**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Чорноморський національний університет імені Петра Могили**

**Факультет комп'ютерних наук**

**Кафедра комп'ютерної інженерії**

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

# Система та протоколи керування віддаленими IoT пристроями

Спеціальність 123 «Комп'ютерна інженерія»

123 – КПР.ПЗ.01 – 605.21810509

***Студент*** _____Д. В. Ієвлєв

*підпис*

«__» _____ 202__ р.

***Керівник к-т техн. наук, доцент*** _____ Я. М. Крайник

*підпис*

«__» _____ 202__ р.

**Миколаїв – 2024**

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**Чорноморський національний університет імені Петра Могили**
**Факультет комп'ютерних наук**
**Кафедра комп'ютерної інженерії**

ЗАТВЕРДЖУЮ

Зав. кафедри _____ І. М. Журавська

«_____» _____ 2024 р.

# ЗАВДАННЯ
## на виконання кваліфікаційної магістерської роботи

Видано студенту групи 605м факультету комп'ютерних наук

Ієвлєву Денису Вадимовичу
*(прізвище, ім'я, по батькові студента)*

1. Тема кваліфікаційної роботи
Система та протоколи керування віддаленими IoT пристроями

Затверджена наказом по ЧНУ ім. Петра Могили від 08.11.2023 № 226.

2. Строк представлення кваліфікаційної роботи «____» _____ 20__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні
Очікуваним результатом роботи є: апаратне та програмне забезпечення системи та протоколів керування віддаленими IoT пристроями. Вхідними даними роботи є специфікація вимог, що описує характеристики зазначеного апаратного та програмного забезпечення.

4. Перелік питань, що підлягають розробці
1) огляд сучасних підходів та систем керування віддаленими IoT пристроями;
2) аналіз переваг та недоліків існуючих систем;
3) розробка апаратної частини системи керування віддаленими IoT пристроями;
4) розробка програмної частини системи керування віддаленими IoT пристроями;

5. Перелік графічних матеріалів

_слайди презентації_____

6. Завдання до спеціальної частини _

Проведення оцінки умов праці фахівців підприємства, а саме аналіз виробничого приміщення, оцінка природного освітлення. Визначення рівня електричної та пожежної безпеки підприємства

_____

7. Консультанти:

| Консультант | Кафедра (організація) | Частина роботи |
|---|---|---|
| Д-р біол. наук, професорка Григор'єва Л. І. | Кафедра екології Навчально-наукового медичного інституту ЧНУ ім. Петра Могили | Спеціальна частина з охорони праці та безпеки у надзвичайних ситуаціях |
| | | |
| | | |

Керівник роботи

_к-т техн. наук, доцент Крайник Ярослав Михайлович_____

*(посада, прізвище, ім'я, по батькові)*

_____

*(підпис)*

Завдання прийнято до виконання

_Ієвлєв Денис Вадимович_____

*(прізвище, ім'я, по батькові студента)*

_____

*(підпис)*

Дата видачі завдання «____» _____ 20_____ р.

**КАЛЕНДАРНИЙ ПЛАН**
**виконання кваліфікаційної магістерської роботи**

Тема: _Система та протоколи керування віддаленими IoT пристроями

| № | Найменування роботи | Початок | Закінчення | Примітки |
|---|---|---|---|---|
| 1. | Розробка та затвердження завдання на виконання КМР | 01.09.2023 | 15.09.2023 | *Виконано* |
| 2. | Огляд літератури за темою роботи | 15.09.2023 | 01.10.2023 | *Виконано* |
| 3. | Складання календарного плану КМР | 01.10.2023 | 03.10.2023 | *Виконано* |
| 4. | Аналіз предметної області | 05.10.2023 | 25.10.2023 | *Виконано* |
| 5. | Розробка проєктних рішень | 25.10.2023 | 10.11.2023 | *Виконано* |
| 6. | Моделювання | 10.11.2023 | 15.11.2023 | *Виконано* |
| 7. | Конструювання АПЗ | 15.11.2023 | 20.12.2023 | *Виконано* |
| 8. | Перевірка працездатності, тестування та апробація розробленого АПЗ | 20.12.2023 | 05.01.2023 | *Виконано* |
| 9. | Аналіз результатів тестування | 05.01.2023 | 10.01.2023 | *Виконано* |
| 10. | Розробка керівництва користувача | 10.01.2023 | 13.01.2023 | *Виконано* |
| 11. | Відгук керівника КМР | 13.01.2023 | 05.02.2023 | *Виконано* |
| 12. | Оформлення КМР та презентації | 07.02.2023 | 08.02.2023 | *Виконано* |
| 13. | Попередній захист | 31.01.2023 | 11.02.2023 | *Виконано* |
| 14. | Рецензування | 12.02.2023 | 18.02.2023 | *Виконано* |
| 15. | Захист кваліфікаційної роботи | 27.02.2023 | 27.02.2023 | *Виконано* |

Розробив здобувач ВО Ієвлєв Денис Вадимович_____

(*прізвище, ім'я, по батькові)* *(підпис)*

«__» _____ 20__ р.

Керівник роботи к-т техн. наук, доцент Крайник Ярослав Михайлович__

*(підпис)*

«__» _____ 20__р

# АНОТАЦІЯ

до кваліфікаційної магістерської роботи

«Система та протоколи керування віддаленими IoT пристроями»

Студент: Ієвлєв Денис Вадимович

Керівник: к-т техн. наук, доцент Крайник Я. М.

**Актуальність теми** кваліфікаційної роботи обумовлена тим, що тема Інтернет речей (IoT) швидко впроваджується у різних галузях, від побутових пристроїв до високотехнологічних систем промисловості та медицини.

Зростання кількості підключених пристроїв призводить до необхідності вдосконалення систем управління цими мережами. Ефективні протоколи та системи керування стають ключовим елементом для забезпечення стабільності, безпеки та високої продуктивності IoT систем.

Важливі аспекти цієї теми включають розробку протоколів передачі даних, які забезпечують надійний обмін інформацією між віддаленими пристроями та центральними системами управління. Врахування обмежень ресурсів, таких як енергія та обчислювальна потужність, є важливим аспектом для оптимізації функціонування IoT пристроїв.

Окрім того, велика увага має бути приділена аспектам безпеки, так як віддалені IoT пристрої можуть стати об'єктом кіберзагроз, а несанкціонований доступ до них може призвести до серйозних наслідків. Розробка механізмів аутентифікації та авторизації є важливою для запобігання несанкціонованому втручанню.

Отже, детальне дослідження систем та протоколів керування віддаленими IoT пристроями враховує не лише технічні аспекти, а й питання ефективності, безпеки та управління ресурсами, що робить цю тему дуже актуальною у сучасному технологічному середовищі.

**Об’єкт дослідження (розробки):** технологія управління віддаленими пристроями за допомогою мікроконтролера та протоколів передачі даних.

**Предмет дослідження (розробки):** особливості розробки системи віддаленного управління IoT пристроями за допомогою мікроконтролера та протоколів передачі даних.

**Мета:** розробити систему керування віддаленними IoT пристроями з використанням мікроконтролера ESP 32.

Для досягнення поставленої мети було поставлено такі завдання:

– розглянути сучасні системи керування віддаленими IoT пристроями;

– проаналізувати недоліки існуючих систем;

– розробити апаратну частину;

– розробити програмну частину;

– створити макетну модель для демонстрації роботи.

Кваліфікаційна робота містить: перелік скорочень, вступ, чотири розділи, висновок, перелік джерел посилання та два додатки.

Вступ містить основні обґрунтування актуальності розробки обраної теми, об’єкт, предмет дослідження, мету та завдання які необхідно виконати для досягнення поставленої мети.

В першому розділі проведено огляд наявних систем керування віддаленими IoT пристроями. Розроблено специфікацію вимог до апаратно-програмного забезпечення.

У другому розділі наведено опис процесу розробки апаратного забезпечення.

У третьому розділі наведено опис процесу розробки програмного забезпечення для системи керування віддаленими IoT пристроями.

Четвертий розділ присвячено тестування розробленого програмного та апаратного забезпечення.

У висновку описано результати виконання кваліфікаційної роботи.

Додатки містять код програмного забезпечення та інформацію про апробацію кваліфікаційної роботи.

В спеціальній частині з охорони праці розглянуто забезпечення вимог охорони праці на робочому місці.

Кваліфікаційна робота містить 71 сторінок (без додатків), 23 рисунка, 20 джерел посилання, 3 додатки.

# ABSTRACT

to the qualifying master's thesis

"System and protocols for managing remote IoT devices"

Student of 605m Ievlev Denys

Consultant: Candidate of Technical Sciences, Associate Professor Kraynik Y. M.

The relevance of the qualification work topic is due to the fact that the Internet of Things (IoT) is rapidly being implemented in various industries, from household devices to high-tech systems in industry and medicine.

The growing number of connected devices leads to the need to improve the management systems of these networks. Efficient protocols and control systems are becoming a key element to ensure the stability, security, and high performance of IoT systems.

Important aspects of this topic include the development of data transfer protocols that ensure reliable information exchange between remote devices and central management systems. Taking into account resource constraints such as energy and computing power is an important aspect to optimize the functioning of IoT devices.

In addition, great attention should be paid to security aspects, as remote IoT devices can be subject to cyber threats, and unauthorized access to them can lead to serious consequences. Developing authentication and authorization mechanisms is important to prevent unauthorized interference.

Thus, a detailed study of systems and protocols for managing remote IoT devices takes into account not only technical aspects but also issues of efficiency, security, and resource management, which makes this topic very relevant in the modern technological environment.

Object of research (development): technology for controlling remote devices using a microcontroller and data transfer protocols.

Subject of research (development): features of the development of a system for remote control of IoT devices using a microcontroller and data transfer protocols.

Objective: to develop a system for controlling remote IoT devices using the ESP 32 microcontroller.

To achieve this goal, the following tasks were set:

- to consider modern systems for controlling remote IoT devices;
- analyze the shortcomings of existing systems;
- to develop the hardware part;
- develop the software part;
- create a model to demonstrate the work.

The qualification work contains: a list of abbreviations, an introduction, four chapters, a conclusion, a list of references and two appendices.

The introduction contains the main justifications for the relevance of the chosen topic, the object, subject of research, purpose and tasks to be performed to achieve the goal.

The first section provides an overview of existing systems for managing remote IoT devices. A specification of hardware and software requirements is developed.

The second section describes the hardware development process.

The third section describes the software development process for the remote IoT device management system.

The fourth section is devoted to testing the developed software and hardware.

The conclusion describes the results of the qualification work.

The appendices contain the software code and information on the qualification work testing.

The special part on labour protection considers the provision of labour protection requirements at the workplace.

The qualification work contains 65 pages (without appendices), 23 figures, 20 references, 3 appendices.

# ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

**IoT** – Internet of Things

**MQTT** – Message Queuing Telemetry Transport

**CoAP** – Constrained Application Protocol

**HTTP** – Hypertext Transfer Protocol

**DDS** – Data Distribution Service

**AMQP** – Advanced Message Queuing Protocol

**IDE** – Integrated Development Environment

**GPIO** – General-purpose input/output

**UART** – Universal asynchronous receiver/transmitter

**SPI** – Serial Peripheral Interface

**RAM** – Random Access Memory

**TCP** – Transmission Control Protocol

**UDP** – User Datagram Protocol

**CWND** – Congestion Window

**АЦП** – Аналого-цифровий перетворювач

**Гб** – Гігабайт

**ОС** – Операційна система

**ПЗ** – Програмне забезпечення

**ЦАП** – Цифро-аналоговий перетворювач

**ШІМ** – Широтно-імпульсна модуляція

# ВСТУП

В сучасному світі Інтернет речей (IoT) займає все більше та більше простору, впроваджуючи інноваційні технології у різні галузі. Однак із зростанням кількості підключених пристроїв IoT, стає все важливіше вирішення питань ефективного та безпечного управління ними з віддаленої точки. Саме ці аспекти є об'єктом вивчення та дослідження під час кваліфікаційної магістерської роботи.

У рамках даної дипломної роботи буде проведений аналіз систем та протоколів керування віддаленими IoT пристроями з урахуванням їхньої надійності, ефективності та безпеки. Метою є виявлення оптимальних рішень для забезпечення стабільності та захищеності IoT екосистем в умовах зростаючого обсягу підключених пристроїв. Магістерська робота спрямована на отримання практичних навичок у розробці, впровадженні та аналізі рішень, спрямованих на управління великим обсягом віддалених IoT пристроїв з використанням сучасних технологій та стандартів.

У контексті вивчення систем та протоколів керування IoT пристроями, дипломна передбачає аналіз відомих рішень на ринку, їхню порівняльну характеристику та визначення оптимальних стратегій для забезпечення ефективного використання підключених пристроїв в різних сценаріях застосування.

**Метою** даного дослідження є вивчення, аналіз та оптимізація систем та протоколів керування віддаленими IoT пристроями з метою підвищення їхньої надійності, безпеки та ефективності у відповідності до сучасних вимог та технологічних стандартів.

**Об'єктом дослідження** є системи управління та комунікаційні пристрої, які входять до складу Інтернету речей та забезпечують віддалене керування підключеними пристроями.

**Предметом дослідження** є протоколи та архітектури систем керування, використовані для забезпечення взаємодії між віддаленими IoT пристроями та центральними системами керування.

**Завдання:**

- ретельно проаналізувати існуючі системи та протоколи керування віддаленими IoT пристроями;
- визначити основні вимоги до систем управління IoT пристроями в контексті безпеки, надійності та швидкодії;
- розглянути сучасні технологічні стандарти та рекомендації, які стосуються систем керування віддаленими IoT пристроями;
- розробити пропозиції щодо оптимізації існуючих протоколів та архітектур з метою поліпшення їх ефективності та безпеки;
- провести експериментальне вивчення запропонованих рішень та зіставити їхню продуктивність з існуючими стандартами;
- сформулювати рекомендації щодо вдосконалення систем та протоколів керування віддаленими IoT пристроями з урахуванням результатів дослідження.

Таким чином, кваліфікаційна магістерська робота на тему «Система та протоколи керування віддаленими IoT пристроями» надасть можливість глибше розібратися у важливих аспектах розвитку та функціонування Інтернету речей, сприятиме вивченню передових технологій управління та забезпечення безпеки IoT пристроїв з віддаленої точки.

Робота пройшла **апробацію** під час XXVI Всеукраїнської науково-практичної конференції «Могилянські читання» (Миколаїв, 06–10 листопада 2023 р.). Основні положення магістерської роботи опубліковані у збірнику матеріалів XXVI Всеукраїнської науково-практичної конференції «Могилянські читання–2023» .

# РОЗДІЛ 1
# АНАЛІТИЧНИЙ ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ СИСТЕМИ ТА ПРОТОКОЛІВ КЕРУВАННЯ ВІДДАЛЕНИМИ ІОТ ПРИСТРОЯМИ.

## 1.1 Системи керування ІОТ

Системи керування в Інтернеті речей відіграють важливу роль у забезпеченні ефективності, безпеки та функціональності підключених пристроїв та ресурсів. Ці системи включають в себе різноманітні компоненти та функціональності, що спрямовані на керування та моніторинг розподіленими мережами IoT.

**Централізовані системи керування IoT**:

У централізованих системах керування Інтернет речей весь процес управління та координації відбувається через центральний вузол або сервер. Це означає, що всі рішення, аналітика та управління приймаються централізовано, а пристрої виконують вказівки, які надсилаються з центрального пункту керування.

Однорідність: Всі пристрої отримують однакові команди та інструкції від центрального сервера.

Ефективність управління: Легше впроваджувати нові стратегії, змінювати налаштування та здійснювати моніторинг.

Зручність обслуговування: Обслуговування та вдосконалення можливі централізовано, що полегшує роботу адміністраторів.

Однак централізовані системи можуть стати обмеженими у разі масштабування, вони можуть мати проблеми з відмовостійкістю, а також вони потребують потужних центральних ресурсів для обробки великих обсягів даних.

**Розподілені системи керування IoT**:

Розподілені системи керування IoT базуються на принципі розсіяння управління та прийняття рішень між різними вузлами чи пристроями в мережі. Кожен пристрій може мати свою локальну систему управління, яка реагує на зміни у середовищі та приймає рішення самостійно.

Гнучкість та адаптабельність: Пристрої можуть адаптуватися до змін у своєму оточенні без централізованих вказівок.

