Міністерство освіти і науки України

Луцький національний технічний університет
(повне найменування закладу вищої освіти)

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

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

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

SMART-СИСТЕМА ВІДОБРАЖЕННЯ ІНФОРМАЦІЇ НА БАЗІ МІКРОКОНТРОЛЕРА СЕРІЇ ESP

SMART INFORMATION DISPLAY SYSTEM BASED ON ESP SERIES MICROCONTROLLER

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

освітня програма Комп'ютерна інженерія
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІс-21
Никитюк Артем Олегович

(підпис)

Керівник:
асистент
Поліщук Лілія Олексіївна

(підпис)

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

(підпис)

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

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

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Никитюку Артему Олеговичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Smart-система відображення інформації на базі мікроконтролера серії ESP*

Керівник роботи *асистент, Поліщук Лілія Олексіївна*

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

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

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
Вступ
Аналіз сучасних рішень у галузі Смарт-систем
Апаратне та програмне забезпечення Смарт-систем
Розробка програмного забезпечення Смарт-систем
Висновки

5. Перелік графічного (ілюстративного) матеріалу:

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз сучасних рішень у галузі Смарт-систем	*Поліщук Л.О., асистент*		
Апаратне та програмне забезпечення Смарт-систем	*Поліщук Л.О., асистент*		
Розробка програмного забезпечення Смарт-систем	*Поліщук Л.О., асистент*		
Нормоконтроль	*Багнюк Н.В., доцент*		
Гарант ОП	*Лавренчук С.В., доцент*		
Показник запозичень тексту	***___%***		
Академічна доброчесність	*Міскевич О.І.,ст.викладач*		

7. Дата видачі завдання 10.01.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	*Огляд літератури із досліджуваної проблеми, аналіз предметної області та наявних рішень*	до 15.02.2025 р.	Виконано
2.	*Аналіз сучасних рішень та проектування апаратної складової пристрою*	до 15.03.2025 р.	Виконано
3.	*Практична реалізація Smart-системи відображення інформації на базі мікроконтролера серії esp*	до 04.05.2025 р.	Виконано
4.	*Висновки та пропозиції*	до 07.05.2025 р.	Виконано
5.	*Формування списку використаних джерел*	до 10.05.2025 р.	Виконано
6.	*Формування додатків*	до 15.05.2025 р.	Виконано
7.	*Оформлення ілюстративного матеріалу*	до 20.05.2025 р.	Виконано
8.	*Нормоконтроль*	до 01.06.2025 р.	Виконано
9.	*Інструментальна перевірка на академічний плагіат*	до 04.06.2025 р	Виконано
10.	*Здача кваліфікаційної роботи та всіх супровідних документів на кафедру*	до 11.06.2025 р.	Виконано

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

Керівник кваліфікаційної роботи _____ Поліщук Л.О.
(підпис) (прізвище, ініціали)

АНОТАЦІЯ

Никитюк А. О. Smart-система відображення інформації на базі мікроконтролера серії ESP.

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

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

Перший розділ присвячений аналізу технологій, архітектури та можливостей серії ESP та сфери їх використання, також проведені протоколи передачі даних у системах IoT та інтеграція Smart-системи з хмарними платформами.

В другому розділі здійснено вибір компонентів та обґрунтування засобів розробки. Створення апаратної частини смарт-системи, а саме схеми підключення, збору прототипу та тестування його роботи для забезпечення ефективності.

Третій розділ присвячено створенню ефективного програмного забезпечення для керування смарт-системою та збору і відображенню даних.

Ключові слова: смарт-система, компоненти, технології, ESP, мікроконтролери, відображення інформації, розумні технології, дані.

ANNOTATION

Nukutiuk A. Smart-information display system based on esp series microcontroller.

Qualification work of the bachelor's degree in Computer Engineering, specialty 123 Computer Engineering. Lutsk National Technical University. Lutsk, 2025.

The qualification work consists of an introduction, three chapters, conclusions, a list of references, and appendices.

The first section is devoted to the analysis of technologies, architecture and capabilities of the ESP series and their use, as well as data transmission protocols in IoT systems and integration of the Smart system with cloud platforms.

