених до таких мереж, створює значну поверхню для атак, якою можуть скористатися зловмисники. Ці проблеми виникають через специфіку безпеки та способи її реалізації в IoT-середовищі» [6]. Ще одним

слабким місцем є відсутність оновлення програмного забезпечення пристроїв. Якщо виробник не передбачив можливість безпечного оновлення прошивки, пристрій залишається вразливим до старих атак, навіть якщо у світі вже давно існують виправлення. Зокрема, такі випадки ставали причиною створення великих ботнет-мереж, де заражені пристрої використовували для організації масованих кібератак на популярні інтернет-сервіси.

Серед основних загроз безпеці в системах Інтернету речей можна виділити кілька типових атак. Наприклад, атака типу відмова в обслуговуванні (DoS) може вивести пристрій з ладу, перевантаживши його запитами. Щоб мінімізувати ризики ще на етапі розробки пристроїв, застосовується принцип Безпека за задумом. Це означає, що захист повинен бути закладений у конструкцію пристрою з самого початку, а не додаватися пізніше як окрема функція. Наприклад, уже на етапі проєктування потрібно передбачити використання безпечних протоколів зв’язку, реалізацію перевірки автентичності користувача, шифрування даних на всіх етапах обробки та передачі.

Щоб уникнути таких проблем, розробники мають приділяти увагу не тільки функціоналу пристроїв, а й засобам автентифікації, перевірки цілісності даних та захисту від фізичного злому. Наприклад, деякі сучасні сенсори вже мають вбудовані механізми самостійної перевірки прошивки під час запуску, що ускладнює несанкціоноване втручання у їх роботу.

Питання безпеки в IoT є настільки широким і складним, що його розглядають як окремий напрям сучасної кібербезпеки. Багато дослідників наголошують, що майбутнє IoT безпосередньо залежить від здатності інженерів і розробників впроваджувати надійні захисні рішення з самого початку розробки пристроїв. Забезпечення конфіденційності, цілісності та доступності даних стає необхідною умовою для подальшого розвитку систем Інтернету речей у побуті, промисловості та міському середовищі.

1.4 Аналіз існуючих рішень

У сучасному світі зростає потреба в моніторингу якості повітря, що сприяє розвитку різноманітних пристроїв для цієї мети. Серед них виділяються як комерційні рішення, так і проєкти на основі відкритого апаратного забезпечення.

Комерційні пристрої зазвичай пропонують високий рівень точності та зручності використання. Наприклад, Air Quality Egg – це платформа Інтернету речей для моніторингу забруднення повітря, яка дозволяє користувачам відстежувати рівні різних забруднювачів у реальному часі. Цей пристрій підтримує підключення до Wi-Fi та може бути налаштований за допомогою мобільного телефону, що робить його зручним для широкого кола користувачів. Його зображено на рисунку 1.1.

Рисунок 1.1 – Пристрій Air Quality Egg [7]

Проєкти на основі Arduino також набули популярності завдяки своїй доступності та гнучкості. Один із таких проєктів – DIY Arduino Air Quality Monitor від HowToMechatronics, який використовує мікроконтролер Arduino, датчик температури та вологості DHT11, датчик якості повітря MQ135 для вимірювання рівня забруднення повітря та OLED-дисплей для відображення даних. Пристрій може попереджати користувачів про небезпечні рівні індексу якості повітря або концентрацію частинок. Цей пристрій дозволяє вимірювати

параметри навколишнього середовища, такі як леткі органічні сполуки озон, температуру та вологість та відображати їх у реальному часі, що робить його корисним для особистого використання та навчальних цілей. Даний проєкт продемонстровано на рисунку 1.2.

Рисунок 1.2 – пристрій DIY Arduino Air Quality Monitor [8]

Аналіз цих рішень показує, що існує широкий спектр пристроїв для моніторингу якості повітря, від комерційних продуктів до самостійно зібраних проєктів. Вибір конкретного рішення залежить від потреб користувача, бюджету та технічних навичок. У рамках даної роботи було вирішено розробити власний пристрій, який поєднує в собі функціональність, доступність та можливість подальшого розширення.

РОЗДІЛ 2

Оцінювання та оптимізація вибору модулів для апаратної реалізації пристрою

2.1 Комплексне дослідження та підбір модулів для апаратної реалізації пристрою

Найголовніше це плата-контролер яка буде керувати, зчитувати дані модулів. Оскільки в майбутньому пристрої планує багато датчиків та можливість розширення функціоналу, потрібно обирати доступне рішення, яке підтримується спільнотою та мало необхідну потужність для обробки всієї інформації. Підходящою платформою є Arduino. «Плата Arduino – це всього лише мозок пристрою. Для повноцінної роботи потрібні датчики і виконуючі пристрої. Їх підключають до плати Arduino через спеціально відведені контакти вводу-виводу» [9], які підтримують широко імпульну модуляцію. Розглянемо популярні наявні моделі.

Для компактних пристроїв добре підходить плата Arduino Nano, на ній є необхідні контакти для підключення датчиків. На даний момент існує декілька комплектацій цієї моделі, від звичайної плати, до вбудованих мережевих та інших модулів. Розглянемо характеристики Arduino Nano 33 BLE Rev2:

– мікроконтролер: nRF52840;

– USB-роз’єм: Мікро-USB;

– аналогові вхідні контакти: 8;

– контакти ШІМ: Усі цифрові контакти (4 одночасно);

– датчик акселерометру та гіроскопу IMU BMI270 та магнітометр BMM150;

– комунікація: UART, I2C, SPI;

– вхідна напруга 5-18 В;

– тактова частота процесора 64 МГц;

– наявна 256 КБ SRAM та 1 МБ флеш-пам’ять;

– маса 5 грам;

– довжина 45 мм;

– ширина 18 мм.

Така плата підходить для невеликих пристроїв з декількома датчиками, оскільки кількість контактів обмежена. Перевагою є наявність вбудованого гіроскопу та акселерометру, що значно полегшить розробку проєктів з даними датчиками. На рисунку 2.1 можна розглянути компонування всіх компонентів та наявність розпаяних конекторів, що значно полегшить розробку виключаючи необхіність в паянні датчиків до плати.

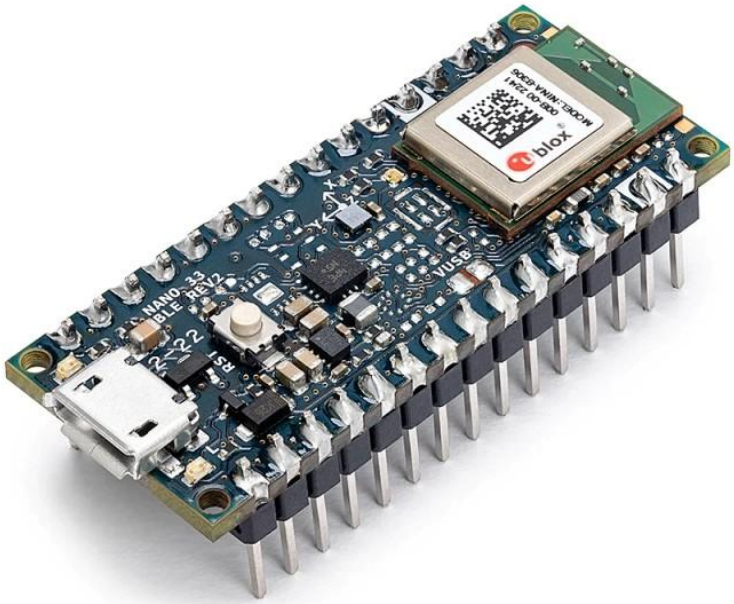

Рисунок 2.1 – Arduino Nano 33 BLE Rev2 [11]

Наступною платою трохи більшою по розмірах виступає Arduino UNO R4. Розглянемо технічні характеристики Arduino UNO R4 Wi-Fi:

– мікроконтролер Renesas RA4M1;

– допоміжний мікроконтролер ESP32-S3;

– USB-роз’єм: USB-C;

– аналогові вхідні контакти: 6;

– цифрові контакти вводу/виводу: 14;

– цифро-аналоговий перетворювач: 1;

– вбудований модуль Wi-Fi та Bluetooth;

– комунікація: UART, I2C, SPI, CAN;

– тактова частота Renesas RA4M1 – 48 МГц;

– тактова частота ESP32-S3 – 240 МГц;

– наявна 32 КБ SRAM та 512 КБ флеш-пам’ять Renesas RA4M1;

– наявна 512 КБ SRAM та 384 КБ флеш-пам’ять ESP32-S3;

– вхідна напруга 6-24 В по лінії VIN;

– довжина 69 мм;

– ширина 53 мм.