Масштабованість: Легко масштабується з ростом кількості підключених пристроїв.

Відмовостійкість: Відсутність єдиної точки відмови, що забезпечує стабільність системи.

Однак розподілені системи можуть бути складнішими у впровадженні та керуванні через децентралізовану природу, та можуть вимагати більше уваги до забезпечення безпеки та взаємодії між пристроями.

Обираючи між централізованими та розподіленими системами керування IoT, компанії враховують вимоги конкретного застосування, потреби щодо ефективності, масштабованості та відмовостійкості системи.

## 1.2 Централізовані системи

Централізовані системи керування IoT є підходом, при якому вся логіка управління та координації розподіляється через центральний пункт – сервер чи хмарний сервіс. Це дає можливість збирати, обробляти та керувати великою кількістю підключених IoT– пристроїв з одного центрального місця.

Основні компоненти централізованих систем керування IOT:

1) Центральний контролер: Це центральний мозок системи, який приймає та обробляє дані від датчиків, приймає рішення та видає команди підключеним пристроям.

2) Датчики та Детектори: Вимірюють різні фізичні параметри навколишнього середовища, такі як температура, вологість, освітленість, рух, дим тощо.

3) Актуатори: Виконують фізичні дії на основі команд від центрального контролера, такі як вмикання/вимикання пристроїв, регулювання рівня освітлення, відкривання/закривання дверей.

4) Мережеві пристрої: Відповідають за забезпечення підключення та комунікації між різними компонентами системи.

5) Централізовані сервери чи Хмарні платформи: Використовуються для зберігання та обробки даних, координації роботи системи та віддаленого керування.

6) Мобільні застосунки: Надають користувачам можливість віддалено керувати та моніторити систему через смартфони чи планшети.

Ці компоненти працюють взаємодіючи для створення розумної та автоматизованої системи керування, яка може оптимізувати споживання енергії, підвищувати безпеку та забезпечувати комфорт в побуті.

Централізовані системи визначаються своєю універсальністю та можливістю вирішення великої кількості завдань в різних областях. Ці системи стають ключовим елементом для ефективного управління, оптимізації ресурсів та забезпечення високої продуктивності в різних галузях сучасного суспільства.

**Центральний контролер** в системах керування Інтернету речей (IoT) відіграє невід'ємну роль у забезпеченні ефективності, надійності та безпеки мережі підключених пристроїв. Його головне завдання – збирати, обробляти дані та приймати рішення для оптимізації роботи системи.

Контролер відповідає за постійний моніторинг підключених пристроїв, збирає дані щодо їхньої діяльності та створює цілісний образ стану системи. Зібрані дані проходять через аналітичний процес, що дозволяє виявити закономірності та розробляти стратегії для прийняття оптимальних рішень.

Центральний контролер також відповідає за віддалене керування, видаючи команди підключеним пристроям та адаптуючи їх до змін у навколишньому середовищі. Постійний моніторинг дозволяє виявляти

аномалії та реагувати на потенційні загрози безпеки, забезпечуючи надійність та конфіденційність даних.

Централізоване керування також допомагає ефективно використовувати ресурси системи, уникати зайвого споживання енергії та оптимізувати процеси. Забезпечення централізованої безпеки спрощує впровадження заходів захисту від кіберзагроз, що робить систему менш вразливою до атак.

Усе враховуючи, центральний контролер в системах керування IoT виявляється необхідним компонентом, що забезпечує не тільки ефективність та безпеку, але й адаптивність до змін у сучасному технологічному середовищі.

**Датчики** та **детектори** є невід'ємними елементами в системах Інтернету речей (IoT), відіграючи ключову роль у зборі та передачі даних з навколишнього середовища. Їхні функції включають в себе збір різноманітних параметрів, таких як температура, вологість, освітленість, а також виявлення змін та подій.

Датчики постійно моніторять своє оточення, передаючи отримані дані до центрального контролера або хмарного сервісу через мережу IoT. Це забезпечує доступ до актуальних даних для подальшого аналізу та використання в системі.

Одні з основних функцій датчиків та детекторів – виявлення змін у навколишньому середовищі. Вони сприяють реагуванню системи на різноманітні події, такі як рух, зміна температури чи інші важливі аспекти. Автоматизація процесів та взаємодія з іншими компонентами системи дозволяє оптимізувати роботу та забезпечити ефективне використання ресурсів.

Датчики можуть також бути мультисенсорними, об'єднуючи кілька функцій для надання комплексного обсягу інформації про стан оточення. Це підвищує точність та повноту даних, які система отримує для подальшого аналізу та використання.

Загалом, використання датчиків та детекторів в системах IoT дозволяє

покращити аналітику, забезпечити ефективне керування ресурсами, автоматизацію процесів та реагування на зміни в реальному часі. Їхня роль в системі визначається не тільки збором даних, але і наданням інтелектуальних можливостей для адаптації системи до змін у навколишньому середовищі.

**Актуатори** в системах Інтернету речей (IoT) виконують ключову функцію взаємодії з фізичним оточенням, реагуючи на отримані дані та впливаючи на роботу підключених пристроїв. Їхнє завдання – перетворювати сигнали або команди, отримані від центрального контролера чи інших частин системи, на фізичні дії.

Актуатори призначені для виконання конкретних фізичних дій відповідно до команд, отриманих від центрального контролера. Це може включати в себе вмикання або вимикання пристроїв, регулювання параметрів оточення або зміну положення механічних частин.

Однак їхне значення не обмежується лише виконанням фізичних дій. Актуатори реагують на зміни в навколишньому середовищі та виконують необхідні дії для підтримки оптимального стану системи. Також, вони можуть бути керовані віддалено через центральний контролер, що забезпечує гнучкість управління та можливість віддаленого керування пристроями.

Узгоджена робота актуаторів з іншими компонентами системи, такими як датчики та детектори, дозволяє системі ефективно реагувати на зміни та автоматизувати виконання різноманітних завдань. Актуатори в системах IoT є ключовим елементом, що забезпечує не лише отримання даних, але й активну взаємодію з фізичним світом для досягнення поставлених цілей.

В Інтернеті речей (IoT) **мережеві пристрої** є критичним компонентом, який забезпечує зв'язок та обмін даними в екосистемі системи. Зокрема, маршрутизатори, комутатори та брендмауери є важливими елементами, що дозволяють взаємодіяти підключеним пристроям у мережі IoT.

Маршрутизатори визначають оптимальний шлях для передачі даних між різними частинами мережі, забезпечуючи ефективний обмін інформацією.

Комутатори вирішують, куди направити пакети даних в межах локальної мережі, сприяючи локальному обміну даними. Брендмауери контролюють та фільтрують трафік, забезпечуючи безпеку мережі від потенційних загроз та недозволених доступів.

Додатково, мережеві пристрої включають управління трафіком та мережевий моніторинг, що дозволяє оптимізувати роботу мережі та виявляти аномалії. Ці компоненти є важливими для стабільності та безпеки систем IoT, забезпечуючи ефективний обмін даними та оптимальне використання мережевих ресурсів. Узгоджена робота цих мережевих пристроїв визначає успішність інтеграції та функціонування систем IoT в різноманітних умовах.

У контексті систем Інтернету речей (IoT) **централізовані сервери** та **хмарні платформи** виступають важливою інфраструктурою, забезпечуючи ефективне управління, обробку та зберігання великої кількості даних, що генеруються підключеними пристроями.

**Централізовані сервери**:

Централізовані сервери слугують центром обробки та зберігання даних в системах IoT. Вони оптимізовані для складної обробки великих обсягів даних, що надходять від датчиків та інших підключених пристроїв. Використання централізованих серверів дозволяє використовувати потужні аналітичні інструменти для отримання цінної інформації та ефективно розподіляти ресурси для оптимального функціонування системи.

**Хмарні платформи**:

Хмарні платформи стають ідеальним середовищем для зберігання та обробки даних в системах IoT. Вони забезпечують гнучкість, масштабованість та доступність для систем, надаючи можливість легко змінювати розмір ресурсів, пристосовуватися до зростання кількості підключених пристроїв та забезпечувати доступ до даних з будь– якого місця через Інтернет.

Вибір між централізованими серверами та хмарними платформами залежить від специфіки системи IoT, її розміру, аналітичних потреб та вимог

до зберігання даних. У більшості випадків використання обох підходів дозволяє досягти оптимальної продуктивності та ефективності системи Інтернету речей.

**Мобільні застосунки** в сфері Інтернету речей (IoT) грають ключову роль у полегшенні взаємодії та управлінні підключеними пристроями, забезпечуючи користувачам зручність та доступність. Зокрема, ці застосунки відзначаються такими характеристиками: зручний інтерфейс, віддалене керування, моніторинг та сповіщення, інтеграція з іншими сервісами, безпека та аутентифікація. Такий підхід до використання мобільних додатків робить взаємодію та управління Інтернетом речей більш зручним та ефективним для кінцевих користувачів.

## 1.3 Розподілені системи

Розподілені системи в сфері Інтернету речей (IoT) представляють собою архітектурний підхід, що дозволяє розподілення управління та прийняття рішень між різними вузлами чи пристроями в мережі. Вони володіють рядом характеристик, що роблять їх ефективними для впровадження в складних IoT–системах.

Гнучкість та Адаптабельність: Розподілені системи в IoT дозволяють пристроям адаптуватися до змін у своєму оточенні без централізованих вказівок. Це забезпечує гнучкість у реагуванні на зміни у реальному часі.

Масштабованість: Система легко масштабується з ростом кількості підключених пристроїв, оскільки кожен пристрій працює автономно. Це забезпечує ефективне розширення мережі з врахуванням зростання обсягів даних та пристроїв.

Відмовостійкість: Відсутність єдиної точки відмови гарантує стабільність системи при виникненні проблем на окремих вузлах. Це призводить до підвищеної надійності та доступності системи.

Локальне Управління: Кожен пристрій має свою систему управління, яка може приймати рішення на основі власних даних та контексту. Це дозволяє локальну обробку та аналіз даних, зменшуючи обсяги передачі інформації по мережі.

Взаємодія та Синхронізація: Процес взаємодії між пристроями та синхронізація їхніх дій спрямовані на досягнення спільної мети. Це дозволяє координацію дій різних пристроїв для досягнення взаємовигідних результатів.

Розподілені системи в IoT стають важливим елементом для розвитку складних та великих мереж підключених пристроїв, забезпечуючи їхню ефективність та високий рівень взаємодії.

## 1.4 Протоколи керування

Протоколи керування в контексті Інтернету речей (IoT) визначають правила та стандарти взаємодії між підключеними пристроями та системами. Ці протоколи грають важливу роль у забезпеченні ефективності, безпеки та стандартизації в області керування розподіленими мережами IoT. Основні аспекти протоколів керування включають:

**MQTT** (Message Queuing Telemetry Transport) – це легкий протокол для обміну повідомленнями в мережах IoT. Спроєктований з урахуванням легкості в реалізації та мінімального обсягу передаваних даних, MQTT часто використовується в сценаріях з обмеженими ресурсами.

Протокол дозволяє асинхронну взаємодію між пристроями, де вони можуть надсилати та отримувати повідомлення у будь– який момент часу. Оптимізований для умов IoT, він враховує обмежену пропускну здатність, можливі втрати з'єднання та інші особливості мереж цього типу.

Ключовою концепцією є топіки, які визначають категорії повідомлень. Пристрої можуть підписуватися на конкретні топіки та отримувати повідомлення, які відповідають цим топікам. Протокол також підтримує рівні якості обслуговування для гарантування доставки повідомлень в мережі.

Однією з переваг є можливість встановлення повідомлення "Last Will and Testament", яке відправляється автоматично, коли пристрій втрачає з'єднання. Це забезпечує додатковий рівень надійності та контролю над з'єднанням.

MQTT є популярним протоколом в Інтернеті речей, завдяки його легкості, ефективності та здатності пристосовуватися до обмежених ресурсів пристроїв.

**CoAP** (Constrained Application Protocol) – це протокол, розроблений спеціально для обмежених пристроїв з низькими ресурсами, які активно використовуються в мережах IoT. Його особливості та характеристики:

CoAP також відомий своєю легкістю та простотою в реалізації, оптимізованою для пристроїв з обмеженими ресурсами. Протокол дозволяє ефективно обмінюватися даними в умовах низької пропускної здатності та забезпечує оптимальне використання енергії.

CoAP використовується в асинхронному режимі, де пристрої можуть взаємодіяти незалежно від часу. Така архітектура сприяє збереженню енергії та оптимальній роботі пристроїв.

Важливим елементом є концепція ресурсів, до яких пристрої може отримати доступ через визначені URI. Це дозволяє стандартизувати взаємодію між пристроями та спрощує роботу з різними типами даних.

CoAP також підтримує рівні якості обслуговування (QoS) для гарантування рівня надійності обміну даними в мережі. Також, використовуючи концепцію блокованої передачі даних, він забезпечує оптимізовану передачу великих обсягів інформації.

Іншою цікавою особливістю є використання UDP для обміну даними, що робить його ефективним для мереж, де пропускна здатність та енергія є обмеженими ресурсами. CoAP стає важливим елементом в арсеналі протоколів для впровадження IoT– рішень.

**HTTP** (Hypertext Transfer Protocol) та його безпечна версія **HTTPS** (Hypertext Transfer Protocol Secure) – протоколи, широко використовувані для обміну даними в Інтернеті та в мережах IoT.

HTTP є основним протоколом для передачі різних видів даних, таких як тексти, графіка, відео тощо. Застосовується для взаємодії між пристроями та серверами, використовуючи різні методи запитів, такі як GET, POST, PUT та DELETE.

HTTPS, як захищена версія HTTP, використовує протокол TLS/SSL для шифрування даних, забезпечуючи конфіденційність та цілісність інформації. Також використовує сертифікати для ідентифікації вебсайтів та взаємного довір'я між клієнтом і сервером.

Обидва протоколи мають просту структуру та зручні для використання в IoT. HTTP використовує заголовки для передачі інформації та управління кешуванням ресурсів, покращуючи продуктивність. HTTPS важливий для захисту важливих даних та забезпечення безпеки в умовах вимогливих до конфіденційності застосувань IoT.

**DDS** (Data Distribution Service) – це протокол для високопродуктивного обміну даними в розподілених системах, також використовується в сфері Інтернету речей (IoT).

DDS розроблений для реального часу та розподіленого обміну даними між пристроями, що можуть розташовуватися в різних частинах мережі. Однією з основних його характеристик є підтримка реального часу, що робить його ефективним для вимогливих до часу застосувань.

DDS гарантує доставку даних між пристроями та має високу масштабованість, що робить його підходящим для сценаріїв із зростанням кількості пристроїв та об'єму даних.

Протокол використовує ідентифікатори та типізацію даних для ефективного та структурованого обміну інформацією. Крім того, DDS

використовує концепцію публікації– підписки, що дозволяє пристроям висилати та отримувати дані за конкретними темами чи категоріями.

У вимогливих до реального часу системах та в IoT– застосуваннях DDS може забезпечити швидку та надійну передачу даних, роблячи його важливим елементом для сучасних розподілених систем.

**AMQP** (Advanced Message Queuing Protocol) – це протокол для передачі повідомлень у розподілених системах. Він ґрунтується на архітектурі черги, де повідомлення відправляються в чергу та можуть бути зчитані іншими компонентами системи.

AMQP є стандартизованим протоколом для ефективного обміну повідомленнями між різними частинами системи. Цей протокол може використовувати різні транспортні механізми, такі як TCP/IP, для передачі повідомлень у різних мережах.

Однією з основних переваг AMQP є його гнучкість у використанні різних моделей обміну повідомленнями, включаючи точку– точку, вентиль та розсилання. Протокол також надає механізми для гарантії доставки повідомлень та вирішення ситуацій з помилками чи втратою з'єднання.

У додатку до цього AMQP підтримує сегментацію та трансформацію даних, що дозволяє розділяти та змінювати повідомлення для оптимізації передачі даних.

Цей протокол широко використовується в різних галузях, включаючи фінанси, телекомунікації та розподілені системи Інтернету речей, де ефективний обмін повідомленнями має важливе значення.

## 1.5 Приклади використання

Система розумного будинку, також відома як «смарт– дом», представляє собою інтегровану мережу різноманітних електронних пристроїв, датчиків та систем управління. Основна мета цих систем – автоматизація та оптимізація різних аспектів побуту за допомогою технологій Інтернету речей (IoT). Вони

спрямовані на поліпшення комфорту, безпеки та раціонального використання енергії у домашньому середовищі.