The second section describes the selection of components and the generalization of development tools. The creation of the hardware part of the smart system is described, namely, the connection scheme, prototype assembly, and testing of its operation to ensure efficiency.

The third section is devoted to the creation of effective software for managing the smart system and collecting and displaying data.

Keywords: smart system, components, technologies, esp, microcontrollers, information display, smart technologies, data.

ЗМІСТ

ВСТУП

У сьогоднішньому світі цифрові технології проникають майже в всі існуючі галузі нашого життя інтегруючись з різними сферами. Особливої актуальності набирають смарт-системи вони комбінуються з різними сферам та створюють розумні рішення. Одним із таких розумних пристроїв є смарт-система відображення інформації.

Однією із найпопулярніших платформ для реалізації смарт-систем є мікроконтролер ESP-01. Завдяки своїм компактним розмірам, зрозумілій архітектурі, вбудованому модулю Wi-Fi та інтеграцією з широкому вибору бібліотек мікроконтролер ESP виявився найкращим варіантом серед конкурентів для реалізації цього проєкту.

Актуальність теми зумовлена стрімкому зростанню популярності смарт-систем в сучасному світі та інтеграцією з телефонами для відображення інформації.

Метою роботи проектування та розробка пристрою який буде використовуватись в сучасному житті для відображення необхідної інформації користувачу.

Об’єкт дослідження – інформаційні смарт-технології, які включають аналіз технологічної архітектури, методів збору та аналізу даних, алгоритмів прийняття рішень у режимі реального часу, а також інтеграцію з іншими платформами та зрозумілий інтерфейс для взаємодії з користувачем.

Предмет дослідження – смарт-система яка поєднує у собі апаратне забезпечення, програмне забезпечення, аналіз та обробку даних.

Завдання, які необхідно виконати:

– реалізувати інтеграцію мікроконтролера з іншими елементами;

– розробити алгоритм для збору інформації в режимі реального часу;

– дослідити методи оптимізації автономності системи пристрою для її портативності;

– запропонувати рішення для синхронізації роботи смарт-системи з мобільними пристроями.

Никитюк А, Поліщук Л. Інтеграція Smart-систем з хмарними платформами. Програмне та апаратне забезпечення в інформаційних технологіях: матеріали Міжнар. наук.-практ. конф. молодих вчених та студентів (6 травня 2025 р). / відп. ред. Т.В. Терлецький. Вип. 1. Луцьк: ЛНТУ, 2025. С. 134-136 [1].

РОЗДІЛ 1

АНАЛІЗ СУЧАСНИХ РІШЕНЬ У ГАЛУЗІ СМАРТ-СИСТЕМ

1.1 Огляд технологій створення Smart-систем

Сучасний світ стрімко занурюється у хвилі цифрової трансформації, де смарт-технології стають невід'ємною складовою багатьох галузей. Розумні системи, які об'єднують апаратно-програмні рішення, стають не лише модними тенденціями, а й основними складовими майбутньої інфраструктури. Їх впровадження створює нові можливості для управління ресурсами, оптимізації процесів і підвищення зручності для кінцевих користувачів.

Одним із ключових принципів розробки смарт-систем є інтегративний підхід. Інтегративний підхід це поняття яке передбачає синергію окремих компонентів, де навіть найменші елементи сприяють створенню значно масштабніших рішень. Від мікроконтролерів серії ESP, які завдяки своїй компактності та енергоефективності використовуються в багатьох проектах, до складних програмних платформ, що організовують обробку даних – кожен елемент виконує свою унікальну роль. Такий підхід дозволяє створити систему, здатну самостійно адаптуватися до умов експлуатації, що є особливо важливим на фоні швидкого розвитку індустрії Інтернету Речей (IoT).

Окрім апаратного забезпечення, важливу роль відіграє й програмне забезпечення, яке інтегрує всі компоненти системи в єдиний механізм для обробки та візуалізації даних. Сучасні технології дозволяють обробляти великі обсяги інформації, що передається між різними елементами системи, роблячи її більш гнучкою та адаптивною до змін у навколишньому середовищі. Це досягається завдяки використанню передових алгоритмів, високорівневих протоколів зв'язку та методів попереднього аналізу даних.