Також є особливості, такі як два мікроконтролера на одній платі, що дозволяє гнучко використовувати ресурси. По-замовчуванню основним використовується Renesas RA4M1 через його точність та стабільність, а ESP32-S3 використовується як допоміжний та через нього виконується зв’язок по бездротовому каналу. Якщо ж ресурсів основного мікроконтролера не вистачить, можна використовувати лише ESP32-S3, який має в п’ять разів швидшку тактову частоту, але якщо не врахувати охолодження, можуть з’явитись проблеми з перегрівом. Якщо ж довгий час використовувати цей мікроконтролер, можуть проявлятись зависання або збої. Розглянемо плату ближче на рисунку 2.2.

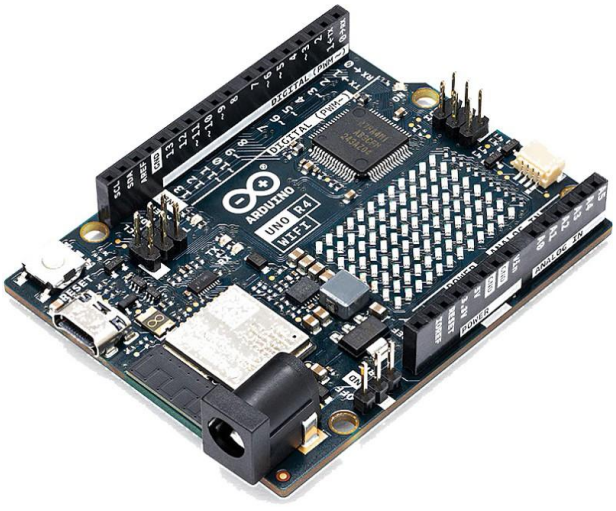

Рисунок 2.2 – Arduino UNO R4 Wi-Fi [13]

Непоганим бонусом є наявність розпаяної світлодіодної матриці розміром 12 на 8, вона може допоміжною матрицею при реалізації проекту. Як і в попередньої моделі, тут вже розпаяні конектори.

Наступною буде Arduino Due, це наступник добре відомої моделі Arduino Mega. Вона вибирається зазвичай для дуже великих проектів, оскільки вона громіздка через велику кількість входів. Розглянемо технічні характеристики Arduino DUE:

– мікроконтролер AT91SAM3X8E;

– вхідна напруга по VIN 7-12 В;

– аналогові вхідні контакти: 12;

– цифрові контакти вводу/виводу: 54;

– цифро-аналоговий перетворювач: 2;

– цифрових контактів з ШІМ: 16;

– тактова частота мікроконтролера 84 МГц;

– Наявна 96 КБ SRAM та 512 КБ флеш-пам’ять.

Як видно з характеристик плата нічим особливо не виділяється, немає вбудованих модулів чи датчиків, лише дуже велика кількість контактів вводу/виводу. Модуль продемонстровано на рисунку 2.3.

Рисунок 2.3 – Arduino DUE [13]

З огляду на подані характеристики різних плат, найкращим вибором буде Arduino UNO R4 Wi-Fi через непогану потужність, стабільність, набір контактів та вбудовану бездротову мережу. Мікроконтролер від компанії Renesas є сертифікованим і має хорошу точність та стабільність, яку не зможе надати ESP32-S3. Великий набір конекторів надасть змогу підключити доволі багато датчиків, в деяких випадках більше 12. Вбудований Wi-Fi та наявні ресурси мікроконтролера дозволяють створювати навіть невеликі сайти.

Для контролю навколишнього середовища потрібно вибрати датчики які зчитують температуру, вологість, якість повітря.

Для початку розберемо які є датчики температури, їхню якість, точність та споживання. Найдешевшим є датчик температури DS18B20, у нього є невелика похибка 0,5° С, що не є критично для домашнього використання, має широкий діапазон вимірювання від -55 до 125° С, можна заживити як від живлення 3,3 В так і 5 В, підключається легко, лише одним контактом. При роботі може трохи нагріватись показуючи вищу температуру. Для включення його в схему потрібно використовувати резистор, що може дещо ускладнити роботу з ним. Датчик продемонтровано на рисунку 2.4.

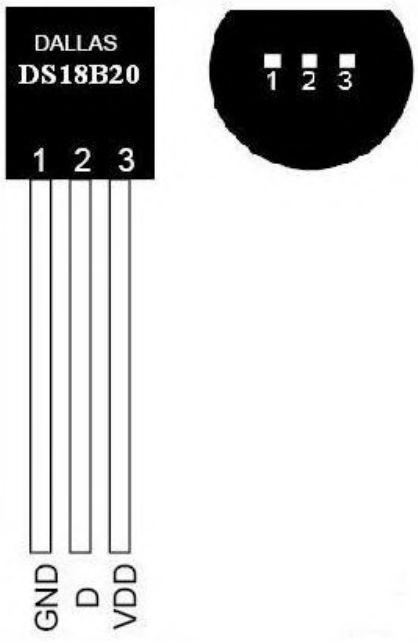

Рисунок 2.4 – Датчик температури DS18B20 [14]

Наступним буде датчик який вимірює температуру та вологість повітря DHT11, він трохи дорожче, має трохи більшу похибку, до 1° С та до 5 % волологості повітря. Можна заживити як й попередній датчик. Має один вихід по якому знімаються дані. Для підключення не обов’язково використовувати резистори. Даний датчик продемонтровано на рисунку 2.5.

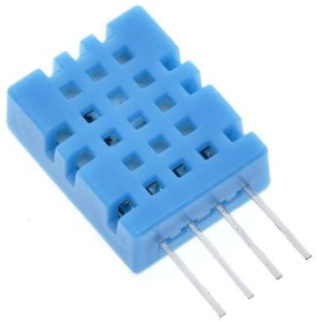

Рисунок 2.5 – Датчик температури та вологості DHT11 [15]

Покращена версія попереднього варіанту буде датчик температури та вологості SHT30. Хоча він коштує більше, але в нього значно менша похибка, всього 0,3 °С в діапазоні вимірювання 5-60° С та зменшена похибка в вимірюванні вологості повітря – 3 %. Діапазон живлення даного датчика становить 2,2-5,5 В. Підключається по шині I2C, що дещо полегшує написання коду та використання датчика вцілому. Датчик готовий до використання відразу, не потрібно щось підключати додатково. Датчик продемонстровано на рисунку 2.6.

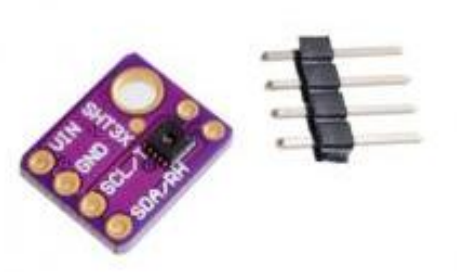

Рисунок 2.6 – Датчик температури та вологості SHT30 [16]

Не менш цікавим буде модуль BME280 в якому об’єднано цілих три датчик, це термометр, датчик вологості повітря та барометр. Хоч він і дорожче, але підключається за допомогою шини I2C, що дозволяє економити обмежену кількість конекторів. Має трохи більшу похибку в терметрів від 0,5 до 1 °C при діапазоні від -40 до 85 °C, така ж похибка 3 % при вимірюванні вологості повітря та похибку 1гПа з діапазоном від 300 до 1100 гПа, що при середніх значеннях ~990 гПа становить менше 0,2 %, що є хорошим результатом. Живиться даний модуль напругою від 1,8 до 5 В. Максимальний струм в режимі вимірювання становить 1.4 мА, а в режимі сну споживання до 0,5 мкА, що значить про хорошу енергоефективність модуля, враховуючи, що більшість часу модуль буде в режимі очікування. Даний модуль продемонстровано на рисунку 2.7.