Основні компоненти системи включають в себе датчики та детектори, які вимірюють різні параметри, такі як температура, вологість, рух, світло, для отримання контекстуальної інформації. Актуатори виконують фізичні дії на основі цієї інформації, наприклад, вмикання/вимикання світла, регулювання опалення, відкривання/закривання дверей.

Центральний контролер є мозком системи, обробляючи інформацію з датчиків та приймаючи рішення щодо управління різними пристроями в будинку. Мережеві пристрої відповідають за забезпечення підключення та комунікації між компонентами системи. Мобільні застосунки дозволяють користувачам віддалено контролювати та моніторити систему через смартфони чи планшети.

Основні функції систем розумного будинку включають автоматизацію завдань, таких як регулювання освітлення та опалення, забезпечення безпеки через відеоспостереження та датчики руху, а також енергоефективність шляхом моніторингу та управління споживанням енергії.

Технології та стандарти, такі як Wi– Fi, Bluetooth, Zigbee та Z– Wave, використовуються для бездротового підключення між пристроями та центральним контролером. Хмарні платформи IoT використовуються для зберігання, обробки та віддаленого керування даними.

Серед переваг систем розумного будинку можна виділити зручність в керуванні та моніторингу, ефективне використання ресурсів та підвищення рівня безпеки. Однак вартість встановлення та обслуговування, а також проблеми з приватністю можуть бути серйозними викликами для користувачів.

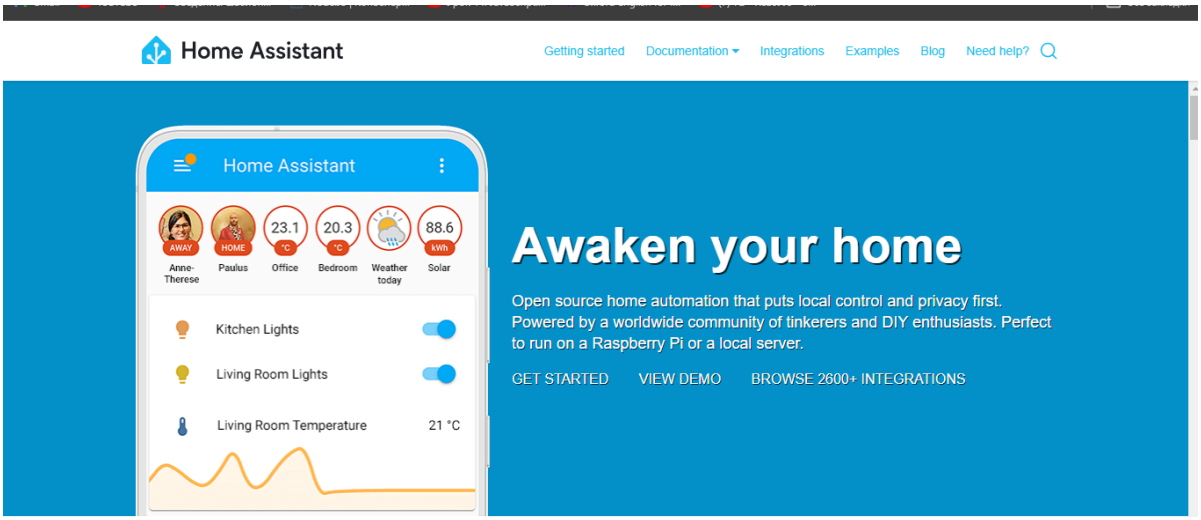


Рисунок 1.1 – Додаток для керування системою розумного будинку

Індустріальні мережі Інтернету речей (IoT) – це комплексні системи, які спрямовані на забезпечення керування та моніторингу виробничих пристроїв та машин з центральної точки. Ці мережі використовують передові технології для оптимізації виробничих процесів, підвищення ефективності та забезпечення надійності у виробничому середовищі.

Основні характеристики та елементи індустріальних мереж IoT включають сенсори та датчики, які розміщені на виробничих пристроях для вимірювання різних параметрів, таких як температура, тиск, вологість, стан обладнання тощо. Збираючи та передаючи дані, ці сенсори дозволяють отримувати інформацію в реальному часі.

Центральний керуючий центр (SCADA) виступає як централізована система, що забезпечує контроль, керування та моніторинг над різними пристроями та процесами в реальному часі. Хмарні рішення використовуються для зберігання та обробки великих обсягів даних, а також для віддаленого доступу та аналізу.

Промислові безпекові системи включають в себе системи виявлення витоків, контролю доступу та відеоспостереження, що сприяють забезпеченню безпеки на виробництві. Модульність та розширюваність системи дозволяють додавати нові пристрої та розширювати функціонал без

значних змін в загальній структурі.

Індустріальні мережі IoT виробникам надають можливість отримувати реальний час інформації про стан обладнання, виробничих процесів та ресурсів. Це сприяє швидкому реагуванню на зміни, зменшенню витрат та підвищенню загальної продуктивності у виробничому середовищі.

Системи транспортного моніторингу є комплексними рішеннями для відстеження та управління транспортними засобами. Вони застосовуються на підприємствах транспортної галузі та організаціях з власним автопарком з метою покращення ефективності, безпеки та управління транспортними ресурсами.

Ключові компоненти таких систем включають GPS– трекінг для точного визначення місцезнаходження транспортного засобу, датчики для вимірювання різних параметрів (швидкість, температура, рівень палива і т.д.), можливість віддаленого моніторингу та управління функціями транспортного засобу, аналітику та звітність для оптимізації логістики та використання ресурсів, системи безпеки та відслідковування для виявлення незаконного використання чи крадіжок, а також екологічні рішення для моніторингу викидів та зменшення негативного впливу на довкілля.

Системи транспортного моніторингу дозволяють підприємствам ефективніше управляти своїм автопарком, зменшити витрати на пальне, покращити безпеку та реагувати на екстрені ситуації. Ці технології стають важливим інструментом для оптимізації транспортних процесів та вирішення виробничих завдань.

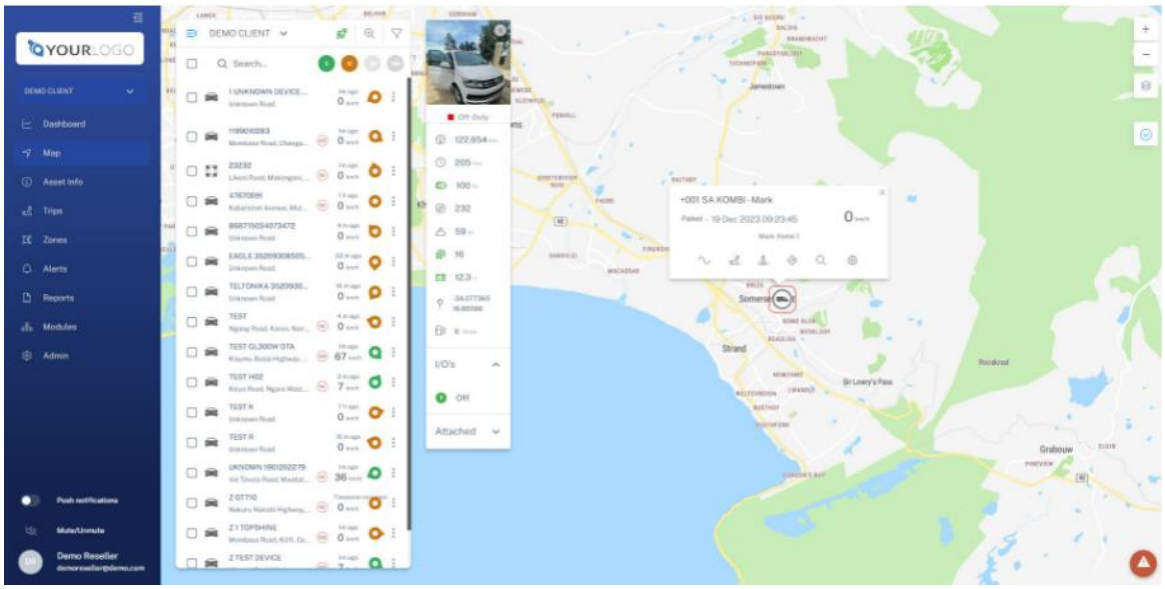


Рисунок 1.2 – Система транспортного моніторингу.

Загалом, приклади використання технологій Інтернету речей свідчать про те, що ці інновації вже впливають на різні аспекти нашого життя, забезпечуючи нові можливості та піднімаючи якість життя в цілому. Однак разом з тим, важливо враховувати питання безпеки та приватності, розробляти стандарти та етичні норми для стабільного та ефективного розвитку цих технологій у майбутньому.

## Висновки до розділу 1

На основі розгляду різноманітних прикладів використання технологій Інтернету речей можна зробити висновок, що ці технології вже стали необхідною складовою в різних сферах життя та бізнесу. Вони відкривають безліч можливостей для покращення ефективності, зручності та безпеки.

Один із яскравих прикладів – системи розумного будинку, де IoT пристрої забезпечують автоматизацію освітлення, опалення, систем безпеки та інших аспектів домашнього комфорту. Це робить життя людей більш зручним і енергоефективним.

У сфері медицини використання IoT пристроїв дозволяє віддалено моніторити стан пацієнтів, надавати оперативну медичну допомогу та вчасно реагувати на погіршення їхнього стану.

Технології IoT знаходять широке застосування у виробництві, де вони допомагають в управлінні обладнанням, моніторингу виробничих процесів та вдосконаленні логістики.

У сфері транспорту та логістики IoT дозволяє відслідковувати рух транспортних засобів, визначати оптимальні маршрути, зменшувати витрати на пальне та підвищувати безпеку.

Розробка проєкту з використанням ESP32 та Arduino IDE для Інтернету речей дозволяє створити ефективні та зручні рішення в сферах автоматизації, моніторингу та управління різноманітними системами. Використання ESP32, який об'єднує в собі можливості Wi-Fi та Bluetooth, в поєднанні з простотою Arduino IDE, робить розробку доступною для широкого кола розробників.

За допомогою цього проєкту можна створити різноманітні застосування, такі як системи моніторингу оточуючого середовища, розумні пристрої для дому, віддалений моніторинг стану обладнання та багато іншого. Використання IoT технологій на базі ESP32 відкриває безліч можливостей для покращення ефективності, зручності та безпеки в різних сферах життя та бізнесу.

# РОЗДІЛ 2
# РОЗРОБКА АПАРАТНОЇ ЧАСТИНИ

## 2.1 Детальний огляд мікроконтролера ESP 32

ESP32 – це мікросхема, яка забезпечує Wi– Fi і (в деяких моделях) Bluetooth– з'єднання для вбудованих пристроїв – іншими словами, для пристроїв IoT. Хоча технічно ESP32 – це лише мікросхема, модулі та плати для розробки, які містять цю мікросхему, часто також називаються виробником "ESP32".

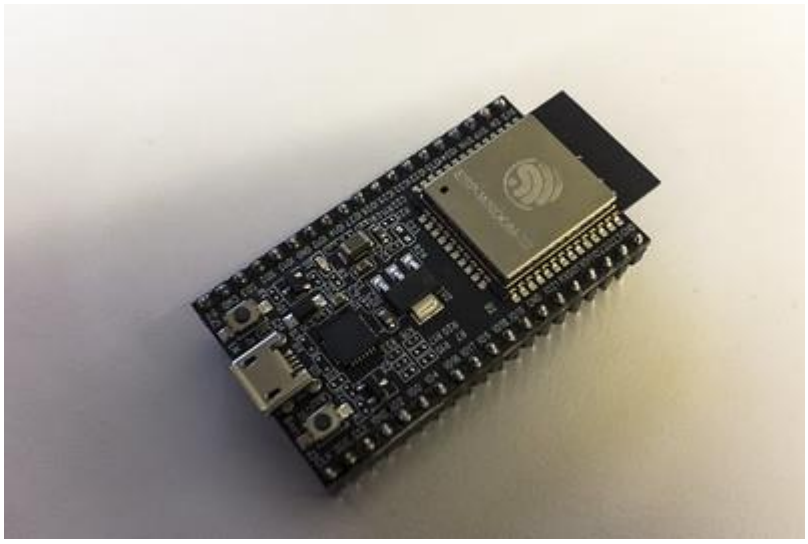

Рисунок 2.1 – мікроконтролер ESP 32.

Оригінальний чіп ESP32 мав одноядерний мікропроцесор Tensilica Xtensa LX6. Тактова частота процесора перевищувала 240 МГц, що забезпечувало відносно високу швидкість обробки даних. Зовсім недавно були додані нові моделі, включаючи серії ESP32–C і –S, які включають як одноядерні, так і двоядерні варіанти. Ці дві серії також покладаються на модель процесора Risc–V замість Xtensa. Risc–V схожа на архітектуру ARM, яка добре підтримується і добре відома, але Risc–V має відкритий вихідний код і проста у використанні. Зокрема, Risc–V і ARM мають хорошу підтримку компіляторів GNU, тоді як Xtensa потребувала додаткової підтримки і розробки для роботи з компіляторами.Arduino для ESP32 в магістерській

роботі на тему "Система керування віддаленими IoT пристроями" виявиться важливим компонентом для розробки та управління високотехнологічними пристроями. Використання мікроконтролера ESP32, обладнаного вбудованими модулями Wi– Fi та Bluetooth, дозволяє забезпечити потужність та ефективність у реалізації проєкту.

| ESP-32 | DESCRIPTION |
|---|---|
| Core | 2 |
| Architecture | 32 bits |
| Clock | Tensilica Xtensa LX106 160-240MHz |
| WiFi | IEEE802.11 b/g/n |
| Bluetooth | Yes - classic & BLE |
| RAM | 520KB |
| Flash | External QSPI - 16MB |
| GPIO | 22 |
| DAC | 2 |
| ADC | 18 |
| Interfaces | SPI-I2C-UART-I2S-CAN |

Рисунок 2.2 – особливості і специфікації ESP32.

ESP32 використовує 32– розрядний мікропроцесор Tensilica Xtensa LX6. Зазвичай це двоядерна архітектура, за винятком одного модуля, ESP32–S0WD, який використовує одноядерну систему. Тактова частота досягає 240 МГц і він виконує до 600 DMIPS (Dhrystone мільйонів інструкцій в секунду). Більше того, його низьке енергоспоживання дозволяє виконувати аналого–цифрові перетворення, а також обчислення та порогові значення рівнів, навіть коли чіп знаходиться в режимі глибокого сну.

ESP32 дозволяє підключитися до вбудованого Wi– Fi за допомогою стандартів 802.11 b/g/n/e/i/. Крім того, підключення по Bluetooth стало можливим завдяки стандарту v4.2 BR/EDR, а серія також підтримує Bluetooth з низьким енергоспоживанням (BLE).