Сучасна парадигма розвитку смарт-технологій також включає концепцію цифрової трансформації, яка виходить за межі чисто технічних аспектів. Вона передбачає зміну управлінських процесів, нові підходи до взаємодії з користувачами та створення нового рівня комунікації між людиною і машиною.

Ця трансформація підкреслює важливість гнучкості, масштабованості та адаптивності систем, що дозволяє максимально відповідати сучасним вимогам як індустрії, так і приватного сектору. Інновації та експерименти стають рушійною силою для подальшого вдосконалення технологічних рішень, які охоплюють як традиційні сфери, так і відкривають нові можливості для інтеграції в повсякденне життя.

Інноваційність смарт-систем полягає в постійному пошуку нових рішень. Основні дослідження в електроніці, програмуванні та штучному інтелекті відкривають можливості для створення ще більш адаптивних, розумних і надійних систем. Як наслідок, інтеграція локальних обчислювальних потужностей із глобальними платформами дозволяє формувати унікальні екосистеми, здатні ефективно обробляти інформацію навіть у найскладніших умовах. Такий підхід дозволяє створювати не лише окремі пристрої, а й цілі мережі, де кожен компонент виконує свою роль у загальному управлінні системою. Розробка ефективних смарт-систем є важливим напрямком у багатьох сферах, від промисловості та енергетики до медицини та міського господарства.

Смарт-система, у своїй суті, є динамічною, інтегрованою платформою, яка об'єднує фізичні об'єкти та цифрові процеси для досягнення певних цілей, використовуючи інтелектуальні можливості. Це визначення підкреслює не просто наявність окремих «розумних» пристроїв, а саме системний підхід, де компоненти взаємодіють між собою, утворюючи єдине ціле, здатне до інтелектуальної діяльності. Ключовими елементами смарт-технологій є комп'ютерні системи та мікропроцесори, які, спираючись на програми, опрацьовують дані, що надходять від датчиків, через дистанційне керування або мережу інтернет [2].

Інтелект смарт-системи полягає не у статичних алгоритмах, а в умінні навчатися, адаптуватися та оптимізуватися з часом, що робить їх більш ефективними та стійкими до змін зовнішнього середовища.

В таблиці 1.1 написано список галузей де використовуються смарт-системи.

Таблиця 1.1 – Використання смарт-технологій в галузях

Галузь	Опис застосування смарт-технологій
Розумні міста(Смарт-сіті)	Використовують інтелектуальні системи моніторингу трафіку, управління громадським транспортом, контролю якості повітря та роботи комунальних служб, а саме (управлінням електроенергією та системою управлінням водопостачанням) також оптимізують міську інфраструктуру та сервіс.
Розумні будинки	Використовують системи автоматизації такі як освітлення, опалення, вентиляцію, безпеку, доступ і енергозбереження створюючи комфортне та ефективне середовище.
Освіта та наука	Використовують інтерактивні навчальні платформи, дистанційне навчання, застосування віртуальної та доповненої реальності для персоналізації навчального процесу і підвищення якості освіти.
Аграрний сектор	Використовують землеробство із застосуванням датчиків для контролю ґрунту, погодних умов, іригаційних систем і використання безпілотних апаратів для оптимізації використання ресурсів і підвищення врожайності.
Торгівля та маркетинг	Використовують персоналізацію маркетингових стратегій, оптимізація управління запасами та автоматизація процесів продажу, що підвищують конкурентоспроможність підприємств.
Фінанси та банківська справа	Використовують штучний інтелект та блокчейн-технологій для забезпечення безпеки операцій, аналізу транзакцій, виявлення шахрайства і автоматизації процесів управління ризиками.
Охорона здоров’я та медицина	Використовують дистанційний моніторинг стану пацієнтів, телемедицина, використання носимих пристроїв і алгоритмів аналізу для ранньої діагностики та оптимізації медичного догляду

Продовження таблиці 1.1

Промисловість та виробничі системи	Використовують інтегровані рішення Industry 4.0, які включають сенсори, автоматизацію виробництва, аналіз даних і прогнозну діагностику для підвищення продуктивності та зниження витрат
Безпека та захисні технології	Використовують системи відеоспостереження, розпізнавання облич, контролю доступу та аналізу подій у режимі реального часу, що забезпечують більш ефективний захист приватних і громадських просторів.