Рисунок 2.7 – Модуль BME280 [17]

Є моделі значно дорожчі, але й вони набагато точніші та відразу готові до роботи, наприклад датчик вологості та температури DHT22. Його точність вимірювання температури 0,1° С, вологості повітря до 2 %, діапазон напруги живлення 3,6-6 В. Не потребує додаткових резисторів чи будь-чого іншого, має дуже мале енергоспоживання. Єдиний недолік – це віднутність шини I2C, через що даний датчик займає два контакти. Датчик продемонстровано на рисунку 2.8.

Рисунок 2.8 – Датчик температури та вологості DHT22 [18]

Проаналізувавши наявні рішення, найкраще підійде модуль BME280, оскільки в ньому об’єднанно три датчики та він легко підключається за допомогою шини I2C. Він добре підійде в даному випадку через те що не потрібна висока точність вимірювання. Отже, при створенні пристрою буде використовуватись саме цей модуль.

Для оцінки якості повітря було обрано датчик AGS02MA. «Кожен датчик повністю калібрується та тестується під час виробництва. AGS02MA розроблений для виявлення та моніторингу різних летких органічних газів, таких як етанол, аміак, сульфід, пари бензолу, дим» [19]. Підключається за допомогою шини I2C та не потребує додаткових компонентів для його встановлення, що робить його використання легким. Датчик продемонстровано на рисунку 2.9.

Рисунок 2.9 – Датчик якості повітря AGS02MA [19]

Не буде зайвим використати датчик газу та диму для додаткової безпеки в приміщенні. Є декільки датчиків газу MQ-2, MQ-4, MQ-5, MQ-7, MQ-8, MQ-9. Кожен з них має чутливість до різного типу газу. Для відображення роботи буде використовуватись світлодіод та пищалка, які будуть приведені в дію у випадку спрацювання датчика. У таблиці 2.1 наведено на який газ розрахований датчик та деякі технічні характеристики.

Таблиця 2.1 – Характеристики модельного ряду MQ датчиків газу

Модель	Цільовий газ	Діапазон чутливості
MQ-2	Горючий газ, дим	300 - 10000 ppm
MQ-4	Природний гах та метан	300 - 10000 ppm
MQ-5	Метан, ізобутан	5000 ppm
MQ-7	Чадний газ	10 - 1000 ppm
MQ-8	Водень, коксовий газ	100 - 10000 ppm
MQ-9	Зріджений нафтовий газ, ізобутан, метан, алкоголь, гідроген, дим	500 - 10000 ppm

Джерело:[20], [21], [22], [23], [24], [25]

Щоб моніторити якість освітлення необхідно додати датчик TEMT6000. Це недорогий та хороший датчик що дозволить проводити вимірювання освітлення в діапазоні від 0 до 10000 люкс, чого буде достатньо для кімнати.

Рисунок 2.10 – Датчик освітленості TEMT6000 [26]

Для виведення отриманих даних з датчик можна використати дисплей. Оскільки пристрій не має бути громіздкий, можна розглянути невелику діагональ. Це також дозволить зменшити енергоспоживання.

Розглянемо кольоровий дисплей 1,14 дюйми з рошириенням 240 на 135 пікселів. Можливою перевагою буде невелика колірна палітра на 65000 кольорів.

Тим матриці – IPS, що не є дуже енергоефективно. Підключається такий дисплей не дуже зручно, за допомогою SPI інтерфейсу, що потребує багато контактів. Вхідна напруга від 3,3 до 5 В. На такий екран буде зручно виводити невелику кількість даних з датчиків. Дисплей продемонстровано на рисунку 2.11.

Рисунок 2.11 – SPI дисплейний модуль 240x135 від Waveshare [27]

Дешевшим варіантом буде звичайний OLED-дисплей, він має трохи меншу діагональ – 0,96 дюйми та меншу роздільну здатність – 128 на 64 пікселі. Підключається значно легше ніж попередній, за допомогою шини I2C, що буде використовувати всього два контакти на платі, до яких можна підключити інші пристрої на аналогічній шині. Завдяки матриці OLED споживання енергії буде менше, оскільки не потрібно використовувати зайві пікселі. Живиться даний екран від 3.3 В до 6 В, що дає трохи запасу по верхній границі. Дисплей продемонстровано на рисунку 2.12.

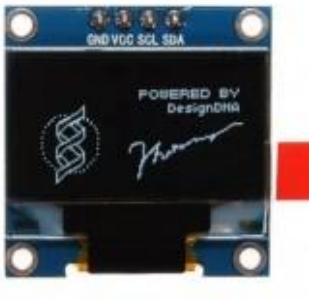

Рисунок 2.12 – OLED дисплей 0,96 дюйми 128 на 64 пікселі [28]

Отже, краще вибрати OLED дисплей через його зручність підключення та менше споживання енергії.

Для покращення часу автономної роботи можна додати датчик перешкоди для увімкнення дисплею лише за потреби. Наприклад, додати до схеми модуль YL-63, що дозволить виявляти перешкоди на відстані від 2 до 3 сантиметрів. Він базується на принципі відбиванні інфра-червоного світла, що робить неможливим використання такого датчика в місцях де попадає пряме сонячне світло. Необхідна напруга живлення від 3,3 до 5 В. Датчик продемонстровано на рисунку 2.13.

Рисунок 2.13 – Датчик перешкоди YL-63 [29]

Якщо ж на пристрій буде попадати сонячне світло, можна використати ультразвуковий датчик, наприклад US-025. Принцип роботи базується на відбиванні ультразвукових імпульсів, що дозволяє налаштовувати відстань спрацювання від 0 до 150 сантиметрів, на практиці можна й дальше фіксувати перешкоду. Якщо пристрій буде розташований на столі, це дозволить вмикати дисплей, коли ви сідаєте за стіл. Недоліком є більший розмір, порівняно з попереднім датчиком. На рисунку 2.14 продемонстровано даний датчик.

Рисунок 2.14 – Ультразвуковий датчик відстані US-025 [30]

2.2 Вибір програмного забезпечення

Для програмування вбудованих систем зазвичай використовуються спеціалізовані середовища розробки, які забезпечують зручне написання, компіляцію та завантаження коду на мікроконтролер. Одним із найпопулярніших інструментів у цій сфері є Arduino IDE.

Arduino IDE – це безкоштовне та відкрите середовище розробки, спеціально створене для роботи з платами Arduino. Завдяки простому та інтуїтивному інтерфейсу Arduino IDE підходить як для початківців, так і для досвідчених розробників. Програмування у цьому середовищі здійснюється мовою, заснованою на C/C++, з використанням спрощеної структури та великої кількості готових бібліотек.

Встановити Arduino IDE досить легко, програму потрібно завантажити з офіційного сайту Arduino. Після встановлення та запуску в середовищі необхідно вибрати та завантажити системні бібліотеки для конкретної моделі плати, встановити для неї драйвери в автоматичному режимі та вибрати порт підключення.

Однією з ключових можливостей Arduino IDE є підтримка серійного монітора, який дозволяє вести діалог між комп’ютером і мікроконтролером у режимі реального часу. Це дає змогу зручно тестувати роботу програми, переглядати змінні, отримувати дані з сенсорів або налагоджувати код без необхідності в складних апаратних засобах.

Ще однією важливою особливістю Arduino IDE є система керування бібліотеками. Бібліотеки дозволяють швидко підключати підтримку різних сенсорів, дисплеїв, модулів зв’язку та інших компонентів. Завдяки великій кількості готових рішень можна значно скоротити час розробки та зосередитись на створенні логіки роботи пристрою.

Інтерфейс складається з редактора коду з підсвічуванням синтаксису, консолі повідомлень та набору інструментів для компіляції, завантаження коду і

моніторингу роботи пристрою. Інтерфейс програми можна розглянути на рисунку 2.15.