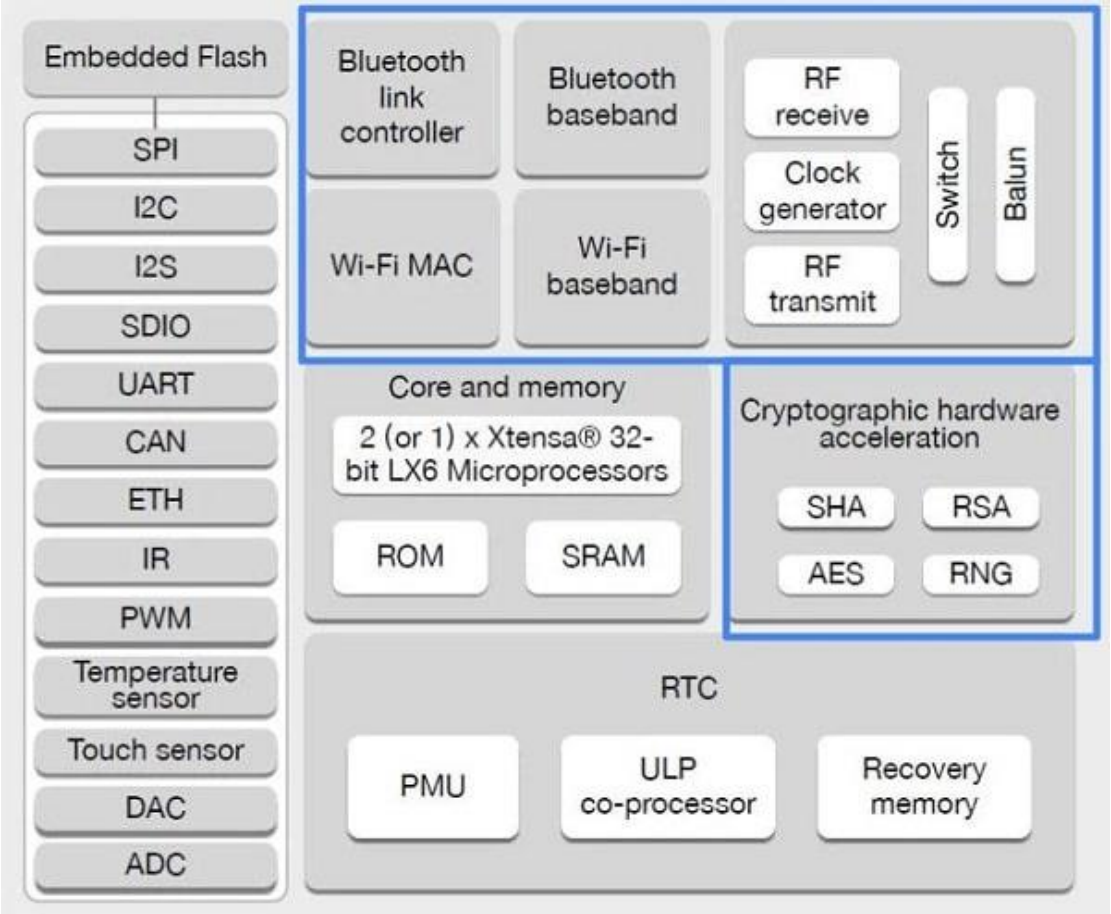


Рисунок 2.3 – Блок-схема мікроконтролера ESP32.

Внутрішня пам'ять для ESP32 наступна. ПЗУ: 448 КБ (для завантаження/функцій ядра), ОЗП: 520 КБ (для даних/інструкцій), швидка ОЗП RTC: 8 КБ (для зберігання даних/основного процесора під час завантаження зі сплячого режиму), повільна ОЗП RTC: 8 КБ (для доступу до співпроцесора під час сплячого режиму) і eFuse: 1 КБ (256 біт використовується для системи (MAC– адреса і конфігурація мікросхеми) і 768 біт зарезервовані для клієнтських додатків). Крім того, деякі мікросхеми ESP32, включаючи ESP32– D2WD і ESP32– PICO– D4, мають внутрішню флеш– пам'ять.

Зовнішня флеш– пам'ять і SRAM – ESP32 підтримує до чотирьох зовнішніх QSPI флеш– пам'яті та SRAM об'ємом 16 МБ з апаратним шифруванням на основі AES для захисту програм і даних розробників. Він отримує доступ до зовнішньої флеш– пам'яті QSPI і SRAM через високошвидкісний кеш. ESP32 підтримує всі функції безпеки стандарту IEEE

802.11, включаючи WFA, WPA/WPA2 і WAPI. Крім того, ESP32 має безпечне завантаження та шифрування флеш– пам'яті.

## 2.2 Детальний огляд світлодіодної стрічки.

Світлодіодна стрічка – це гнучкий полімерний стрічковий матеріал, на якому розташовані світлодіоди. Цей елемент освітлення став дуже популярним через свою гнучкість, енергоефективність і можливість використання в різних областях.

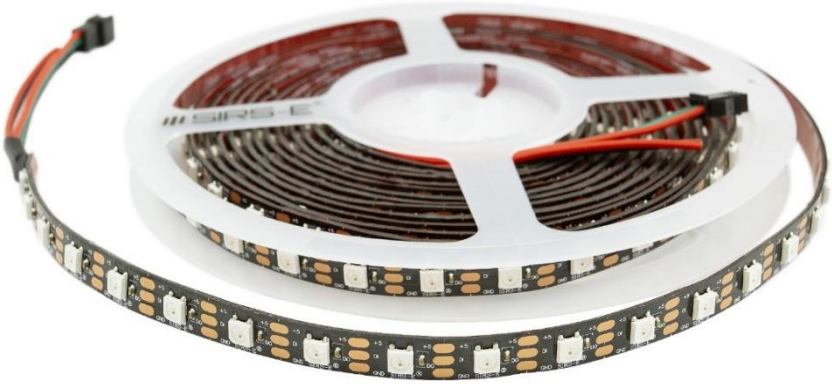

Рисунок 2.4 – Світлодіодна стрічка.

Світлодіоди є енергоефективнішими порівняно з традиційними лампами. Вони виробляють більше світла за ту ж саму кількість спожитої енергії, що дозволяє заощаджувати електроенергію. Світлодіодні стрічки можуть бути гнучкими, що дозволяє їх використання в різних архітектурних і дизайнерських проектах. Вони можуть легко вписуватися в різні форми і кути.

Багато світлодіодних стрічок можуть змінювати кольори, що робить їх ідеальними для декоративного підсвічування та створення атмосферного освітлення. Світлодіоди мають довгий термін служби, порівняно з традиційними лампами. Вони мають менше шансів вийти з ладу і можуть працювати до кількох десятків тисяч годин. Важливо враховувати, що при виборі світлодіодних стрічок слід звертати увагу на потужність, світловий потік, споживану енергію, захист від вологи та пилу (якщо вони будуть використовуватися на вулиці або в умовах високої вологості), а також на можливості керування (наприклад, за допомогою дистанційного пульта або смарт– систем).

### 2.2.1 Як працює світлодіодна стрічка

Робота світлодіодної стрічки базується на технології світлодіодів (Світлодіод, LED), які є напівпровідниковими пристроями, які випромінюють світло, коли через них пропускається електричний струм.

Основною складовою світлодіодної стрічки є світлодіоди, які випромінюють світло при подачі на них електричного струму. Світлодіоди виробляють світло в результаті рекомбінації електронів та дірок у напівпровідниковому матеріалі. Для того щоб світлодіоди на стрічці світилися, необхідно подати електричний струм. Це може бути забезпечено за допомогою джерела живлення, яке може подавати правильну напругу та струм для світлодіодів. Деякі світлодіодні стрічки можуть бути здатні змінювати кольори. Це досягається за допомогою різноманітних технологій, таких як RGB (червоний, зелений, синій) світлодіоди або плавне змінювання інтенсивності світіння окремих світлодіодів.

Світлодіодні стрічки зазвичай використовують спеціальні драйвери та контролери для управління потужністю, колірною гамою та іншими параметрами. Деякі стрічки можуть бути обладнані інтелектуальними системами керування, такими як дистанційні пульті або смарт– системи, що дозволяють користувачам зручно керувати освітленням. Світлодіодна стрічка

може бути розміщена на гнучкому полімерному стрічковому матеріалі, який дозволяє легко вписувати стрічку в різні форми та кути, що робить її універсальною для різних застосувань.

Узагальнюючи, світлодіодні стрічки працюють завдяки використанню світлодіодів як джерела світла, які випромінюють світло при подачі електричного струму, і їх можна ефективно керувати для створення різноманітного освітлення.

### 2.3 Детальний огляд NPN транзисторів.

NPN-транзистори – це тип біполярного транзистора з трьома шарами, які використовуються для посилення сигналу. Це пристрій, який керується струмом. Транзистор типу "негативний-позитивний-негативний" позначається абревіатурою NPN. У цій конфігурації напівпровідник p-типу вплавляється між двома напівпровідниковими матеріалами n-типу.

Він розділений на три секції: емітер, базу і колектор. У NPN-транзисторі потік електронів - це те, що змушує його проводити.

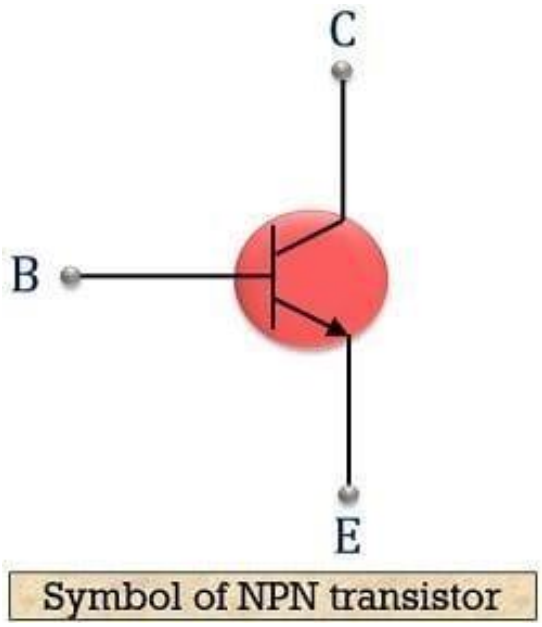


Рисунок 2.5 – Умовне позначення NPN транзистора.

Напрямок протікання струму через пристрій чітко показано стрілкою, спрямованою назовні, на емітерному виводі в символічному зображенні. Електрони складають більшість носіїв в NPN транзисторах.

NPN-транзистор будується двома способами. NPN-транзистори утворюються, коли напівпровідниковий матеріал p-типу (кремній або германій) сплавляється між двома напівпровідниковими матеріалами n-типу, як ми вже знаємо.

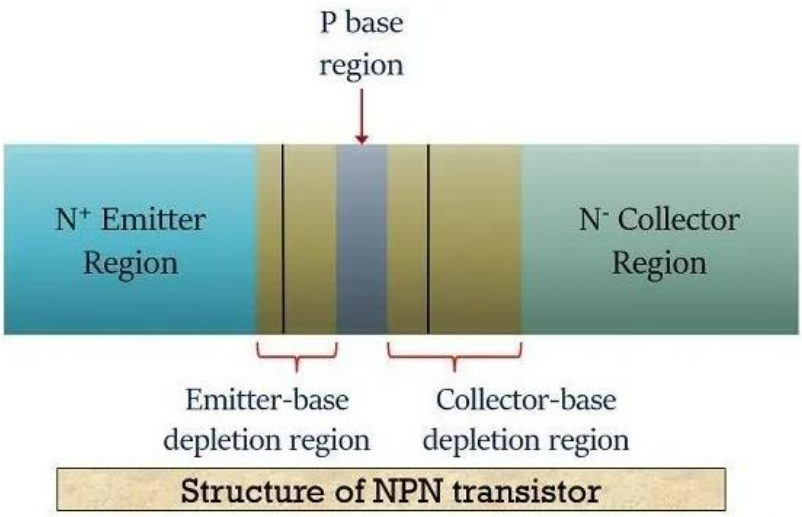


Рисунок 2.6 – Структура NPN транзистора.

NPN-транзистор складається з низки різних компонентів. Він поділяється на три секції: емітер, база і колектор.

Емітерно-базовий перехід – це область, яка з'єднує емітер і базову область. Перехід колектор-база, з іншого боку, є точкою, де з'єднуються база і колектор. Він функціонує як два PN-перехідних діода завдяки наявності двох переходів між трьома областями.

Рівні легування в кожній з трьох областей різні. Емітерна область має багато легування, тоді як базова область також має багато легування. А рівень легування колекторної області є помірним і знаходиться десь між емітерною та базовою областями. Його протилежністю є PNP-транзистор, який має P-область, затиснуту між двома областями N-типу.

Варто зазначити, що області емітера і колектора не можна поміняти місцями. Причиною цього є те, що товщина колекторної області трохи більша, ніж емітерної. Так що більше енергії може розсіюватися.

Коли до транзистора не прикладений зсув або коли між його виводами не підключена батарея. Такий стан транзистора називається незміщеним. Я вже говорив про те, як працює PN-перехідний діод за відсутності зсуву. Як ми вже знаємо, транзистор складається з двох PN-переходів.

В результаті, за відсутності зміщення, електрони в області емітера починають рухатися до області бази під впливом температурних коливань. Однак через певний час на переході емітер-база транзистора утворюється область виснаження. Лише близько 5% електронів об'єднуються з дірками в цій області після досягнення базової області, тоді як решта дрейфує через колекторну область. Аналогічно, через певний час на переході база-колектор транзистора утворюється область виснаження.

Варто зазначити, що товщина або тонкість області виснаження визначається концентрацією легування матеріалу. Інакше кажучи, у випадку слаболегованої області ширина області збіднення буде більшою, ніж у випадку сильнолегованої області. Ось чому ширина області збіднення на переході колектор-база ширша, ніж на переході емітер-база. Ці дві області збіднення служать потенційним каменем спотикання для будь-якого подальшого потоку основних носіїв.

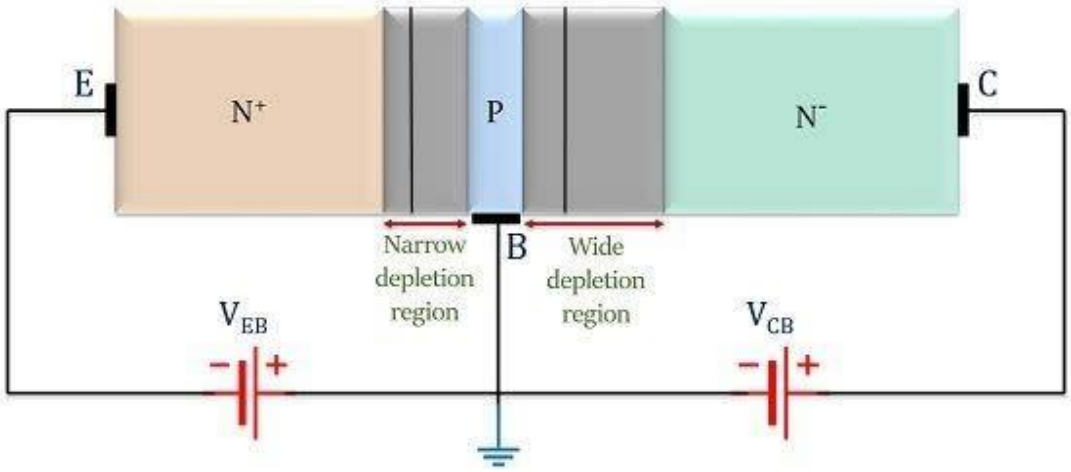


Рисунок 2.7 – Зміщений стан NPN транзистора.

Ширина області виснаження, яку також називають PN-переходом, звужується в результаті прямої напруги на емітерно-базовому переході. Аналогічно, ширина переходу колектор-база розширюється під дією зворотної напруги. Ось чому, порівняно з переходом колектор-база на попередньому малюнку, перехід емітер-база має тонку область виснаження.

Електрони починають інжектуватися в емітерну область під дією прямої напруги VBE. Електрони в цій області мають достатню енергію, щоб подолати бар'єрний потенціал емітерно-базового переходу і досягти області бази.

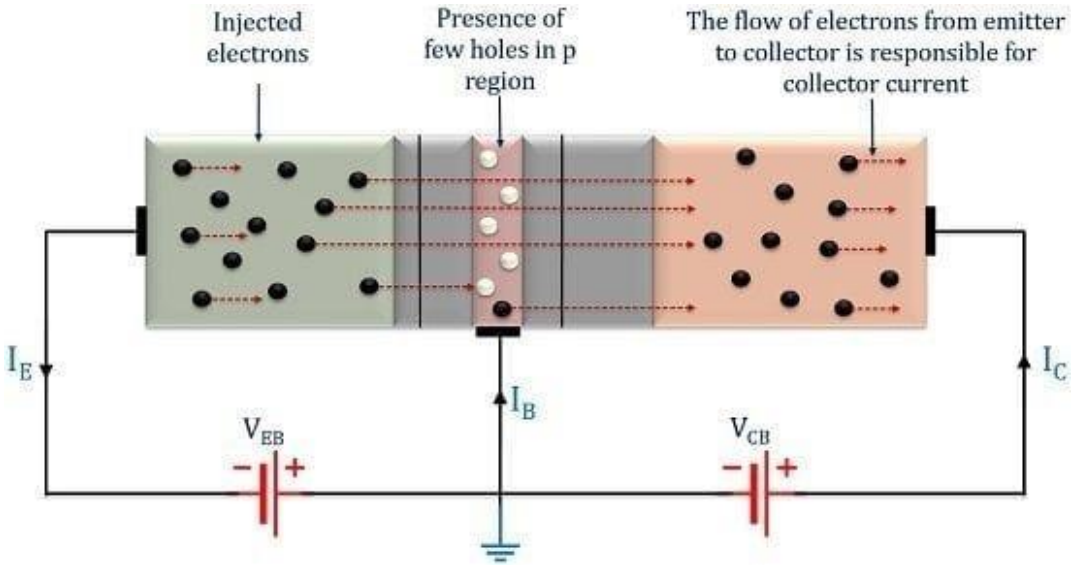


Рисунок 2.8 – Рух носіїв заряду в NPN-транзисторі.