Сучасні технології стрімко розвиваються, а смарт-системи стають невід'ємною складовою сучасного цифрового світу. Вони відіграють важливу роль у впровадженні автоматизації, оптимізації управління ресурсами, інтеграції різних пристроїв та вдосконаленні бізнес-процесів. Завдяки своїй здатності ефективно аналізувати великі обсяги даних, ці системи радикально змінюють спосіб, яким люди взаємодіють з технологіями. Вони створюють основу для впровадження адаптивних і гнучких рішень у різноманітних завданнях. Цей технологічний прорив відкриває нові можливості для модернізації як побутових, так і промислових процесів, що, в свою чергу, сприяє значному підвищенню продуктивності та оптимальному використанню ресурсів.

Новітні смарт-системи характеризуються високим рівнем автоматизації, що дозволяє їм самостійно виконувати завдання відповідно до заданих сценаріїв або навіть адаптувати свою поведінку на основі аналізу даних у реальному часі. Вони можуть працювати автономно, що зменшує потребу в постійному втручанні людини, що є особливо важливим для систем, де швидкість реакції та стабільність роботи є критично важливими.

Ще однією важливою характеристикою є їхня адаптивність. Системи постійно аналізують умови навколишнього середовища та коригують свою роботу відповідно до уподобань користувачів і зовнішніх факторів. Завдяки

цьому вони не лише демонструють ефективність у стабільних умовах, а й успішно справлятися з викликами, що виникають у динамічному середовищі.

Основною перевагою смарт-систем є їх здатність до інтелектуального аналізу даних, що дозволяє перетворювати інформацію на конкретні рішення. Завдяки використанню алгоритмів машинного навчання та штучного інтелекту, ці системи можуть прогнозувати події, виявляти аномалії та оптимізувати робочі процеси, що суттєво підвищує загальну ефективність роботи

Також важливою є інтеграція в мережеву екосистему. Мережеве підключення дозволяє окремим пристроям спілкуватися між собою, формуючи єдину систему, в якій дані збираються, обробляються та використовуються для глибшого аналізу і прийняття оперативних рішень.

Не залишаються поза увагою і питання безпеки та конфіденційності. Використання сучасних методів шифрування, автентифікації та інших засобів захисту даних гарантує високий рівень збереження інформації, що є надзвичайно важливим у цифрову епоху.

Ефективне управління ресурсами, зокрема енергозбереження, також є важливим аспектом функціонування таких систем. Використання спеціалізованих алгоритмів дозволяє оптимізувати споживання енергії, що не лише знижує операційні витрати, але й має позитивний вплив на екологічну ситуацію.

Зручність користування визначає успіх впровадження смарт-технологій. Інтуїтивно зрозумілі інтерфейси, можливість користуватися голосовими командами та мобільними додатками роблять взаємодію з системою легкою та комфортною, що сприяє її широкому застосуванню в різних сферах життя.

Крім технологічних переваг смарт-системи сприяють формуванню нового соціально-економічного середовища. Смарт-системи є основним чинником створення «розумних міст», де сучасні технологічні рішення оптимізують транспортні потоки, технології спостереження, енергопостачання та управління комунальними ресурсами зображені на рисунку 1.1.