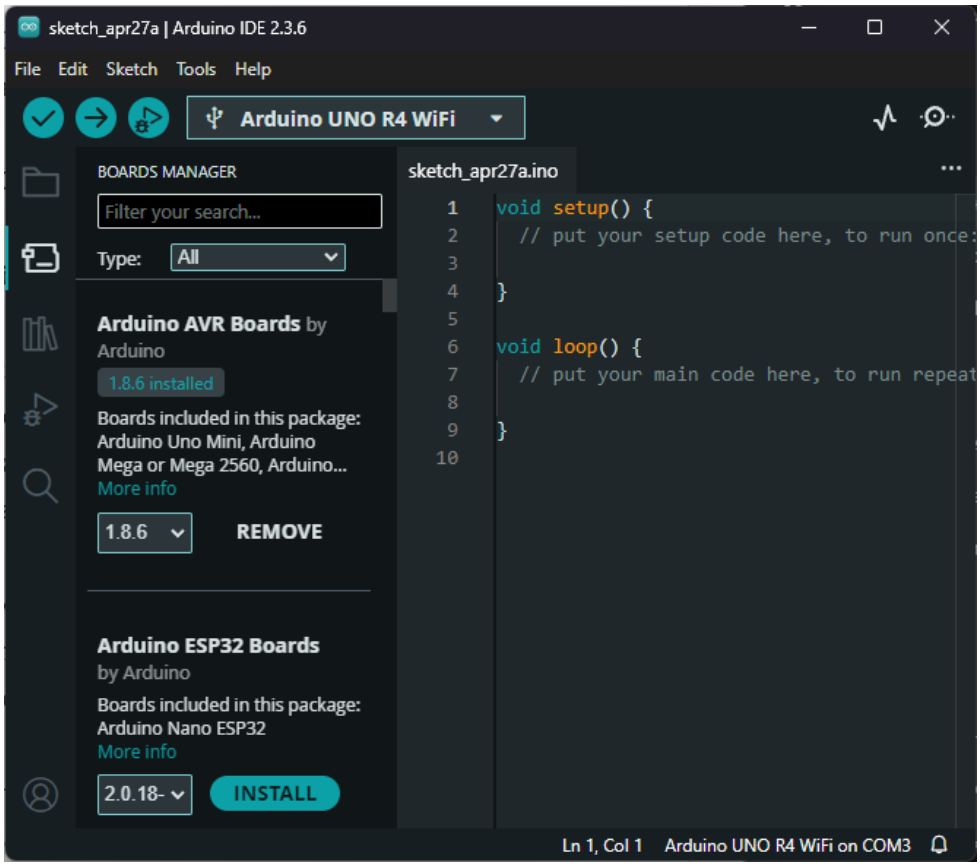

Рисунок 2.15 – Інтерфейс програми Arduino IDE

Завдяки відкритій платформі Arduino IDE постійно розвивається, з’являються нові версії середовища, які підтримують сучасні плати та розширені можливості, зокрема роботу через Wi-Fi або Bluetooth. Саме Arduino IDE було обрано для реалізації проєкту у цій кваліфікаційній роботі, оскільки воно забезпечує стабільність роботи з мікроконтролерами Arduino, має великий набір бібліотек та документації, а також підтримує підключення сторонніх модулів, необхідних для побудови системи на базі Інтернету речей.

У процесі розробки електронних пристроїв важливо не лише створити працездатну схему, але й мати можливість її наочно представити та задокументувати. Для цього використовується програмне забезпечення Fritzing,

яке дозволяє створювати візуальні схеми підключення компонентів на макетній платі.

Fritzing – це умовно безкоштовне програмне забезпечення з відкритим кодом, розроблене для спрощення процесу створення електронних схем, особливо для початківців, студентів та ентузіастів. Програма надає можливість створювати тривимірні зображення макетних плат, що значно полегшує розуміння структури проекту та сприяє ефективному навчальному процесу.

Основною перевагою Fritzing є інтуїтивно зрозумілий інтерфейс, який дозволяє користувачам перетягувати компоненти на робоче поле, з’єднувати їх провідниками та створювати повноцінні схеми. Програма підтримує три основні режими перегляду: макетна плата, принципова схема та друкована плата, що забезпечує повний цикл розробки – від прототипу до готового виробу.

Крім того, Fritzing дозволяє експортувати створені схеми у різних форматах, таких як PNG, SVG або PDF, що є зручним для включення ілюстрацій у документацію, презентації або дипломні роботи. Також існує можливість створення власних компонентів та оновлення існуючої бібліотеки, що робить програму гнучкою та адаптивною до потреб користувача.

Завдяки своїй простоті та функціональності, Fritzing стала популярним інструментом у сфері освіти та серед розробників, які працюють з платформами Arduino, Raspberry Pi та іншими мікроконтролерами.

РОЗДІЛ 3

Розробка та дослідження запропонованого рішення

3.1 Постановка завдання

У рамках даної кваліфікаційної роботи було поставлено завдання розробити вбудовану систему, здатну виконувати функції моніторингу навколишнього середовища та сигналізації про небезпечні зміни параметрів. Пристрій повинен забезпечувати вимірювання температури, вологості повітря, атмосферного тиску, рівня освітленості та якості повітря. Окрім цього, система має бути здатною виявляти наявність диму для своєчасного попередження про потенційну пожежну небезпеку.

Окремою важливою функцією пристрою є відображення поточного часу на OLED-дисплеї, що забезпечується використанням модуля реального часу та синхронізацією через мережу Інтернет. Також система повинна бути енергоефективною, працювати автономно від акумулятора, підтримувати бездротове з’єднання Wi-Fi для можливості віддаленого доступу до даних. Для зменшення споживання енергії, додано датчик перешкоди, щоб вмикати дисплей на певний час лише коли приблизитись до такого датчика.

Основними вимогами до пристрою є достатня точність сенсорних вимірювань, енергоефективність, компактні габарити, можливість автономної роботи від акумуляторного живлення та підтримка бездротового з’єднання для спрощення інтеграції у побутові чи офісні умови. Пристрій має працювати стабільно в умовах звичайного житлового приміщення при температурі від 5° C до 40° C і відносній вологості до 80 %.

Розробка має передбачати компактність пристрою, зручність використання, а також можливість візуального оповіщення користувача про зміну стану середовища, наприклад, за допомогою дисплея або звукового сигналу у разі виявлення диму.

Для реалізації проєкту було обрано платформу Arduino Uno R4 Wi-Fi, яка завдяки вбудованому модулю Wi-Fi та широкому набору підтримуваних

периферійних пристроїв ідеально підходить для побудови компактних IoT-рішень. Використання стандартних сенсорних модулів дозволяє скоротити час розробки та підвищити надійність готового пристрою.

3.2 Розробка апаратної частини

У рамках розробки пристрою першим кроком стало формування апаратної бази, яка включає вибір мікроконтролера, сенсорних модулів, відображувальних пристроїв та допоміжних елементів, необхідних для забезпечення стабільної роботи системи.

В якості основного обчислювального модуля було обрано плату Arduino Uno R4 Wi-Fi. Це сучасна плата, яка базується на продуктивному мікроконтролері Renesas RA4M1 ARM Cortex-M4 та має вбудований модуль Wi-Fi. Використання цієї плати дозволило спростити інтеграцію з бездротовими мережами та забезпечити необхідний рівень обчислювальної потужності при невеликому енергоспоживанні. Перевагою Arduino Uno R4 Wi-Fi є наявність широкого набору цифрових і аналогових портів введення/виведення, що дало змогу підключити декілька сенсорів та додаткових пристроїв одночасно та залишився запас на майбутні розробки.

Для вимірювання параметрів навколишнього середовища використовувався модуль BME280, який об’єднує в собі сенсори температури, вологості та атмосферного тиску. Цей модуль має цифровий інтерфейс I2C, що дозволило підключити його без значного навантаження на апаратні ресурси мікроконтролера. BME280 забезпечує достатню точність вимірювань і має низьке енергоспоживання, що є критичним для автономної роботи пристрою.

Контроль рівня освітленості в приміщенні реалізовано за допомогою датчика TEMT6000. Це фототранзисторний сенсор, що забезпечує перетворення інтенсивності світлового потоку в аналоговий сигнал. Він дозволяє відслідковувати зміну рівня освітлення для коригування освітлення до комфортного рівня.