Тому що базова область дуже тонка і легко легована. Як результат, лише кілька електронів об'єднуються з дірками, коли вони досягають місця призначення. Через сильне електростатичне поле електрони починають дрейфувати в області колектора через дуже тонку базову область і зворотну напругу на переході колектор-база. В результаті ці електрони збираються на колекторному виводі транзистора. Електрони починають рухатися до колектора, оскільки рекомбіновані дірки і електрони відокремлюються один від одного. В результаті цього руху через пристрій також протікає дуже малий струм бази. Ось чому струм емітера дорівнює сумі струмів бази і колектора.

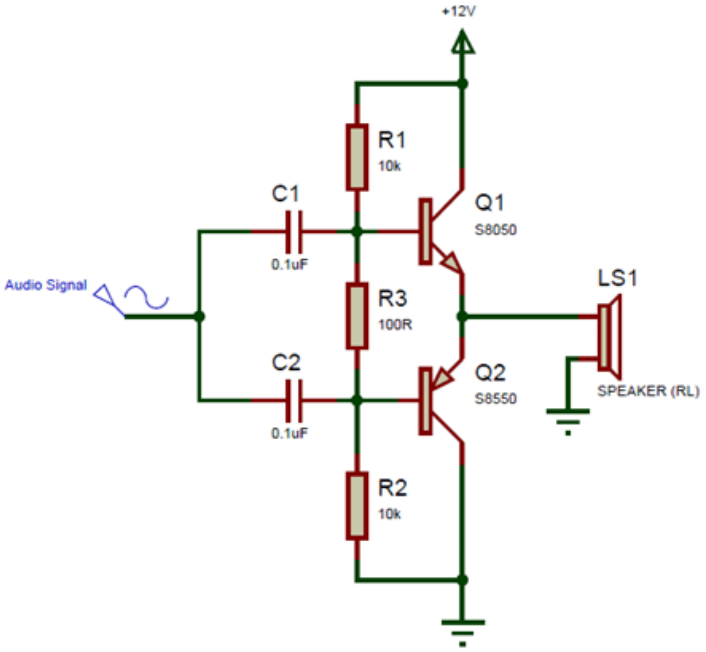


Рисунок 2.9 – Схема транзистору S8050.

Ми будемо використовувати NPN транзистор S8050. Однак, залежно від кількості світлодіодів у нашій стрічці, нам може знадобитися NPN-транзистор, здатний витримати більший постійний струм на колекторному виводі.

### 2.4 Порівняння ESP 32 та ESP 8266.

І ESP32, і ESP8266 - це дешеві SOC (системи на кристалі) на базі Wi-Fi, які ідеально підходять для DIY-проектів у сфері Інтернету речей. Обидва мають 32-розрядні процесори, ESP32 – двоядерний процесор з тактовою частотою від 80 до 240 МГц, а ESP8266 – одноядерний процесор з тактовою частотою 80 МГц. Ці модулі оснащені GPIO, які підтримують різні протоколи, такі як SPI, I2C, UART, АЦП, ЦАП і ШІМ. ESP32 і ESP8266 працюють при напрузі 3,3 В.

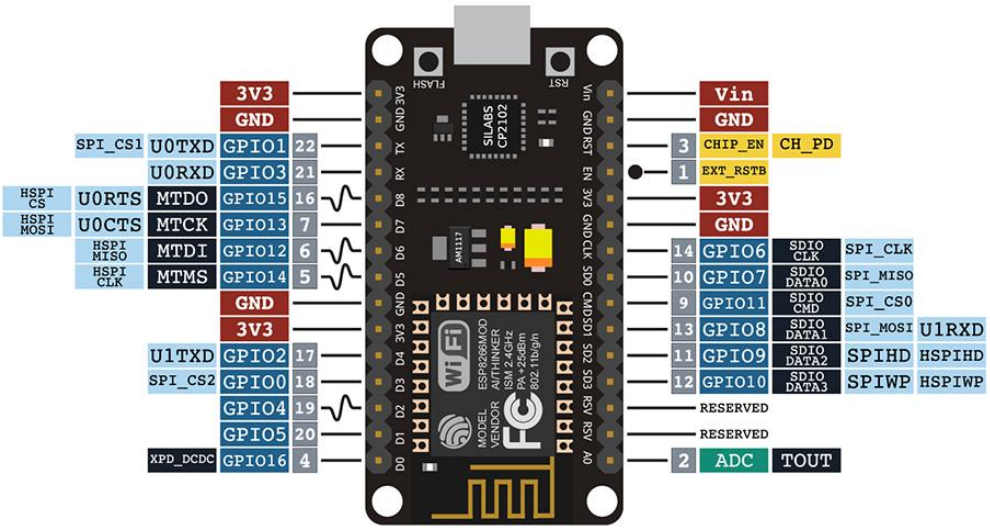


Рисунок 2.10 – Схема мікроконтролера ESP 8266.

Модуль ESP-12E на платі для розробки містить мікросхему ESP8266 з 32-бітним RISC-мікропроцесором Tensilica Xtensa LX106, який підтримує RTOS і працює з регульованою тактовою частотою від 80 до 160 МГц. ESP8266 – це недорогий, зручний пристрій для підключення ваших IoT-проектів до Інтернету. Нодемку ESP8266 включає в себе Wi-Fi трансивер 802.11b/g/n HT40, що дозволяє йому підключатися до мережі WiFi і взаємодіяти з Інтернетом, модуль може функціонувати як точка доступу (створюючи точку доступу) і як станція (підключаючись до Wi-Fi), він може просто отримувати дані і публікувати їх в Інтернеті, роблячи Інтернет речей настільки простим, наскільки це можливо. Це розширює можливості ESP8266 NodeMCU.

Цей модуль має досить потужні вбудовані можливості обробки та зберігання даних, що дозволяє інтегрувати його з датчиками та іншими специфічними пристроями через GPIO з мінімальною попередньою розробкою та мінімальним навантаженням під час роботи. 128 КБ оперативної пам'яті і 4 МБ флеш-пам'яті (для зберігання додатків і даних), чого більш ніж достатньо для обробки величезних рядків, з яких складаються веб-сторінки, даних JSON/XML і всього іншого, що ми сьогодні передаємо на пристрої IoT.

Модуль ESP8266 NodeMCU доступний в двох варіантах, один з яких побудований з мостом CP2102 USB to UART, а інший з мостом CH340 USB до UART.

Плата має стабілізатор напруги LDO для підтримки стабільної напруги на рівні 3,3 В, в той час як робочий діапазон напруги ESP8266 становить від 3 В до 3,6 В. Коли ESP8266 споживає до 80 мА під час радіочастотної передачі, він може надійно подавати до 600 мА, чого має бути більш ніж достатньо. Вихід регулятора також виведено на одну зі сторін плати і позначено як 3V3. Через цей вихід можна подавати живлення на зовнішні компоненти. Вбудований роз'єм MicroB USB забезпечує живлення для ESP8266 NodeMCU. Ви можете використовувати вихід VIN для безпосереднього живлення ESP8266 та його периферійних пристроїв, якщо подавати живлення через джерело 5В. Для зв'язку ESP8266 вимагає живлення 3,3 В і логічних рівнів 3,3 В. Виводи GPIO не мають допуску на 5 В.

Плата для розробки забезпечена модулем ESP-WROOM-32, що містить двоядерний 32-розрядний мікропроцесор Tensilica Xtensa Dual-Core LX6. Цей процесор можна порівняти з ESP8266, за винятком того, що він має два процесорних ядра (кожне з яких може працювати окремо), тактову частоту від 80 до 240 МГц і продуктивність до 600 DMIPS (Dhrystone Million Instructions Per Second).

ESP32 інтегрований Wi-Fi трансивер 802.11b/g/n HT40, який дозволяє не тільки підключатися до мережі WiFi для взаємодії з Інтернетом, але й створювати власну мережу, до якої інші пристрої можуть підключатися напряму. WiFi Direct також підтримується ESP32, що є підходящою альтернативою для однорангових з'єднань, які не потребують точки доступу. Налаштувати WiFi Direct простіше, а швидкість передачі даних значно вища, ніж у Bluetooth. Чіп також підтримує Bluetooth 4.0 (BLE/Bluetooth Smart) і Bluetooth Classic (BT), що робить його ще більш універсальним.

nodemcu ESP32 – це серія недорогих, малопотужних мікроконтролерів з вбудованим Wi-Fi ESP32 і дворежимним Bluetooth. ESP32 призначений для малопотужних додатків Інтернету речей. Висока обчислювальна потужність у поєднанні з вбудованими функціями Wi-Fi, Bluetooth і глибокого сну, 520 Кб ОЗП, 448 Кб ПЗП і 4 МБ флеш-пам'яті (для зберігання програмного забезпечення і даних) роблять його придатним для більшості портативних пристроїв IoT.

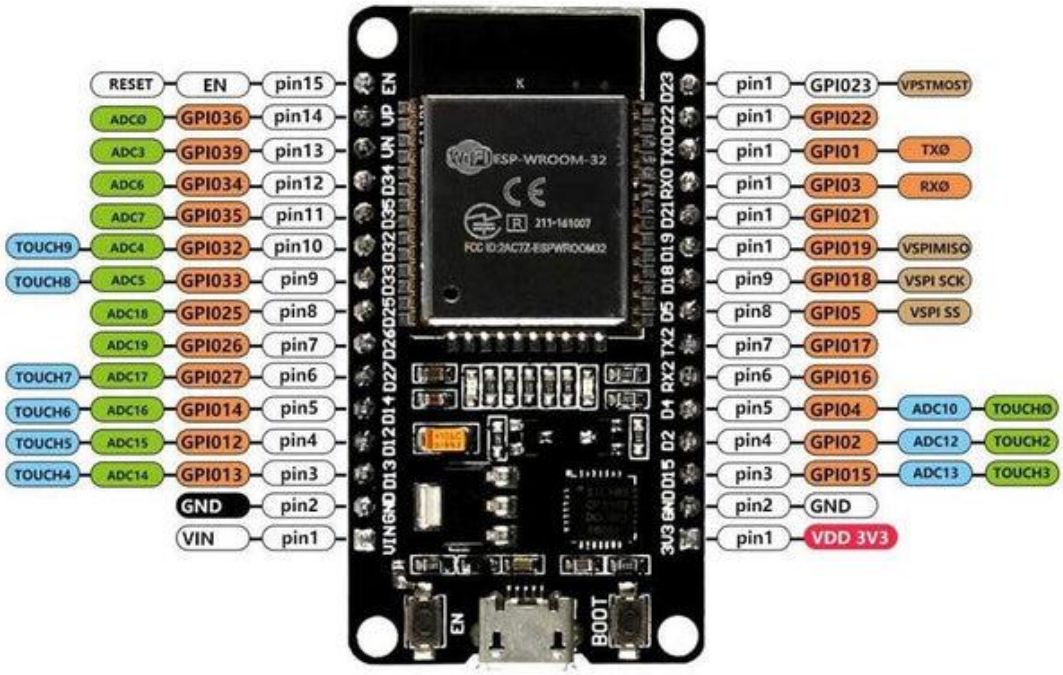


Рисунок 2.11– Схема мікроконтролера ESP 32.

Плата має стабілізатор напруги LDO для підтримки стабільної напруги на рівні 3,3 В, в той час як діапазон робочої напруги Arduino ESP32 становить від 2,2 В до 3,6 В. Коли ESP32 споживає до 250 мА під час радіопередачі, він може надійно подавати до 600 мА, чого має бути більш ніж достатньо. Вихід

регулятора також виведено на одну зі сторін плати і позначено як 3V3. Через цей вивід можна подавати живлення на зовнішні компоненти. Вбудований порт MicroB USB забезпечує живленням плату для розробки ESP32. Ви можете використовувати вивід VIN для живлення ESP32 та її периферійних пристроїв безпосередньо через зовнішнє джерело живлення 5В. Для зв'язку ESP32 вимагає живлення 3,3 В і логічних рівнів 3,3 В. Виводи GPIO не мають допуску на 5В.

ESP8266 є дуже популярною, доступною платформою для реалізації енергоефективних IoT-додатків, які працюють на основі з'єднання Wi-Fi.

У свою чергу, Espressif ESP32 є відносно новим і більш досконалим рішенням, в якому творці збільшили швидкість Wi-Fi, додали підтримку Bluetooth 4.2 і Bluetooth Low Energy, а також збільшили кількість входів/виходів.

ESP32 має більше GPIO, ніж ESP8266, і можливо вказати в коді, які виводи використовуються для UART, I2C і SPI. Це можливо завдяки можливості мультиплексування мікросхеми ESP32, яка дозволяє призначити безліч функцій на один вихід. ШІМ-сигнали можуть бути встановлені в будь-якому GPIO з налаштованими в коді частотами і робочими циклами. Аналогові виводи є статичними, але ESP32 підтримує вимірювання на 18 каналах (аналогові виводи), тоді як ESP8266 Arduino має лише один 10-розрядний вивід АЦП. ESP32 також підтримує два 8-бітних канали ЦАП. Крім того, ESP32 містить 10 ємнісних сенсорних GPIO, які виявляють дотик і можуть бути використані для запуску подій або пробудження ESP32 від глибокого сну. ESP32 підтримує протокол зв'язку Bluetooth за замовчуванням, в той час як ESP8266 не підтримує.

ESP8266 має вбудований процесор, але через багатозадачність, пов'язану з оновленням стеку Wi-Fi, більшість додатків використовують окремий мікроконтролер для взаємодії з датчиками, цифрового вводу/виводу та обробки даних. При використанні ESP32 вам може не знадобитися

додатковий мікроконтролер, оскільки ESP32 має два 32-бітних мікропроцесори і працює на рознімних платах і модулях з частотою від 160 МГц до 240 МГц. Це забезпечує достатню швидкість для будь-якої програми, яка потребує мікроконтролера з можливістю підключення.

Підсумовуючи цей короткий огляд, можна сказати, що ESP8266 – відмінний бюджетний мікроконтролер на базі Wi-Fi, але якщо нам потрібно щось більш енергоефективне і сумісне з Bluetooth, краще розглянути його наступника – модуль ESP32. Обидва мікроконтролери ESP8266 і ESP32 SoC надають радіоаматорам пристрій для зв'язку з Інтернетом, але ESP32 є дещо кращим варіантом. У будь-якому випадку, обидва пристрої є гідними представниками своїх ніш.

### 2.5 Як працюють RGB світлодіоди?

RGB-світлодіод – це поєднання 3 світлодіодів в одному корпусі:

- червоний світлодіод
- зелений світлодіод
- синій світлодіод

Комбінуючи ці три кольори, можна отримати майже будь-який колір. RGB-світлодіод показаний на наступному малюнку.

Рисунок 2.12 – RGB-світлодіод.

Звісно, за допомогою RGB-світлодіода ми можемо створювати червоне, зелене та синє світло, а налаштувавши інтенсивність кожного світлодіода, ми можемо створювати й інші кольори.

Наприклад, щоб отримати чисто синє світло, потрібно встановити синій світлодіод на найвищу інтенсивність, а зелений і червоний світлодіоди на найнижчу. Для отримання білого світла потрібно встановити всі три світлодіоди на найвищу інтенсивність.

Щоб отримати інші кольори, ми можемо комбінувати три кольори з різною інтенсивністю. Для регулювання інтенсивності кожного світлодіода можна використовувати ШІМ-сигнал. Оскільки світлодіоди розташовані дуже близько один до одного, наші очі бачать результат комбінації кольорів, а не три кольори окремо.

Щоб мати уявлення про те, як комбінувати кольори, поглянемо на наступну схему. Це найпростіша схема змішування кольорів, але вона дає уявлення про те, як це працює і як отримати різні кольори.