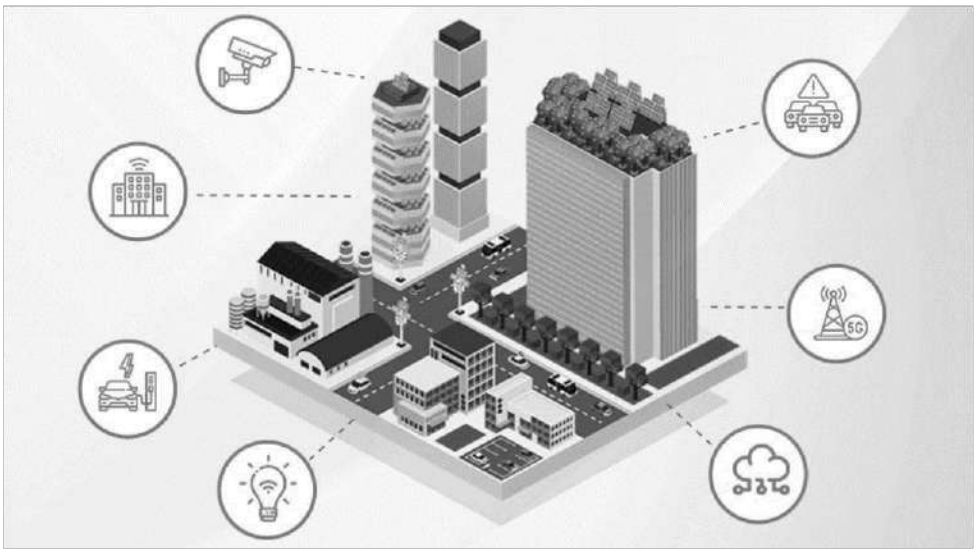


Рисунок 1.1 – Смарт-сіті і використання їх технологій в місті [3]

Завдяки інтеграції з іншими інноваційними технологіями, ці системи допомагають вирішувати глобальні виклики, зокрема питання сталого розвитку та ефективного використання природних ресурсів.

Отже, смарт-системи не лише сприяють швидшому технологічному розвитку, але й створюють нові можливості для стійкого, ефективного та інклюзивного майбутнього, де технологічний прогрес поєднується з підвищенням якості життя та охороною природних ресурсів.

1.2 Архітектура мікроконтролерів та їх можливості

Мікроконтролер – це компактний інтегральний електронний пристрій, який містить процесор, оперативну пам’ять, постійну пам’ять та периферійні інтерфейси на одній мікросхемі [4].

Мікроконтролер призначений для автоматизованого управління електронними системами та пристроями, такими як побутова техніка, автомобілі, роботи та промислове обладнання, приклад використання мікроконтролерів у пристроях зображено на рисунку 1.2.

Рисунок 1.2 – Використання мікроконтролера в таких пристроях [5]

На відміну від звичайних комп'ютерів, які призначені для різноманітних завдань, мікроконтролери мають специфічну спеціалізацію. Вони функціонують за визначеною програмою, яка залишається незмінною протягом усього часу роботи пристрою. Це забезпечує високу ефективність, економне споживання енергії та відсутність потреби в складній операційній системі. Основними складовими мікропроцесора є регістри, арифметико-логічний пристрій та блок керування. Регістри використовуються для тимчасового зберігання даних, тоді як арифметико-логічний пристрій виконує арифметичні та логічні операції, обробляючи дані. Блок керування забезпечує послідовне виконання команд програми та коректне спрямування потоків даних [6].

Мікроконтролери використовуються у багатьох сферах нашого життя наприклад у побутових приладах мікроконтролери управляють холодильниками, пральними машинами та кондиціонерами, забезпечуючи їх ефективну роботу. У автомобілях вони контролюють двигун, систему антиблокування гальм (ABS) та паркувальні сенсори, приклад де використовуються мікроконтролери в автомобілі зображений на рисунку 1.3.