Для виявлення наявності диму у повітрі було обрано датчик MQ-2. Цей модуль здатний виявляти присутність частинок диму та інших забруднень у повітрі на основі зміни електропровідності. Сигналізація при виявленні диму реалізується через генерацію звукового сигналу або виведення попереджувального повідомлення на екран.

Крім того, для контролю загального стану якості повітря використовується сенсор AGS02MA, який вимірює концентрацію летких органічних сполук. Датчик підключається через інтерфейс I2C та забезпечує швидке та енергоефективне вимірювання показників якості повітря, що є важливим у побутових та офісних умовах.

Для виведення інформації застосовується OLED-дисплей із роздільною здатністю 128 на 64 пікселі. Цей тип дисплеїв забезпечує високу чіткість зображення при мінімальному енергоспоживанні. Завдяки компактним розмірам OLED-дисплея вдалося зберегти зручність використання пристрою при збереженні компактності всієї системи.

Усі компоненти було підключено до макетної плати, що дозволило швидко змінювати конфігурацію схеми під час тестування і налагодження. Компоненти з’єднувалися через спеціальні кабелі. Живлення пристрою передбачено через два акумулятори типу 18650 з’єднані послідовно, що дасть великий запас енергії та довгий час роботи без додаткового живлення. Також передбачений режим роботи від мережі за допомогою підключення кабелю.

Таким чином, розроблена апаратна частина повністю відповідає поставленим вимогам до функціональності, енергоефективності та зручності використання пристрою у складі системи Інтернету речей.

У процесі розробки апаратної частини для зручності монтажу та налагодження була створена візуальна схема підключення компонентів. Для цього використовувалось спеціалізоване програмне забезпечення Fritzing, яке дозволяє швидко моделювати розташування елементів на макетній платі.

На схемі зображено підключення мікроконтролерної плати Arduino Uno R4 Wi-Fi до основних функціональних модулів:

– сенсора температури, вологості та тиску BME280 через шину I2C;

– фотосенсора TEMT6000 через аналоговий вхід A0;

– сенсора диму US-025 через цифровий порт;

– сенсора якості повітря AGS02MA через шину I2C;

– OLED-дисплея 128×64 через ту саму I2C-лінію.

На рисунку 3.1 представлена схема підключення модулів до Arduino та макетної плати.

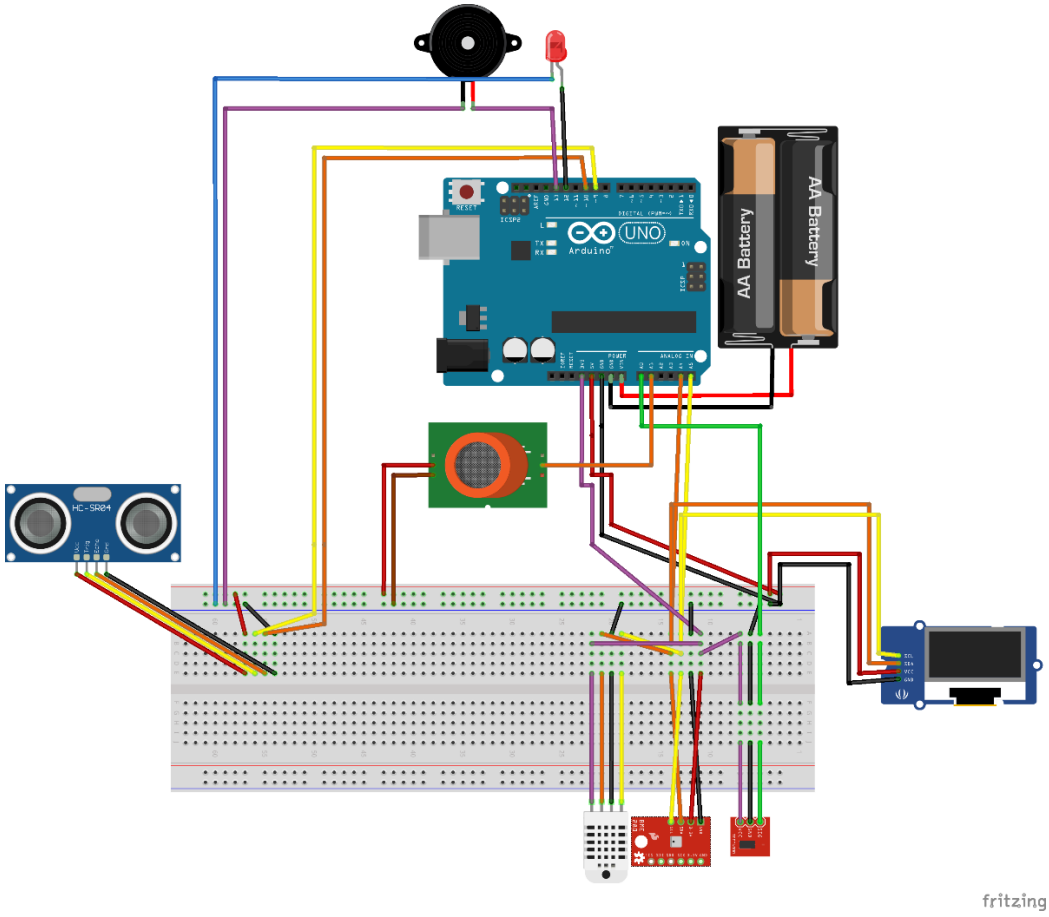

Рисунок 3.1 – Схема підключення в Fritzing

3.3 Розробка програмного забезпечення

Для забезпечення роботи розробленого пристрою було створене спеціалізоване програмне забезпечення, яке реалізує зчитування даних із сенсорів, обробку інформації, виведення результатів на OLED-дисплей та виконання функцій сигналізації.

Програмування здійснювалось за допомогою середовища розробки Arduino IDE. Це середовище забезпечує зручне написання, компіляцію та завантаження програмного коду безпосередньо на мікроконтролер Arduino Uno R4 Wi-Fi. Для реалізації функціоналу використовувалась мова програмування C++ із застосуванням відповідних бібліотек для роботи з периферійними пристроями.

Основною задачею програмного забезпечення було забезпечити періодичне зчитування даних з наступних сенсорів:

– BME280 – температура, вологість, атмосферний тиск;

– TEMT6000 – рівень освітленості;

– MQ-2 – детектор газу та диму;

– AGS02MA – показники якості повітря.

Для взаємодії із сенсорами використовувались стандартні бібліотеки, такі як WiFi.h, Wire.h, RTC.h, а також спеціальні бібліотеки для модулів які цього потребують, зокрема GyverOLED.h, AGS02MA.h, GyverBME280.h для ефективного виведення інформації.

Структура програмного забезпечення передбачає виконання основних етапів роботи:

– у функції setup() налаштовуються всі порти, ініціалізуються бібліотеки сенсорів та OLED-дисплея, встановлюються початкові параметри шини I2C;

– у циклі loop() здійснюється періодичне опитування сенсорів для отримання актуальних значень параметрів навколишнього середовища;

– отримані значення аналізуються для виявлення перевищення допустимих меж, наприклад, при виявленні диму або погіршенні якості повітря;

– результати вимірювань відображаються на OLED-дисплеї у зручному для сприйняття форматі, а також на веб-сторінку;

– при виявленні критичних змін параметрів активується відповідне оповіщення, наприклад, звуковий сигнал або зміна відображення на дисплеї.

Для забезпечення зручності роботи пристрою було реалізовано динамічне оновлення дисплея з буфером, що дозволяє змінювати тільки необхідні пікселі

без необхідності перезаписувати поточний вміст та без затримок роботи основного циклу. Крім того, програмне забезпечення передбачає можливість синхронізації часу реального годинника через Інтернет, що забезпечує точність відображення поточного часу навіть після вимкнення живлення пристрою.

Особливу увагу було приділено оптимізації енергоспоживання. В алгоритмах роботи передбачені паузи між опитуваннями сенсорів та можливість переходу пристрою у енергоефективний режим, що дозволяє збільшити автономність при живленні від акумуляторних батарей.