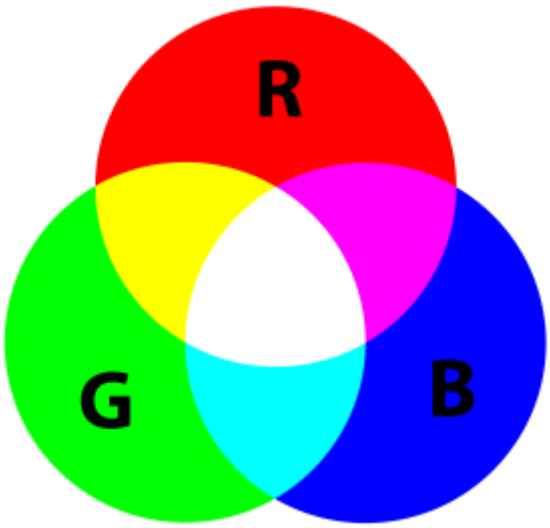


Рисунок 2.13 – RGB-діаграма .

Існує два типи RGB-світлодіодів: світлодіод зі спільним анодом і світлодіод зі спільним катодом. На малюнку нижче показані світлодіоди зі спільним анодом і спільним катодом.

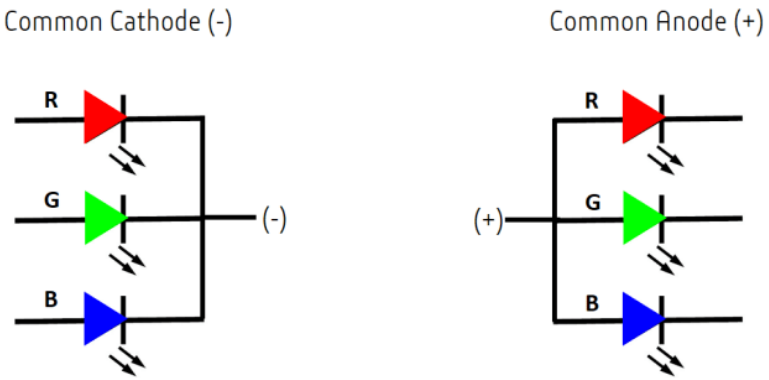


Рисунок 2.14– Схема світлодіода зі спільним анодом і катодом.

У RGB-світлодіоді зі спільним катодом усі три світлодіоди мають спільний негативний вивід (катод). У RGB-світлодіоді зі спільним анодом три світлодіоди мають спільний позитивний вивід (анод). Це призводить до того, що світлодіод має 4 виводи, по одному для кожного світлодіода, і один спільний катод або один спільний анод. RGB-світлодіоди мають чотири виводи – по одному для кожного світлодіода і ще один для загального анода або катода.

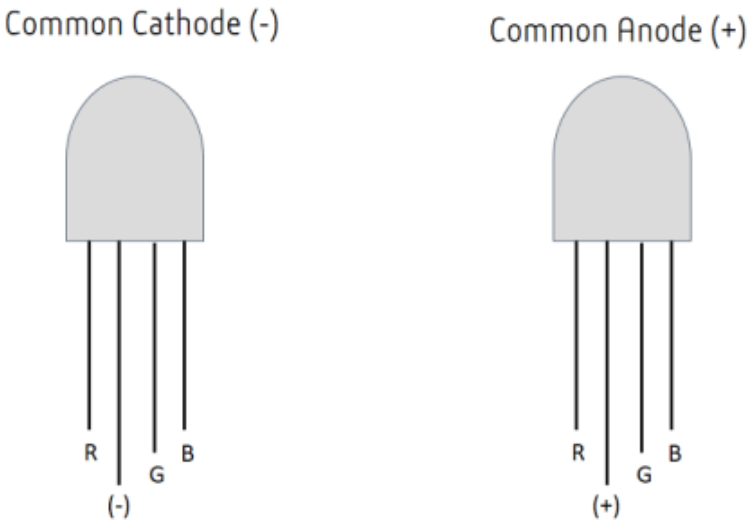


Рисунок 2.15 – Виводи анода і катода.

Якщо світлодіод повернутий до нас так, щоб анод або катод (найдовший вивід) був другим зліва, виводи повинні бути в такому порядку: червоний, анод або катод, зелений і синій.

## Висновки до розділу 2

В другому розділі кваліфікаційної магістерської роботи було обгрунтовано та висвітлено вибір модуля ESP32 для реалізації системи та протоколів керування віддаленими IoT пристроями. Аналізуючи апаратні аспекти проєкту та обираючи технології, я прийшов до наступних висновків.

ESP32 представляє собою універсальний мікроконтролер, який поєднує в собі високу продуктивність та широкі можливості з підтримкою бездротового зв'язку, такого як Wi-Fi та Bluetooth. Вибір ESP32 дозволяє забезпечити необхідні функціональні можливості для керування IoT пристроями та забезпечити їх ефективну взаємодію. ESP32 користується популярністю та підтримкою у спільноті, що визначається наявністю різноманітних бібліотек, таких як для керування світлодіодними стрічками, використання сенсорів, роботи з бездротовим зв'язком, тощо. Це полегшує процес розробки та інтеграції функцій у проект.

Вбудовані можливості Wi-Fi та Bluetooth дозволяють ефективно і надійно взаємодіяти з іншими пристроями в мережі та забезпечують можливість інтеграції у системи IoT. Це розширює можливості віддаленого керування та моніторингу. Також в контексті мого дослідження та розробки проєкту, було вибрано Arduino IDE як середовище для програмування мікроконтролера ESP32. Arduino IDE є відкритим програмним забезпеченням, спеціально розробленим для спрощення процесу програмування мікроконтролерів, таких як ESP32.

NPN-транзистор S8050 буде використовуватися як ключ у схемі керування світлодіодною стрічкою. Цей транзистор дозволяє ефективно керувати потужністю, яка подається на світлодіодну стрічку від ESP32.

Протокол HTTP/HTTPS забезпечує стандартний механізм комунікації через Інтернет. Він є універсальним та широко використовується для керування віддаленими пристроями через веб-інтерфейси.

# РОЗДІЛ 3
# ПРОГРАМНА ЧАСТИНА КОМПЛЕКСУ

Проект дозволяє керувати кольором та яскравістю RGB LED стрічки за допомогою ESP32 мікроконтролера. Це може бути використано як демонстрація здатностей ESP32 у керуванні різними пристроями. Вбудований веб-сервер на ESP32 надає можливість вибирати бажаний колір за допомогою спеціального інтерфейсу на веб-сторінці. Це зручний спосіб керування освітленням, особливо якщо стрічка встановлена в важкодоступному місці або якщо ви хочете дистанційно змінювати колір. Проект може бути корисним в різних областях, таких як декоративне освітлення, автоматизація приміщень, створення атмосферних ефектів та багато іншого. Його можливо використовувати як частину дослідження з вбудованими системами, Інтернетом речей або зв'язком між мікроконтролерами та веб-інтерфейсами.

Загалом, цей проект може слугувати як добрий приклад практичного використання мікроконтролерів, веб-серверів та керування освітленням.

## 3.1 Детальний огляд роботи протоколів керування.

Протокол управління передачею (Transmission Control Protocol, TCP) – це стандарт зв'язку, який використовують програми для обміну даними. Він встановлює параметри обміну, підтверджує, що надсилається, звідки надходить, куди надходить і чи правильно доставлено.

На відміну від User Datagram Protocol (UDP) – іншого стандарту, який програми використовують для обміну даними, – TCP розроблений для точності, а не швидкості. Під час передачі даних пакети даних іноді можуть виходити з ладу або губитися. TCP нумерує кожен пакет, щоб гарантувати, що кожен фрагмент досягне місця призначення і може бути переставлений, якщо потрібно. Коли пакети не надходять протягом визначеного часу, протокол керування передачею запитує повторну передачу втрачених даних.

Протягом усього обміну TCP підтримує зв'язок між двома додатками, гарантуючи, що обидві сторони надсилають і отримують все, що потрібно передати, і підтверджуючи, що це правильно.

Протокол управління передачею – це найпопулярніший стандарт для обміну даними через Інтернет– протокол (IP), і його часто називають TCP/IP. Оскільки він допомагає полегшити обмін даними через Інтернет, TCP є частиною так званого "транспортного рівня" мережі. Це щось на кшталт кур'єра в інтернеті. Після того, як TCP встановлює з'єднання і визначає взаємодію, він упаковує дані, завантажує їх на окремі транспортні засоби і відправляє до місця призначення через IP– магістраль.

"В дорозі" затори (перевантаження мережі), об'їзди (балансування навантаження) та автомобільні аварії (помилки мережі) можуть призвести до того, що дані вийдуть з ладу або не надійдуть вчасно.

Коли пакети даних надходять, одержувач, по суті, розписується за них за допомогою підтвердження: "Я отримав пакет 4". Якщо відправник не отримує цього підтвердження протягом певного часу, він повторно надсилає передачу. Після того, як все отримано, TCP розриває з'єднання. Весь процес відбувається в три окремі фази.

Операції TCP включають в себе численні кроки, на яких дві кінцеві точки використовують TCP, щоб робити запити, підтверджувати один одного і підтверджувати, що обмін відбувається так, як було заплановано. Ці кроки можна поділити на три основні етапи:

- Встановлення з'єднання
- Передача даних
- Завершення з'єднання

Назви етапів говорять самі за себе, але на кожному етапі процесу відбувається набагато більше.

На етапі встановлення з'єднання TCP забезпечує "тристороннє рукостискання", під час якого програми запитують синхронізацію та

підтвердження один одного. На цьому етапі TCP встановлює параметри для обміну і підтверджує, що обидва об'єкти (наприклад, сервер і клієнт) можуть брати участь в обміні.

Коли протокол керування передачею отримує потік даних, розділяє його і додає до передачі TCP– заголовок, потік даних стає "TCP– сегментом". TCP– заголовок гарантує, що коли окремі пакети даних прибудуть до місця призначення, їх можна буде легко розташувати в правильному порядку, і одержувач зможе чітко бачити, чи немає чогось пропущеного.

На етапі встановлення з'єднання програми можуть оголосити свій максимальний розмір сегмента (MSS), який визначає найбільший TCP– сегмент, яким вони будуть обмінюватися. Це найбільший обсяг даних, який буде передано в одному сегменті. Якщо MSS занадто великий, фрагментація IP розбиває окремі пакети на менші частини, збільшуючи ризик того, що деякі пакети будуть загублені, і програмам доведеться повторно передавати дані кілька разів (це збільшує затримку).

Під час передачі даних TCP гарантує, що пакети надсилаються зі швидкістю, з якою можуть впоратися ресурси мережі. Спочатку TCP дозволяє передавати через мережу лише кілька байт. Це так зване "вікно перевантаження" (Congestion Window, CWND). Коли одержувач підтверджує, що дані надійшли успішно, TCP експоненціально збільшує вікно перевантаження і пропускає більше даних.

Після того, як CWND досягає певного порогу, збільшення стає лінійним. Якщо пакет втрачається, TCP значно зменшує вікно перевантаження і знову починає повільну передачу. З часом пропускна здатність TCP формує пилкоподібну форму, де швидкість передачі різко збільшується і зменшується, щоб контролювати перевантаження мережі.

На відміну від UDP, TCP перевіряє передачу на наявність помилок. Використовуючи порядкові номери та контрольну суму, він визначає, чи

передача надійшла правильно. Якщо один біт неточний, контрольна сума буде неправильною. Коли це трапляється, TCP відкидає неправильний сегмент.

### 3.2 Опис технології та мови програмування

Розробляти електроніку стало простіше завдяки програмному забезпеченню Arduino (IDE) та платам Arduino (апаратному забезпеченню). Цей набір допомагає створювати цифрові та інтерактивні пристрої за допомогою інших компонентів. У попередній статті ми говорили про плати Arduino. У цій статті ми розберемося, що таке програмне забезпечення (IDE) ардуіно і як ним користуватися.

Програмне забезпечення Arduino (IDE) – це програмне забезпечення з відкритим вихідним кодом, яке використовується для програмування плат Arduino, і являє собою інтегроване середовище розробки, розроблене компанією arduino.cc. Дозволяє писати та завантажувати код на плати Arduino. Складається з безлічі бібліотек і набору прикладів міні– проектів.

Програмне забезпечення (IDE) arduino сумісне з різними операційними системами (Windows, Linux, Mac OS X) і підтримує мови програмування (C/C++).

Програмне забезпечення Arduino просте у використанні як для початківців, так і для досвідчених користувачів. З його допомогою можна почати програмувати електроніку та робототехніку, а також створювати інтерактивні прототипи.

Отже, програмне забезпечення Arduino – це інструмент для розробки нових речей і створення нових електронних проектів будь– ким.

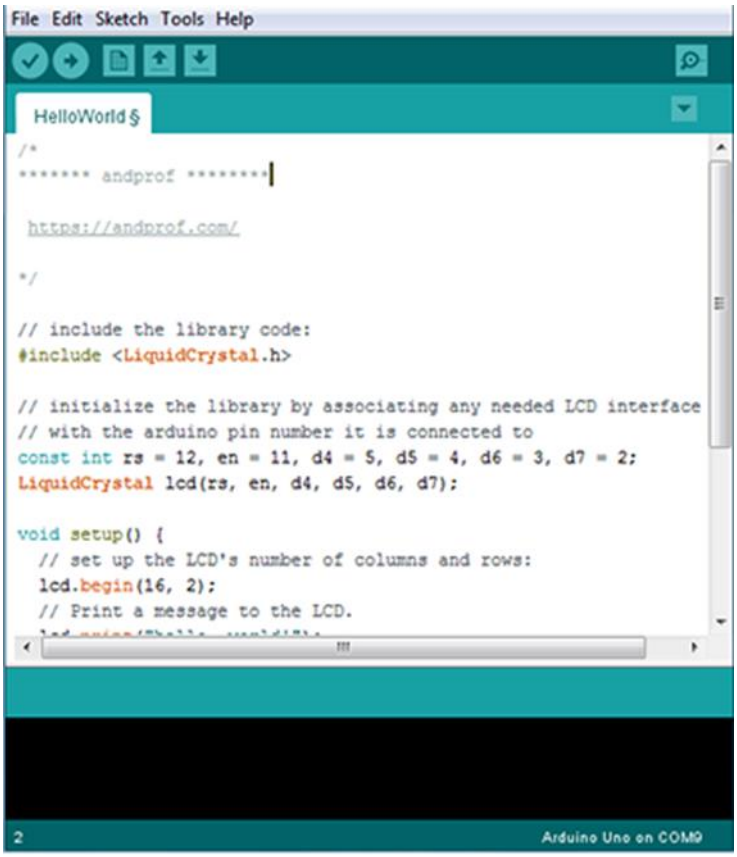


Рисунок 3.1 – Програмний інтерфейс Arduino.

Arduino IDE для ESP32 забезпечує зручність програмування та швидкість розробки завдяки своєму інтуїтивно зрозумілому інтерфейсу та великій спільноті розробників. Можливість використання розширень та бібліотек Arduino дозволяє спростити інтеграцію додаткових функцій у ваш проект.

ESP32 підтримує багатоядерність, що робить його ідеальним для обробки складних завдань та одночасного виконання різних функцій. Вбудовані бездротові модулі ESP32 роблять його надзвичайно зручним для забезпечення сполучення між пристроями та передачі даних.

Застосування ESP32 у магістерській роботі дозволить нам реалізувати різноманітні функції, такі як віддалене керування, збір та обробка даних, взаємодія зі сенсорами та актуаторами. Додатково, вбудовані функції енергозбереження ESP32 стануть важливим аспектом для пристроїв IoT, які працюють на батареях.

Вибір Arduino для ESP32 дозволяє створити потужну та ефективну систему керування віддаленими IoT пристроями, відповідаючи вимогам магістерської роботи.

3.1.1 Опис коду.