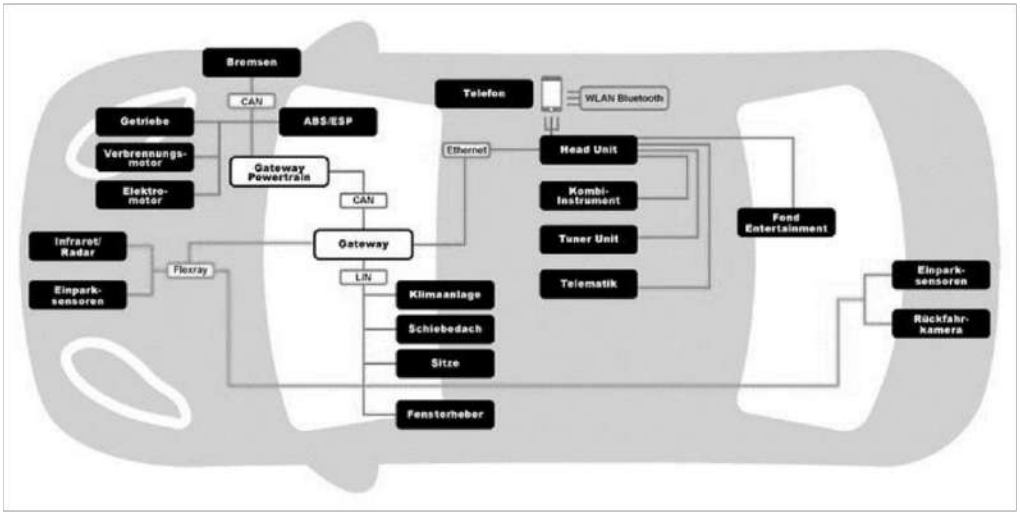


Рисунок 1.3 – Використання мікроконтролерів в автомобілі [7]

В IoT відкриває можливості для створення розумних будинків, де мікроконтролери ESP8266 та ESP32 автоматизують освітлення, контроль клімату та забезпечення безпеки. У медичній сфері ці технології використовуються в кардіостимуляторах, пристроях для вимірювання рівня кисню та аналізаторах крові. У виробництві мікроконтролери управляють роботизованими системами та автоматизованими лініями. У сфері робототехніки вони відповідають за навігацію дронів, автономних пристроїв і різноманітних технологічних рішень.

Мікроконтролери розрізняються за розрядністю, продуктивністю, енергоспоживанням та сферою застосування. Найпростіші з них – 8-бітні моделі, які виконують базові завдання і споживають мало енергії. Вони часто знаходять застосування в побутових пристроях, таких як мікрохвильові печі, терморегулятори та датчики руху. Серед найпопулярніших серій можна виділити ATmega який зображений на рисунку 1.4, ATmega використовується в платах Arduino, та PIC, який широко застосовується в автомобільній електроніці та простих автоматизованих системах.

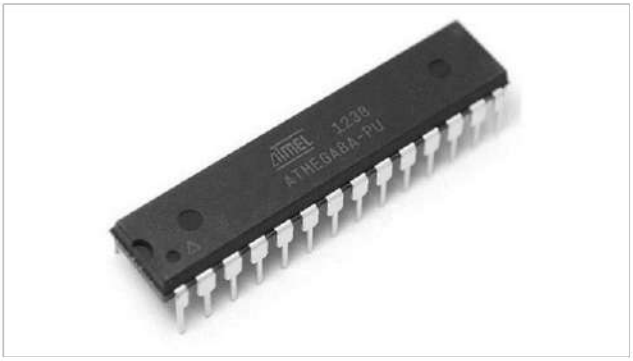


Рисунок 1.4 – Мікроконтролер 8-бітної серії ATMEGA [8]

Більш продуктивними є 16-бітні мікроконтролери, які забезпечують точнішу обробку даних і можливість реалізації складніших алгоритмів. Вони використовуються в медичних пристроях, системах управління двигунами та промисловій автоматизації. Серед найвідоміших моделей цієї категорії можна виділити MSP430 від Texas Instruments який зображений на рисунку 1.5, який оптимізований для енергоефективності та високої точності.

Рисунок 1.5 – Мікроконтролер 16-бітної серії MSP430 [9]

Найпотужнішими є 32-бітні мікроконтролери, що можуть виконувати складні обчислення та обробляти великі обсяги пам'яті. Вони знаходять застосування в робототехніці, розумних пристроях та промисловій автоматизації. Серед таких мікроконтролерів особливо популярні мікроконтролери серії ESP32 та ESP8266, завдяки вбудованим модулям Wi-Fi та Bluetooth, ці пристрої є ідеальним вибором для рішень в сфері Інтернету речей (IoT). Їх архітектура забезпечує швидку обробку даних, низьке

енергоспоживання та підтримує різноманітні периферійні інтерфейси, що робить ESP32 та ESP8266 універсальними і ефективними для інтеграції в сучасні технологічні проєкти. На рисунку 1.6 зображено модель ESP32, що дозволяє детальніше ознайомитися з її конструктивними особливостями та портами

Рисунок 1.6 – Мікроконтролер серії ESP32 з роз'ємом USB Type-c [10]

Таким чином, вибір мікроконтролера залежить від потреб системи для простих завдань підійдуть 8-бітні моделі, тоді як для складніших алгоритмів та IoT-розробок доцільніше використовувати 32-бітні або ARM-мікроконтролери. Сучасні технології значно розширюють функціональні можливості цих пристроїв, роблячи їх незамінними в автоматизації та впровадженні розумних рішень.