У процесі розробки програмного забезпечення було виконано налагодження та тестування окремих функціональних блоків з метою забезпечення коректної роботи всієї системи у комплексі. Кінцевий варіант програми відповідає всім вимогам, що були поставлені на етапі постановки завдання.

3.4 Налагодження та тестування пристрою

Після розробки апаратної та програмної частини було проведено комплексне налагодження і тестування пристрою з метою перевірки правильності його роботи та відповідності вимогам, сформульованим на етапі постановки завдання.

Налагодження розпочалося з перевірки працездатності окремих сенсорних модулів. Для кожного елемента було створено комплексну тестову програму, яка дозволила перевірити правильність підключення та функціонування. Сенсор температури, вологості та атмосферного тиску BME280 стабільно передавав дані через шину I2C, показуючи значення, що відповідали очікуваним при кімнатній температурі. Датчик освітленості TEMT6000 демонстрував коректну реакцію на зміну рівня освітлення: збільшення світла приводило до відповідного збільшення значення. Датчик перешкод US-025 впевнено визначав наближення об’єктів, що було підтверджено під час контрольних випробувань. Датчик якості повітря AGS02MA надавав стабільні дані про концентрацію частиць у повітрі без

значних коливань показників. Датчик диму MQ-2 правильно спрацьовував при наявності джерела диму поблизу, активуючи попереджувальне сповіщення.

Після окремої перевірки сенсорів було протестовано роботу всього пристрою в інтегрованому режимі. OLED-дисплей стабільно відображав поточні значення параметрів навколишнього середовища, включно з температурою, вологістю, атмосферним тиском, рівнем освітленості, якістю повітря та часом. Особливу увагу було приділено для налагодження коректного оновлення годинника реального часу, при цьому синхронізація часу через мережу Інтернет працювала без збоїв. Реалізоване програмне забезпечення забезпечувало стабільну періодичну обробку даних без затримок у виведенні інформації на дисплей.

Для перевірки працездатності в реальних умовах пристрій було розміщено у типовому житловому приміщенні і залишено на безперервну роботу протягом одного тижня. За час тестування не було зафіксовано жодних помилок у роботі пристрою. Читання даних з усіх сенсорів здійснювалося регулярно та без збоїв. Під час контрольного експерименту зі створенням димової завіси поблизу пристрою датчик MQ-2 спрацював коректно, активувавши звуковий сигнал та відповідне повідомлення на дисплеї.

Тестування також підтвердило, що пристрій здатний працювати в автономному режимі від акумуляторних батарей протягом тривалого часу. Енергоспоживання пристрою відповідало очікуваним розрахункам завдяки використанню оптимізованих режимів зниження споживання енергії під час очікування.

У цілому, результати налагодження та тестування показали відповідність функціональних можливостей розробленого пристрою усім вимогам технічного завдання. Система працює стабільно, демонструє високу надійність і готова до подальшого використання у побутових умовах. Результат тестування наведено в таблиці 3.1 та 3.2.

Таблиця 3.1 – Результат тестування пристрою на постійному джерелі живлення

Час тестування	1 год.	2 год.	4 год.	8 год.	12 год.	24 год.
Кількість збоїв	0	0	0	0	0	0
Кількість перезавантажень	0	0	0	0	0	0
Кількість фальшивих спрацювань	0	0	0	0	0	0
Кількість неточних значень	0	0	0	0	1	1

Таблиця 3.2 – Результат тестування пристрою на акумуляторах

Час тестування	1 год.	2 год.	4 год.	8 год.	12 год.	24 год.
Кількість збоїв	0	0	0	0	0	0
Кількість перезавантажень	0	0	0	0	0	0
Кількість фальшивих спрацювань	0	0	0	0	0	0
Кількість неточних значень	0	0	0	0	1	1

ВИСНОВКИ

За результатами виконаної роботи було досягнуто основної мети – розроблено пристрій для контролю параметрів повітря в приміщенні з можливістю використання у побуті як індикатора стану навколишнього середовища та системи оповіщення при виявленні диму.

Здійснено аналіз існуючих рішень у сфері вбудованих систем та IoT-технологій, що дало змогу визначити сучасні підходи до створення компактних, енергоефективних та функціональних пристроїв для моніторингу довкілля. Було розглянуто різні реалізації аналогічних пристроїв, вивчено їхні сильні й слабкі сторони.

Досліджено роботу доступних модулів, зокрема датчиків температури, вологості, освітленості, якості повітря, диму та перешкод. Визначено технічні характеристики, умови роботи, особливості підключення та сумісність із мікроконтролером Arduino. Обрано найбільш відповідні компоненти для поставленої задачі, враховуючи стабільність, енергоспоживання та точність.

Візуалізовано роботу пристрою в середовищі Fritzing, що дозволило наочно відобразити структуру підключення модулів, логіку розміщення елементів на макетній платі та у корпусі. Це забезпечує зручність під час подальшого обслуговування або доопрацювання пристрою.

Спроектовано та реалізовано повноцінну схему з мікроконтролером Arduino UNO R4 Wi-Fi, OLED-дисплеєм та набором сенсорів. Розроблено програмне забезпечення для збору, обробки та відображення даних у реальному часі, а також забезпечено звукове оповіщення при виявленні диму. Усі компоненти інтегровані в єдиний корпус, пристрій працює автономно з можливістю живлення від акумуляторів.

Отримано стабільну та інформативну систему, яка в режимі реального часу дозволяє контролювати стан повітря в приміщенні. Встановлено, що при правильному виборі модулів та відповідному програмному забезпеченні можлива надійна та безперебійна робота пристрою протягом тривалого періоду.

Розроблену систему можна легко масштабувати та модернізувати. Виведення даних на веб-інтерфейс, керування пристроєм через інтернет, додавання нових функцій або нових датчиків відкриває можливості для розширення функціоналу.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Воробйов А., Костючко С., Кулакевич О. Вбудована система управління IoT датчиками. *Програмне та апаратне забезпечення в інформаційних технологіях: матер. міжнар. наук.-практ. конф.* (м. Луцьк, 6 травня 2025 р.). Луцьк, 2025. С. 36-38.

2. ТОВ «Айті Проект». Що таке IoT (інтернет речей) простими словами?. *Atiko*. URL: https://cutt.ly/WrvXCfZH (дата звернення: 01.02.2025).

3. Herencsar N., Elbaz A. Internet of Things: A Comprehensive Overview on Protocols, Architectures, Technologies, Simulation Tools, and Future Directions. *ResearchGate*. URL: https://cutt.ly/srvXMPd7 (дата звернення: 02.02.2025).

4. HEMACHANDRA R. BHAT. Understanding Embedded Systems Architecture - Maven Silicon. *Maven Silicon*. URL: https://cutt.ly/MrvX2ZYQ (дата звернення: 03.02.2025).

5. Gopinath Krishniah. Choose the Best MCU Architecture for IoT Development - Silicon Labs. *Silicon Labs*. URL: https://cutt.ly/frvX3bG2 (дата звернення: 05.02.2025).

6. Yujing L. Security and Privacy of Internet of Things: A Review of Challenges and Solutions. *ResearchGate*. URL: https://cutt.ly/lrvX5mUn (дата звернення: 10.02.2025).

7. Air Quality Egg. Air Quality Egg - Science is Collaboration. *Air Quality Egg*. URL: https://airqualityegg.com/egg (дата звернення: 08.02.2025).

8. HowToMechatronics. DIY Air Quality Monitor - PM2.5, CO2, VOC, Ozone, Temp & Hum Arduino Meter. *How To Mechatronics*. URL: https:// https://cutt.ly/YrvX8k5v (дата звернення: 08.02.2025).

9. Jazzzman. Еволюція мікроконтролерів. *Мій Проект*. URL: https://myproject.com.ua/evoliutsiia-mikrokontroleriv.html (дата звернення: 09.02.2025).

10. Мікроконтролер Arduino. *BitKit*. URL: https://bitkit.com.ua/shho-take-arduino (дата звернення: 10.02.2025).