1. Завантажуємо бібліотеку Wi-Fi

```
#include <WiFi.h>
```

2. Замінюємо на наші мережеві облікові дані

```
const char* ssid     = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";
```

3. Встановлюємо номер порту веб-сервера на 80

```
WiFiServer server(80);
```

4. У наступних рядках визначено рядкові змінні для зберігання параметрів R, G та B із запиту.

```
String redString = "0";
String greenString = "0";
String blueString = "0";
```

5. Наступні чотири змінні використовуються для подальшого декодування HTTP-запиту.

```
int pos1 = 0;
int pos2 = 0;
int pos3 = 0;
int pos4 = 0;
```

6. Змінна для зберігання HTTP-запиту

```
String header;
```

7. Створюємо три змінні для GPIO, які керуватимуть параметрами смуг R, G і B. У цьому випадку ми використовуємо GPIO 13, GPIO 12 і GPIO 14.

```
const int redPin = 13;
const int greenPin = 12;
const int bluePin = 14;
```

8. Ці GPIO мають виводити ШІМ-сигнали, тому спочатку потрібно налаштувати властивості ШІМ. Встановимо частоту ШІМ-сигналу 5000 Гц. Потім призначимо канал ШІМ для кожного кольору.

```
const int freq = 5000;
const int redChannel = 0;
const int greenChannel = 1;
const int blueChannel = 2;
```

9. І, нарешті, встановимо роздільну здатність каналів ШІМ на 8 біт.

```
const int resolution = 8;
```

10. Поточний час

```
unsigned long currentTime = millis();
```

11. Попередній час

```
unsigned long previousTime = 0;
```

12. Визначаємо час таймеру в мілісекундах

```
const long timeoutTime = 2000;
```

13. Налаштування функціоналу ШІМ для світлодіодів

```
ledcSetup(redChannel, freq, resolution);
ledcSetup(greenChannel, freq, resolution);
ledcSetup(blueChannel, freq, resolution);
```

14. Приєднаємо канали ШІМ до відповідних GPIO.

```
ledcAttachPin(redPin, redChannel);
ledcAttachPin(greenPin, greenChannel);
ledcAttachPin(bluePin, blueChannel);
```

15. Підключаємось до мережі Wi-Fi за допомогою SSID та пароля

```
Serial.print("Connecting to ");
Serial.println(ssid);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}
```

16. Створюємо локальну IP-адресу та запускаємо веб-сервер

```
   Serial.println("");
   Serial.println("WiFi connected.");
   Serial.println("IP address: ");
   Serial.println(WiFi.localIP());
   server.begin();
 }
```

17. Зчитуємо нових клієнтів

```
void loop(){
  WiFiClient client = server.available();
```

18. Виводимо повідомлення у послідовний порт, також для зберігання вхідних даних від клієнта ставимо рядковий тип даних

```
if (client) {
  currentTime = millis();
  previousTime = currentTime;
  Serial.println("New Client.");
  String currentLine = "";
```

19. Виконується цикл, поки клієнт підключений

```
while (client.connected() && currentTime - previousTime <= timeoutTime) {
```

20. Якщо є байти для читання з клієнта, прочитати його, а потім вивести його на послідовний монітор

```
currentTime = millis();
if (client.available()) {
  char c = client.read();
  Serial.write(c);
  header += c;
```

21. HTTP-заголовки завжди починаються з коду відповіді (наприклад, HTTP/1.1 200 OK) і типом вмісту, щоб клієнт знав, що прийде, а потім порожній рядок.

```
client.println("HTTP/1.1 200 OK");
client.println("Content-type:text/html");
client.println("Connection: close");
client.println();
```

22. Відображаємо веб-сторінку HTML.

```
client.println("<!DOCTYPE html><html>");
client.println("<head><meta name=\"viewport\" content=\"width=device-width,
client.println("<link rel=\"icon\" href=\"data:,\">");
client.println("<link rel=\"stylesheet\" href=\"https://stackpath.bootstrap
client.println("<script src=\"https://cdnjs.cloudflare.com/ajax/libs/jscolo
client.println("</head><body><div class=\"container\"><div class=\"row\"><h
client.println("<a class=\"btn btn-primary btn-lg\" href=\"#\" id=\"change_
client.println("<input class=\"jscolor {onFineChange:'update(this)'}\" id=\"
client.println("<script>function update(picker) {document.getElementById('r
client.println("document.getElementById(\"change_color\").href=\"?r\" + Mat
```

23. HTTP-відповідь закінчується ще одним порожнім рядком.

```
client.println();
```

24. Коли ми отримуємо колір, ми отримаємо запит наступного формату.

```
/?r201g32b255&
```

25. Отже, нам потрібно розділити цей рядок, щоб отримати параметри R, G і B. Параметри зберігаються у змінних redString, greenString та blueString і можуть мати значення від 0 до 255.

```
if(header.indexOf("GET /?r") >= 0) {
  pos1 = header.indexOf('r');
  pos2 = header.indexOf('g');
  pos3 = header.indexOf('b');
  pos4 = header.indexOf('&');
  redString = header.substring(pos1+1, pos2);
  greenString = header.substring(pos2+1, pos3);
  blueString = header.substring(pos3+1, pos4);
```

26. Для керування стрічкою за допомогою ESP32 використаємо функцію ledcWrite() для генерації ШІМ-сигналів зі значеннями, декодованими з HTTP-запиту.

```
ledcWrite(redChannel, redString.toInt());
ledcWrite(greenChannel, greenString.toInt());
ledcWrite(blueChannel, blueString.toInt());
```

27. Якщо отримали щось інше, окрім символу повернення каретки, додати його в кінець поточного рядка currentLine.

```
  } else if (c != '\r') {
    currentLine += c;
  }
```

28. Очистити змінну заголовка.

```
header = "";
```

29. Закриваємо з'єднання.

```
client.stop();
Serial.println("Client disconnected.");
Serial.println("");
```

## 3.3 Опис інтерфейсів АПЗ

### 3.3.1 Схема підключення ESP 32.

Щоб визначити максимальний струм, який споживає наша світлодіодна стрічка, ми можемо виміряти струм споживання, коли всі світлодіоди працюють на максимальній яскравості (коли колір білий).

Оскільки ми використовуємо 12 світлодіодів, максимальний струм становить приблизно 630 мА при повній яскравості білого кольору. Отже, ми можемо використовувати NPN транзистор S8050, який може витримувати до 700 мА.

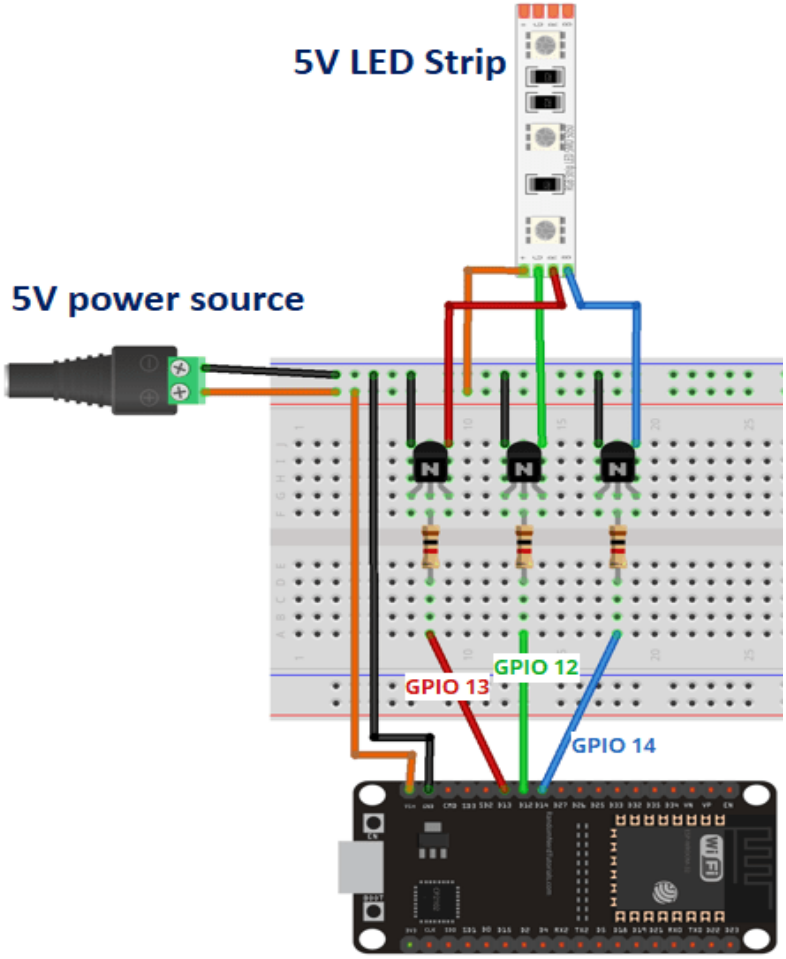


Рисунок 3.2 – Схема підключення ESP 32.

**Висновки до розділу 3**

У висновку даного розділу важливо відзначити, що вибір Arduino для програмування мікроконтролерів взяв свій початок з обґрунтованих технічних

та практичних переваг. Arduino, завдяки своїй простоті, широкій спільноті користувачів і великому асортименту готових бібліотек, дозволив ефективно реалізувати програмну частину проєкту.

У цьому розділі ми розглянули ключові аспекти проекту системи керування віддаленими IoT пристроями, використовуючи Arduino IDE. Почали ми з вибору платформи розробки, визначивши, що Arduino IDE є оптимальним інструментом для створення програмного забезпечення для наших пристроїв. Детально розглянули процес підключення до ESP32, використовуючи Arduino IDE, і представили схему підключення, яка включає основні компоненти та їх взаємодію.

Особливий акцент був зроблений на протоколах керування, які визначають взаємодію між системою та віддаленими IoT пристроями. Було вказано, що використовуємо певні протоколи для забезпечення ефективної передачі даних та керування пристроями. Це дозволяє забезпечити надійну і стабільну роботу системи при взаємодії з багатьма пристроями одночасно.

Загалом, використання Arduino IDE, схема підключення до ESP32 та використання конкретних протоколів керування становлять фундамент для успішної реалізації нашого проекту. Цей розділ надає важливу основу для розуміння аспектів розробки та взаємодії у контексті віддалених IoT пристроїв.

# РОЗДІЛ 4
# АНАЛІЗ РЕЗУЛЬТАТІВ ПРОЄКТУ

## 4.1 Тестування приладу

Розроблений проект включає в себе можливість створення креативного освітлення для різних просторів. За допомогою світлових ефектів ми можемо створювати унікальні атмосфери, використовувати його в ігрових проектах для покращення ігрового досвіду та забезпечувати інтерактивні можливості через дистанційне керування.

Цей проект може допомогти нам створити привабливий та функціональний елемент освітлення для різних потреб, а його готовість до використання полегшить інтеграцію у будь-яку сферу.

### 4.1.1 Як мікроконтролер ESP32 взаємодіє з веб-сервером

Веб– сервер ESP32 відображає палітру кольорів. Коли ми вибрали колір, наш браузер робить запит на URL– адресу, яка містить параметри R, G і B вибраного кольору.

ESP32 отримує запит і розділяє значення для кожного параметра кольору. Потім він надсилає ШІМ– сигнал з відповідним значенням на GPIO, які керують стрічкою.

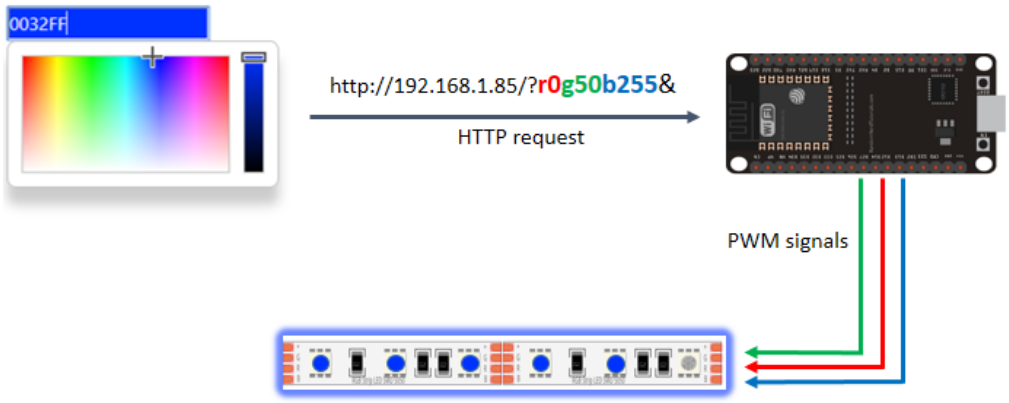

Рисунок 4.1 – Принцип роботи модуля ESP32 зі світлодіодною стрічкою.

ESP32 отримує цей запит та використовує розбір URL для виділення значень R, G і B. Значення цих параметрів вказують на інтенсивність червоного (R), зеленого (G) і синього (B) компонентів вибраного кольору. Наприклад, у вищезазначеному URL синій колір є максимально насиченим (255), тоді як зелений і червоний рівні нулю.

Отримавши значення R, G і B, ESP32 використовує їх для генерації відповідного аналогового сигналу на GPIO пінах, які керують світлодіодною стрічкою. Зазвичай, для керування кольорами світлодіодів використовують техніку широкоімпульсної модуляції (ШІМ), що дозволяє регулювати інтенсивність світіння світлодіодів в широкому діапазоні.

Такий підхід дає можливість створювати різноманітні освітлювальні ефекти, такі як зміна кольору, плавні переходи між кольорами, створення анімацій тощо. Це відкриває широкі можливості для індивідуального керування освітленням та створення атмосфери в приміщенні з використанням світлодіодної стрічки.

Принцип роботи полягає в тому, що мікроконтролер ESP32 взаємодіє зі світлодіодною стрічкою через цифрові виводи, використовуючи спеціальні бібліотеки для керування кольорами та іншими параметрами світлодіодів. Такий підхід дозволяє створювати різноманітні освітлювальні ефекти та керувати ними за допомогою програмного забезпечення.

### 4.1.2 Робота приладу

Процес підключення ESP32 до транзисторів S8050 та LED стрічки за допомогою GPIO 12, 13 та 14 включає в себе наступні кроки. Спочатку ESP32 пов'язується з джерелами живлення та заземлення через контакти VCC та GND відповідно. Далі, обираються GPIO 12, 13 та 14 для керування транзисторами та LED стрічкою.

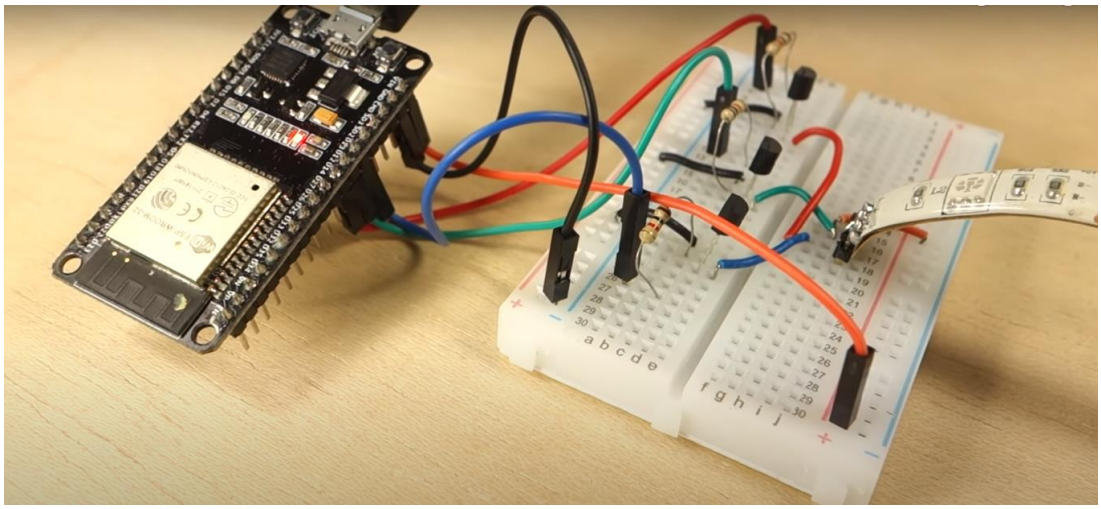

Рисунок 4.2 – Підключення до ESP32.

Три транзистори S8050 підключаються до ESP32. Бази транзисторів приєднуються до GPIO 12, 13 та 14 відповідно. Емітери заземлюються через GND на ESP32. Колектори транзисторів підключаються до анодів (+) LED стрічки.

Катоди (-) LED стрічки підключаються до заземлення (GND) ESP32. Також, важливо забезпечити необхідний струм та напругу для живлення LED стрічки через відповідне джерело.

Ця конфігурація дозволяє ESP32 ефективно керувати транзисторами S8050 за допомогою обраних GPIO пінів, регулюючи потужність живлення LED стрічки та контролюючи її світлове випромінювання. Такий підхід дозволяє створювати різноманітні світлові ефекти та динамічне освітлення.

В даній конфігурації ESP32 виступає як мозковий центр, керуючи світлодіодною стрічкою через транзистори S8050. Кожен транзистор відповідає за один регіон LED стрічки і дозволяє незалежно управляти кожним сегментом.

Коли сигнал подається на базу кожного транзистора через відповідний GPIO пін, він відкриває транзистор, дозволяючи електричному струму пройти через колектор і анод світлодіода. Це створює ефект включення частини LED стрічки, контрольованого програмним забезпеченням, що виконується на ESP32.