1.3 Мікроконтролери серії ESP

Мікроконтролери ESP – це інноваційна серія електронних компонентів, розроблена компанією Espressif Systems [11], яка стала справжнім проривом у сфері IoT. Головна особливість ESP мікроконтролера це вбудовані модулі Wi-Fi та Bluetooth, що забезпечують легку інтеграцію цих пристроїв у бездротові мережі без необхідності використання додаткових модулів зв'язку. Завдяки цим характеристикам мікроконтролери ESP знайшли широке застосування в автоматизації, моніторингу, розумних домах, промислових рішеннях та носимих пристроях.

Першим мікроконтролером, який здобув популярність у ESP серії, став ESP8266, що з'явився на ринку в 2014 році. Його головна перевага – вбудована підтримка Wi-Fi, що дозволило розробникам створювати бездротові пристрої без потреби в дорогих зовнішніх модулях. Завдяки своїй низькій вартості та доступності ESP8266 швидко став популярним серед розробників DIY-проєктів, хакерів, інженерів та ентузіастів.

Основні можливості ESP мікроконтролерів наведені на рисунку 1.7

Рисунок 1.7 – Основні можливості ESP мікроконтролерів [12]

Мікроконтролери ESP32 та ESP8266 є надзвичайно популярними рішеннями в галузі IoT, автоматизації та робототехніки. Обидва пристрої забезпечують бездротову комунікацію, низьке енергоспоживання та високу продуктивність, проте між ними існують суттєві відмінності, які можуть вплинути на вибір розробника. Порівняння найпопулярніших версій ESP між собою описані в таблиці 1.2

Таблиця 1.2 – Порівняння ESP32 та ESP8266 [13]

Характеристика	ESP32	ESP8266
Процесор	Два ядра Xtensa LX6, до 240 МГц	Одне ядро Xtensa LX3, до 80/160 МГц
Оперативна пам'ять (SRAM)	520 КБ	80 КБ
Флеш-пам'ять	До 16 МБ	До 4 МБ
Бездротовий зв'язок	Wi-Fi 802.11 b/g/n і Bluetooth BLE та Classic	Wi-Fi 802.11 b/g/n
Протокол ESP-NOW	Є	Немає
Кількість GPIO	30-36 контактів, залежно від моделі	17 контактів
Інтерфейси	SPI, I2C, UART, CAN, PWM, ADC, DAC	SPI, I2C, UART, PWM, ADC
Аналогові входи	18 каналів ADC, 2 DAC	1 канал ADC
Енергоспоживання	Високоефективні режими сну (Deep Sleep, Light Sleep, ULP)	Спрощений режим сну
Швидкість Wi-Fi	Вища (потужніша антена)	Нижча
Можливість програмування	Arduino IDE, ESP-IDF, PlatformIO, MicroPython	Arduino IDE, MicroPython

Порівнявши між собою ESP32 та ESP8266 я можу сказати що, якщо проєкт потребує високої продуктивності, багатопотоковості, розширених можливостей комунікації (Wi-Fi + Bluetooth) та гнучкого управління периферійними пристроями, ESP32 стане ідеальним вибором. Цей модуль дозволяє розробляти складні IoT-рішення, де важливі швидка реакція на зміни та захист даних.

У той же час, якщо займатися простим проєктом, таким як розумна розетка, термометр або мінімалістичне рішення для автоматизації, ESP8266 буде більш економічним вибором. Цей модуль споживає менше ресурсів, має компактні габарити і при цьому достатньо потужний для виконання багатьох стандартних завдань.