11. Arduino Nano 33 BLE Rev2. *Arduino в Україні*. URL: https://cutt.ly/ZrvCwzDE (дата звернення: 17.02.2025).

12. Arduino UNO R4 Wi-Fi. *Arduino в Україні*. URL: https://cutt.ly/CrvCwB0r (дата звернення: 17.02.2025).

13. Arduino Due. *Arduino в Україні*. URL: https://arduino.ua/prod314-arduino-due-a000062 (дата звернення: 17.02.2025).

14. Датчик температури DS18B20 цифровий. *Arduino в Україні*. URL: https://cutt.ly/hrvCe8Fp (дата звернення: 17.02.2025).

15. Датчик вологості та температури DHT11. *Arduino в Україні*. URL: https://cutt.ly/drvCr1Z2 (дата звернення: 19.02.2025).

16. Датчик температури та вологості SHT30 I2C. *Arduino в Україні*. URL: https://cutt.ly/ErvCtdzP (дата звернення: 19.02.2025).

17. Барометр BME280 5В I2C. *Arduino в Україні*. URL: https://cutt.ly/trvCt4Li (дата звернення: 19.02.2025).

18. Датчик вологості та температури DHT22. *Arduino в Україні*. URL: https://arduino.ua/prod301-datchik-vlajnosti-i-temperatyri-dht22 (дата звернення: 19.02.2025).

19. Датчик якості повітря AGS02MA. *Arduino в Україні*. URL: https://arduino.ua/prod6275-datchik-yakosti-povitrya-ags02ma-tvoc (дата звернення: 19.02.2025).

20. Датчик диму MQ-2. *Arduino в Україні*. URL: https://arduino.ua/prod188-datchik-dima-mq-2 (дата звернення: 19.02.2025).

21. Модуль датчика газу MQ-4. *Arduino в Україні*. URL: https://arduino.ua/prod1245-modyl-datchika-gaza-mq-4 (дата звернення: 28.02.2025).

22. Модуль датчика газу MQ-5. *Arduino в Україні*. URL: https://arduino.ua/prod1239-datchik-gaza-mq-5 (дата звернення: 28.02.2025).

23. Модуль датчика газу MQ-7. *Arduino в Україні*. URL: https://arduino.ua/prod1389-modyl-datchika-gaza-mq-7 (дата звернення: 28.02.2025).

24. Модуль датчика газу MQ-8. *Arduino в Україні*. URL: https://arduino.ua/prod1387-modyl-datchik-vodoroda-mq-8 (дата звернення: 28.02.2025).

25. Модуль датчика газу MQ-9. *Arduino в Україні*. URL: https://arduino.ua/prod1238-modyl-datchika-gaza-mq-9 (дата звернення: 28.02.2025).

26. Датчик освітленості TEMT6000. *Arduino в Україні*. URL: https://arduino.ua/prod2560-datchik-osveshhennosti-temt6000-dlya-ardyino (дата звернення: 03.03.2025).

27. 1.14 SPI дисплейний модуль 240x135 65K кольорів від Waveshare. *Arduino в Україні*. URL: https://arduino.ua/prod4307-1-14inch-lcd-module-display (дата звернення: 03.03.2025).

28. OLED дисплей 0.96 I2C 128x64 (білий). *Arduino в Україні*. URL: https://arduino.ua/prod1263-oled-displei-modyl-belii (дата звернення: 04.03.2025).

29. Датчик перешкоди (модуль YL-63). *Arduino в Україні*. URL: https://arduino.ua/prod1212-datchik-prepyatstviya-modyl-yl-63 (дата звернення: 04.03.2025).

30. Ультразвуковий датчик відстані US-025. *Arduino в Україні*. URL: https://arduino.ua/prod3710-yltrazvykovoi-datchik-rasstoyaniya-us-025 (дата звернення: 04.03.2025).

31. Liam A. Using the Arduino Software (IDE). *Arduino Documentation*. URL: https://docs.arduino.cc/learn/starting-guide/the-arduino-software-ide/ (дата звернення: 10.03.2025).