Крім того, обрані GPIO піни можуть бути програмовані для зміни інтенсивності світіння, робити плавні переходи між кольорами чи створювати різноманітні світлові ефекти відповідно до задачі проекту. Ця конфігурація відкриває широкі можливості для реалізації різноманітних освітлювальних сценаріїв в залежності від творчого підходу та вимог користувача.

Після введення мережевих облікових даних обираємо потрібну плату і COM-порт та завантажуємо код у ESP32.

Після завантаження відкриваємо послідовний монітор на швидкості 115200 і натискаємо кнопку ESP Enable/Reset. Ми повинні отримати IP-адресу плати.

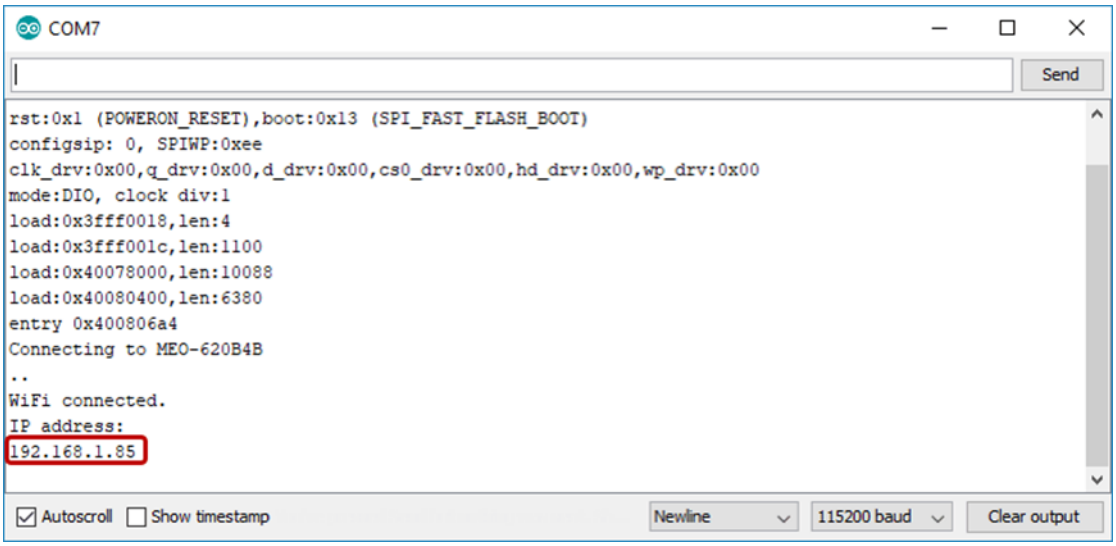


Рисунок 4.3 – Підключення до плати.

Відкриваємо браузер і вводимо IP-адресу ESP. Тепер за допомогою палітри кольорів обираємо колір для смуги.

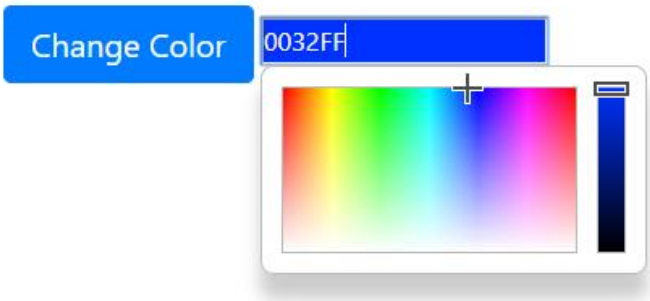


Рисунок 4.4 – Вибір палітри кольорів для смуги.

Потім потрібно натиснути кнопку "Змінити колір", щоб колір набув чинності.


Рисунок 4.5 – Робота приладу

Щоб вимкнути RGB світлодіодну стрічку, потрібно обрати чорний колір. Найяскравіші кольори (у верхній частині палітри кольорів) дають найкращі результати.

Тепер ми можемо використоувати стрічку, щоб прикрасити свій будинок: під ліжком, за телевізором, під кухонною шафою і багато чого іншого.

**Висновки до розділу 4**

У контексті розробленого проєкту було продемонстровано успішне поєднання Arduino-програмування та використання ESP32 мікроконтролера для керування LED стрічкою. Використання вбудованого веб-сервера для вибору кольору робить систему зручною та доступною для використання в різних сценаріях, де важлива дистанційна або програмна інтеракція.

У данному розділі розглянуто процес взаємодії між мікроконтролером ESP32 та веб-сервером з метою керування LED стрічкою. Завдяки цій інтеграції, користувач отримує можливість маніпулювати кольорами та інтенсивністю світіння світлодіодної стрічки через зручний веб-інтерфейс.

ESP32 виступає як центральний контролер, який приймає HTTP-запити від браузера та обробляє їх, розпізнаючи параметри кольорів R, G і B. З отриманих значень ESP32 вирішує, які інтенсивності сигналів потрібно відправити на GPIO піни, які управляють світлодіодною стрічкою.

Цей підхід не лише забезпечує зручний та віддалений доступ до керування освітленням, але й відкриває можливість для творчого впливу на атмосферу приміщення. Користувач може ефективно змінювати кольори, встановлювати плавні переходи між ними та створювати різноманітні світлові ефекти.

Важливим аспектом цього проекту є його модульність та гнучкість. Можливість взаємодії з LED стрічкою через веб-сервер дозволяє розширювати функціонал та вдосконалювати систему. Крім простого змінення кольору, можна додати додаткові функції, такі як таймери для автоматичного включення та вимикання освітлення, зміна яскравості згідно з розкладом дня чи взаємодія з іншими "розумними" пристроями у домашньому середовищі.

Цей проект також відкриває можливість використання датчиків світла чи температури для автоматичного регулювання освітлення, що забезпечує енергоефективність та забезпечує комфортне оточення. Узгоджена робота мікроконтролера та веб-сервера розширює сферу можливостей та надає користувачеві інструмент для творчого вираження через взаємодію з освітленням в їхньому просторі.

За допомогою цього проєкту вдалося показати практичну реалізацію інтернету речей у вигляді управління освітленням через мережу Інтернет. Він може слугувати важливим компонентом для подальших розробок та досліджень в галузі автоматизації, вбудованих систем та розумного освітлення.

Цей проект не лише використовує технологічні можливості ESP32 та Arduino для ефективного управління RGB LED стрічкою, але також реалізує зручний та інтуїтивно зрозумілий інтерфейс через веб-сервер. Вибір кольору

через веб-інтерфейс дозволяє користувачам легко персоналізувати освітлення відповідно до своїх уподобань та потреб.

Проект може бути використаний як важлива складова для подальших досліджень у сфері розумного освітлення та вбудованих систем, де важливо мати зручні та ефективні засоби керування освітленням. Також, з огляду на використання ESP32, він показує можливості високопродуктивного мікроконтролера з вбудованим Wi-Fi, що робить його ідеальним вибором для застосувань в Інтернеті речей.

Завдяки легкій розширюваності та гнучкості коду, цей проект може слугувати основою для подальших вдосконалень та розширень, таких як додавання нових функцій, інтеграція з іншими пристроями або розвиток більш складних сценаріїв автоматизації.

# ВИСНОВКИ

У ході магістерської роботи, на тему «Система та протоколи керування віддаленими IoT пристроями», було проведено детальний аналіз сучасних підходів до управління Internet of Things (IoT) пристроями в віддалених мережах. Досліджено різноманітні системи та протоколи, які використовуються для керування та взаємодії з IoT– пристроями, а також проведено порівняльний аналіз їхніх переваг та обмежень.

Однією з ключових висновків є те, що розвиток IoT технологій вимагає вдосконалення систем керування, щоб забезпечити ефективну взаємодію між великою кількістю розподілених пристроїв. Це важливо як для підприємств, що використовують IoT у виробничих процесах, так і для домогосподарств, які використовують різноманітні розумні пристрої.

Магістерська робота дозволила глибше розуміти принципи роботи різних систем керування та їхній вплив на функціональність IoT екосистем. Зазначено, що вибір конкретної системи чи протоколу має бути обдуманим рішенням, враховуючи конкретні вимоги та особливості конкретного застосування.

Також, важливо відзначити необхідність стандартизації та уніфікації в галузі керування віддаленими IoT пристроями для підвищення сумісності та спрощення інтеграції різноманітних пристроїв у єдину мережу.

Загалом, кваліфікаційна магістерська робота дала можливість здобути важливі практичні навички у сфері IoT, розширити теоретичні знання та поглибити розуміння проблем та перспектив розвитку систем та протоколів керування віддаленими IoT пристроями.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Інтернет речей: мережева архітектура та архітектура безпеки. URL: https://www.bizmaster.xyz/2020/12/internet-rechei-merezheva-arkhitektura-ta-arkhitektura-bezpeky.html (дата звернення: 15.12.2023).

2. The Internet of Things by Samuel Greengard. URL: https://www.goodreads.com/book/show/23461464– the– internet– of– things (Last accessed: 16.12.2023).

3. Learning Internet of Things by Peter Waher. URL: https://www.perlego.com/book/117850/learning– internet– of– things– pdf (Last accessed: 17.12.2023).

4. Internet of Things (IoT). URL: https://www.internetsociety.org/iot/?gclid=CjwKCAiAqY6tBhAtEiwAHeRope59YYhZ8z5SsOdKa7nrYrnlBcLIbwswQNvuZK7Pxvt8lZE6ZYhqVhoCFjEQAvD_BwE (Last accessed: 17.12.2023).

5. Timothy Chou. Precision: Principles, Practices and Solutions for the Internet of Things. 2016. 312 с. (Last accessed: 19.12.2023).

6. Erik Brynjolfsson and Andrew McAfee. The Second Machine Age: Work, Progress and Prosperity in a Time of Brilliant Technologies. 2014. 320 с. (Last accessed: 19.12.2023).

7. Нові протоколи зв'язку для IoT та систем безпеки. URL: https://worldvision.com.ua/populyarnost– iot– ustroystv– vozrastaet– v– kommercheskoy– srede/ (дата звернення: 22.12.2023).

8. Архітектура інтернет речей. URL: https://nau.edu.ua/download/Quality%20Assurance_ukr/Silabus/2020/b/2/123/S_b_2_123_4_%D0%90%D1%80%D1%85%D1%96%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0%20.pdf (дата звернення: 22.12.2023).

9. Використання хмарних сервісів для IoT. URL: https://pupenasan.github.io/TI40/%D0%9B%D0%B5%D0%BA%D1%86/cloud.html (дата звернення: 24.12.2023).

10. Ефективність та безпека в централізованих системах керування IoT. URL: https://learn.ztu.edu.ua/pluginfile.php/162168/mod_resource/content/1/%D0%9B15%20%20%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D0%BA%D0%B0–%D0%B7%D0%B0%D0%B3.pdf (дата звернення: 24.12.2023).

11. Paperback Bunko. The Official ESP32 Book. 2017. 286 c. (Last accessed: 27.12.2023).

12. Vedat Ozan Oner. Developing IoT Projects with ESP32: Automate your home or business with inexpensive Wi-Fi devices. 2021. 474 c. (Last accessed: 29.12.2023).

13. Massimo Banzi. Getting Started With Arduino. The Open Source Electronics Prototyping Platform. 2022. 270 c. (Last accessed: 03.01.2024).

14. Vedat Ozan Oner. Developing-IoT-Projects-with-ESP32. 2023. 578 c. (Last accessed: 05.01.2024).

15. Arduino IDE documentation. URL: https://docs.arduino.cc/software/ide/ (Last accessed: 05.01.2024).

16. Craig Hunt. TCP/IP Network Administration. 2002. 746 c. (Last accessed: 07.01.2024).

17. What is TCP/IP and How Does it Work? URL: https://www.techtarget.com/searchnetworking/definition/TCP-IP#:~:text=TCP%2FIP%20stands%20for%20Transmission,(an%20intranet%20or%20extranet) (Last accessed: 07.01.2024).

18. SS8050 NPN Epitaxial Silicon Transistor. URL: https://www.mouser.com/datasheet/2/149/SS8050-117753.pdf (Last accessed: 12.01.2024).

19. Agus Kurniawan. Internet of Things Projects with ESP32: Build exciting and powerful IoT projects using the all-new Espressif ESP32. 2019. 252 c. (Last accessed: 12.01.2024).

20. Що таке ESP? Детальний аналіз. URL: https://cyberdid.medium.com/%D1%89%D0%BE-%D1%82%D0%B0%D0%BA%D0%B5-esp-d1267d5877cf (Last accessed: 12.01.2024).

# Додаток А

# Лістінг програмного коду

```
#include <WiFi.h>

const char* ssid     = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";

WiFiServer server(80);

String redString = "0";
String greenString = "0";
String blueString = "0";
int pos1 = 0;
int pos2 = 0;
int pos3 = 0;
int pos4 = 0;

String header;

const int redPin = 13;
const int greenPin = 12;
const int bluePin = 14;

const int freq = 5000;
const int redChannel = 0;
const int greenChannel = 1;
const int blueChannel = 2;
// Bit resolution 2^8 = 256
const int resolution = 8;

unsigned long currentTime = millis();
unsigned long previousTime = 0;
const long timeoutTime = 2000;

void setup() {
```

```
  Serial.begin(115200);
  ledcSetup(redChannel, freq, resolution);
  ledcSetup(greenChannel, freq, resolution);
  ledcSetup(blueChannel, freq, resolution);

  ledcAttachPin(redPin, redChannel);
  ledcAttachPin(greenPin, greenChannel);
  ledcAttachPin(bluePin, blueChannel);

  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected.");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  server.begin();
}

void loop(){
  WiFiClient client = server.available();

  if (client) {
    currentTime = millis();
    previousTime = currentTime;
    Serial.println("New Client.");
    String currentLine = "";
    while (client.connected() && currentTime - previousTime <= timeoutTime) {
// loop while the client's connected
      currentTime = millis();
      if (client.available()) {
        char c = client.read();
        Serial.write(c);
```

```
        header += c;
        if (c == '\n') {
          if (currentLine.length() == 0) {

            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println("Connection: close");
            client.println();

            client.println("<!DOCTYPE html><html>");
            client.println("<head><meta                    name=\"viewport\"
content=\"width=device-width, initial-scale=1\">");
            client.println("<link rel=\"icon\" href=\"data:,\">");
            client.println("<link                        rel=\"stylesheet\"
href=\"https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.c
ss\">");
            client.println("<script
src=\"https://cdnjs.cloudflare.com/ajax/libs/jscolor/2.0.4/jscolor.min.js\"><
/script>");
            client.println("</head><body><div       class=\"container\"><div
class=\"row\"><h1>ESP Color Picker</h1></div>");
            client.println("<a  class=\"btn  btn-primary  btn-lg\"  href=\"#\"
id=\"change_color\" role=\"button\">Change Color</a> ");
            client.println("<input                          class=\"jscolor
{onFineChange:'update(this)'}\" id=\"rgb\"></div>");
            client.println("<script>function                  update(picker)
{document.getElementById('rgb').innerHTML = Math.round(picker.rgb[0]) + ', ' +
Math.round(picker.rgb[1]) + ', ' + Math.round(picker.rgb[2]);");

client.println("document.getElementById(\"change_color\").href=\"?r\"       +
Math.round(picker.rgb[0])  +  \"g\"  +   Math.round(picker.rgb[1])  +  \"b\"  +
Math.round(picker.rgb[2]) + \"&\";}</script></body></html>");
            client.println();

            if(header.indexOf("GET /?r") >= 0) {
              pos1 = header.indexOf('r');
              pos2 = header.indexOf('g');
```

```
            pos3 = header.indexOf('b');
            pos4 = header.indexOf('&');
            redString = header.substring(pos1+1, pos2);
            greenString = header.substring(pos2+1, pos3);
            blueString = header.substring(pos3+1, pos4);
            /*Serial.println(redString.toInt());
            Serial.println(greenString.toInt());
            Serial.println(blueString.toInt());*/
            ledcWrite(redChannel, redString.toInt());
            ledcWrite(greenChannel, greenString.toInt());
            ledcWrite(blueChannel, blueString.toInt());
          }
          break;
        } else {
          currentLine = "";
        }
      } else if (c != '\r') {
        currentLine += c;
      }
    }
  }
  header = "";
  client.stop();
  Serial.println("Client disconnected.");
  Serial.println("");
 }
}
```