ДОДАТКИ

Додаток А

Код програми

```
#include <GyverBME280.h>
#include <GyverOLED.h>
#include <Wire.h>
#include <AGS02MA.h>
#include <WiFi.h>
#include <RTC.h>
#include <NTPClient.h>
#include <WiFiUdp.h>
WiFiUDP udp;
NTPClient timeClient(udp, "pool.ntp.org", 0, 0);
time_t epochTime = 0;

const char* ssid = "Natasha";
const char* password = "Home1337";
const int gasPin = A1;
const int ledPin = 12;
const int speakPin = 13;
const int gasAlyarm = 300;

GyverOLED<SSD1306_128x64, OLED_BUFFER> oled;
GyverBME280 bme;
AGS02MA airQualitySensor;
#define LIGHT_SENSOR_PIN A0
#define TRIG_PIN 9
#define ECHO_PIN 10
WiFiServer server(80);
unsigned long lastSyncTime = 0;
unsigned long syncInterval = 120000;
unsigned long oledOnTime = 0;
bool isOledCurrentlyOn = false;

void setup() {
  Serial.begin(115200);
  delay(10);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(ledPin, OUTPUT);
  pinMode(speakPin, OUTPUT);

  oled.init();
  oled.setScale(2);
  oled.home();
  oled.setCursorXY(10, 2);
  oled.print("Запуск");
  oled.update();
  oled.setScale(1);
  RTC.begin();
  int poyas = 3;
  oled.setPower(true);

  if (!bme.begin(0x76)) {
    oled.setCursorXY(0, 0);
    oled.print("BME FAIL");
    oled.update();
    delay(2000);
  }
  if (!airQualitySensor.begin()) {
    oled.setCursorXY(0, 0);
    oled.print("Air Sensor FAIL");
```

```
    oled.update();
    Serial.println("Air Quality Sensor not found!");
    delay(2000);
  }
  WiFi.begin(ssid, password);
  delay(1500);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Підключення до Wi-Fi...");
  }
  Serial.println("Підключено до Wi-Fi!");
    Serial.println(WiFi.localIP());
  oled.setCursorXY(25, 40);
  oled.print(WiFi.localIP());
  oled.update();
  server.begin();
  Serial.println("Веб-сервер запущено...");
  timeClient.begin();
  delay(100);
  timeClient.update();
  epochTime = timeClient.getEpochTime();
  struct tm *timeinfo = gmtime(&epochTime);
  int adjHour = (timeinfo->tm_hour + poyas ) % 24;
  delay(800);
  RTCTime newTime(
    timeinfo->tm_mday,
    (Month)(timeinfo->tm_mon),
    timeinfo->tm_year + 1900,
    adjHour,
    timeinfo->tm_min,
    timeinfo->tm_sec,
    DayOfWeek::MONDAY,
    SaveLight::SAVING_TIME_INACTIVE
  );
  delay(300);
  RTC.setTime(newTime);
  oled.setPower(false);
}

void loop() {
  RTCTime currentTime;
  WiFiClient client = server.available();
  float t = bme.readTemperature() -2.0;
  float h = bme.readHumidity();
  int hum = bme.readHumidity();
  float pask = bme.readPressure();
  float p = pask * 0.0075006;
  int poyas = 3;
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  long duration = pulseIn(ECHO_PIN, HIGH, 25000);
  float distance = duration > 0 ? duration * 0.0343 / 2.0 : -1;
  if (distance < 40) {
    oledOnTime = millis()+10000;
    isOledCurrentlyOn = true;
  }
  if (isOledCurrentlyOn) {
    oled.setPower(true);
  }
  if (oledOnTime <= millis()){
    isOledCurrentlyOn = false;
    oled.setPower(false);
```

```
}
float l = analogRead(LIGHT_SENSOR_PIN) * (5.0 / 1023.0) * 2000 - 200;
if (l < 0) l = 0;
int lux = analogRead(LIGHT_SENSOR_PIN) * (5.0 / 1023.0) * 2000 - 200;
if (lux < 0) lux = 0;

int tvoc = airQualitySensor.readPPB();
float airQualityPercent = 100.0 - (tvoc / 15000.0) * 100.0;
if (airQualityPercent < 0) airQualityPercent = 0;
airQualityPercent = roundf(airQualityPercent * 100.0) / 100.0;

RTC.getTime(currentTime);
if (millis() - lastSyncTime >= syncInterval) {
  timeClient.update();
  epochTime = timeClient.getEpochTime();
  struct tm *timeinfo = gmtime(&epochTime);
  int adjHour = (timeinfo->tm_hour + poyas) % 24;
  if (adjHour < timeinfo->tm_hour) {
    timeinfo->tm_mday += 1;
  }
  RTCTime newTime(
    timeinfo->tm_mday,
    (Month)(timeinfo->tm_mon),
    timeinfo->tm_year + 1900,
    adjHour,
    timeinfo->tm_min,
    timeinfo->tm_sec,
    DayOfWeek::MONDAY,
    SaveLight::SAVING_TIME_INACTIVE
  );
  RTC.setTime(newTime);
  lastSyncTime = millis();
  Serial.println("Час синхронізовано!");
}
oled.setScale(1);
oled.clear();
oled.setCursorXY(0, 56); oled.print(t, 1);
oled.setCursorXY(33, 56); oled.print(hum, 1);
oled.setCursorXY(55, 56); oled.print(p, 1);

oled.setCursorXY(0, 0); oled.print("lux");
oled.setCursorXY(19, 0); oled.print(lux);

oled.setCursorXY(96, 56); oled.print(airQualityPercent, 1);
oled.setCursorXY(120,56); oled.print("%");

oled.setCursorXY(100, 0);
if (currentTime.getDayOfMonth() < 10) oled.print("0");
oled.print(currentTime.getDayOfMonth());

oled.setCursorXY(112, 0);
oled.print(".");

oled.setCursorXY(117, 0);
if (Month2int(currentTime.getMonth()) < 10) oled.print("0");
oled.print(Month2int(currentTime.getMonth()));

oled.setScale(2);
oled.setCursorXY(58, 25); oled.print(":");

oled.setScale(3);
oled.setCursorXY(21, 20);
if (currentTime.getHour() < 10) oled.print("0");
oled.print(currentTime.getHour());
```

```
  oled.setCursorXY(71, 20);
  if (currentTime.getMinutes() < 10) oled.print("0");
  oled.print(currentTime.getMinutes());
  oled.update();
  int gasValue = analogRead(gasPin);

if (gasValue >= gasAlyarm) {
    digitalWrite(ledPin, HIGH);
    tone(speakPin, 500); // звук 1 кГц
  } else {
    digitalWrite(ledPin, LOW);
    noTone(speakPin);     // вимкнути звук
  }
    delay(50);

  Serial.print("T: "); Serial.print(t);
  Serial.print("  H: "); Serial.print(h);
  Serial.print("  P: "); Serial.print(p);
  Serial.print("  D: "); Serial.print(distance);
  Serial.print("  L: "); Serial.print(lux);
  Serial.print("  TVOC: "); Serial.print(tvoc);
  Serial.print(" ppb  Air %: "); Serial.println(airQualityPercent);
  Serial.println(pask);
  Serial.print("Time: ");
  Serial.print(currentTime.getHour()); Serial.print(":");
  Serial.print(currentTime.getMinutes()); Serial.print(":");
  Serial.print(currentTime.getSeconds()); Serial.print("   ");
  Serial.print(currentTime.getDayOfMonth()); Serial.print(".");
  Serial.print(Month2int(currentTime.getMonth())); Serial.print(".");
  Serial.println(currentTime.getYear());
  Serial.println(gasValue);
  delay(300);

  if (client) {
    String request = "";
    while (client.available()) {
      char c = client.read();
      request += c;
    }
    if (request.indexOf("GET /setTime?ts=") >= 0) {
      int tsIndex = request.indexOf("ts=") + 3;
      String timestamp = request.substring(tsIndex, request.indexOf(" ",
tsIndex));
      time_t unixTime = timestamp.toInt();
      struct tm *timeinfo = gmtime(&unixTime);
      int adjHour = (timeinfo->tm_hour + poyas) % 24;
       if (adjHour < timeinfo->tm_hour) {
        timeinfo->tm_mday += 1;
    }
      RTCTime newTime(
        timeinfo->tm_mday ,
        (Month)(timeinfo->tm_mon),
        timeinfo->tm_year,
        adjHour,
        timeinfo->tm_min,
        timeinfo->tm_sec,
        DayOfWeek::MONDAY,
        SaveLight::SAVING_TIME_INACTIVE
      );
      RTC.setTime(newTime);
      client.print("HTTP/1.1 200 OK\r\nContent-Type: text/plain\r\nConnection:
close\r\n\r\n");
      client.print("RTC updated");
      client.stop();
      return;
```

```
    }
    if (request.indexOf("GET /data") >= 0) {
      String data = "";
      data += "Температура: " + String(t, 1) + " C<br>";
      data += "Вологість: " + String(h, 1) + " %<br>";
      data += "Тиск: " + String(p, 1) + " мм рт. ст.<br>";
      data += "Освітленість: " + String(l, 1) + " люкс<br>";
      data += "Якість повітря: " + String(airQualityPercent, 2) + " %<br>";
      data += "Відстань: " + String(distance, 1) + " см<br>";

      client.print("HTTP/1.1 200 OK\r\n");
      client.print("Content-Type: text/html\r\n");
      client.print("Connection: close\r\n\r\n");
      client.print(data);
      client.stop();
      return;
    }
    if (request.indexOf("GET /time") >= 0) {
  String currentTimeStr = "";
  currentTimeStr += (currentTime.getHour() < 10 ? "0" : "") +
String(currentTime.getHour()) + ":";
  currentTimeStr += (currentTime.getMinutes() < 10 ? "0" : "") +
String(currentTime.getMinutes()) + ":";
  currentTimeStr += (currentTime.getSeconds() < 10 ? "0" : "") +
String(currentTime.getSeconds()) + "<br> ";
  currentTimeStr += (currentTime.getDayOfMonth() < 10 ? "0" : "") +
String(currentTime.getDayOfMonth()) + ".";
  currentTimeStr += (Month2int(currentTime.getMonth()) < 10 ? "0" : "") +
String(Month2int(currentTime.getMonth())) + ".";
  currentTimeStr += String(currentTime.getYear());
  client.print("HTTP/1.1 200 OK\r\n");
  client.print("Content-Type: text/plain\r\n");
  client.print("Connection: close\r\n\r\n");
  client.print(currentTimeStr);
  client.stop();
  return;
}
    // HTML-інтерфейс
    String response = "";
    response += "<html><head><meta charset='UTF-8'><title>Дані Сенсора</title>";
    response += "<style>body{font-family:Arial;text-
align:center;background:#f2f2f2;padding:20px}h1{color:#333}";
    response += "#sensorData{background:#fff;padding:15px;border-radius:10px;box-
shadow:0 2px 5px rgba(0,0,0,0.1)}";
    response += "p{margin:10px 0;font-size:1.1em}</style>";
    response += "<script>";
    response +=
"setInterval(()=>{  fetch('/data').then(r=>r.text()).then(data=>{    document.getE
lementById('sensorData').innerHTML=data  });  ";
    response +=
"fetch('/time').then(r=>r.text()).then(t=>{    document.getElementById('timeDispla
y').innerHTML='Час: ' + t;  });},1000);";
    response += "function syncTime(){";
    response += "let ts=Math.floor(Date.now()/1000);";
    response +=
"fetch('/setTime?ts='+ts).then(r=>r.text()).then(msg=>alert(msg));";
    response += "}";
    response += "</script></head><body>";
    response += "<h1>Моніторинг навколишнього середовища</h1>";
    response += "<div id='timeDisplay' style='font-size:1.5em;margin-
bottom:10px;color:#000'>Час: --:--:--</div>";
    response += "<div id='sensorData'>Завантаження даних...</div>";
    response += "<button onclick='syncTime()'>Синхронізувати час</button>";
    response += "</body></html>";
```

```
    client.print("HTTP/1.1 200 OK\r\n");
    client.print("Content-Type: text/html\r\n");
    client.print("Connection: close\r\n\r\n");
    client.print(response);
    delay(10);
    client.stop();
  }
}
```