1.4 Протоколи передачі даних у системах IoT (Wi-Fi, Bluetooth, MQTT)

Розвиток технологій IoT відкрив нові можливості для створення смарт-систем, які можуть збирати, аналізувати та відображати інформацію в реальному часі. Однією з важливих складових таких систем є протоколи передачі даних, що забезпечують взаємодію між пристроями, сервісами та користувачами. У цьому розділі ми детально розглянемо найпоширеніші протоколи передачі даних у IoT, зокрема Wi-Fi, Bluetooth та MQTT. Кожен з цих протоколів має свої особливі характеристики та функції, які роблять його незамінним у певних ситуаціях, а також їх інтеграцію в смарт-системи.

Wi-Fi є одним з найпоширеніших протоколів передачі даних, що використовується в IoT-системах завдяки своїй високій швидкості та можливості підключення до глобальної мережі Інтернет. Мікроконтролери серії ESP, зокрема ESP8266 та ESP32, підтримують Wi-Fi, що робить їх важливими елементами для розробки IoT-рішень. Наприклад, мікроконтролер ESP може отримувати дані з серверів або інших пристроїв через Wi-Fi, обробляти їх і виводити на екран або інший пристрій. Цей протокол є ідеальним вибором для пристроїв, які потребують постійного підключення до Інтернету та обміну великими обсягами даних. Одночасно Wi-Fi споживає значну кількість енергії, що може стати недоліком для пристроїв з обмеженим живленням, наприклад, сенсорів. Проте його можливість інтегруватися в уже наявні інфраструктури робить його популярним вибором у багатьох проєктах Інтернету речей (IoT).

Bluetooth відомий своєю низькою енергоспоживаністю, що є важливою перевагою для пристроїв, які працюють на батарейках. Bluetooth Low Energy (BLE) – це спеціалізована версія цього протоколу[14], створена для Інтернет речей (IoT). Вона забезпечує передачу даних на відносно короткі відстані, що робить її ідеальним вибором для носимих пристроїв, «розумних» домашніх систем та медичних апаратів. Основною перевагою використання Bluetooth в IoT є його здатність швидко встановлювати з'єднання між пристроями, хоча варто враховувати обмежену дальність передачі.

MQTT (Message Queuing Telemetry Transport) є одним із найпопулярніших протоколів обміну даними в IoT. Він розроблений для передачі невеликих обсягів інформації через мережі з обмеженою пропускною здатністю або високою затримкою. Протокол працює за принципом «публікація/підписка», що дозволяє пристроям обмінюватися даними через центрального брокера. MQTT є ідеальним вибором для систем моніторингу та управління, таких як «розумне» сільське господарство або промисловий IoT, де стабільність і надійність передачі даних мають критичне значення. Головним недоліком MQTT є його залежність від центрального брокера. Вся система передачі даних у цьому протоколі зосереджена навколо брокера, який відповідає за обмін повідомленнями між клієнтами. Якщо брокер зазнає збоїв через технічні проблеми або атаки, система перестає функціонувати, що може призвести до втрати зв'язку, даних і навіть загального порушення роботи IoT-системи. Чим відрізняються Wi-Fi, Bluetooth та MQTT показані в таблиці 1.3

Таблиця 1.3 – Відмінності між Wi-Fi, Bluetooth та MQTT [15]

Параметр	Wi-Fi	Bluetooth	MQTT
Тип зв'язку/Архітектура	Працює за стандартами IEEE 802.11 потребує точки доступу	Peer-to-peer або мережі типу PAN	TCP/IP; використовує модель публікації–підписки
Діапазон дії	50–100 метрів залежить від маршрутизатора	до 10 метрів для деяких пристроїв може досягати 100 метрів	фактичний діапазон залежить від базової мережевої інфраструктури

Продовження таблиці 1.3

Параметр	Wi-Fi	Bluetooth	MQTT
Швидкість передачі даних	Висока (до декількох гігабіт/с у сучасних стандартів	Середня або низька швидкість, достатня	Призначено для передачі невеликих повідомлень
Споживання енергії	високий рівень енерговитрат	Низький рівень споживання (особливо в режимі BLE)	споживання енергії залежить від базової технології зв’язку
Особливості роботи	Забезпечує високошвидкісний інтернет-зв’язок	Простота створення з’єднання між пристроями та легкість інтеграції	Базується на моделі публікації–підписки, що дозволяє ефективно управляти потоками повідомлень із зворотним зв'язком

Інтеграція протоколів Wi-Fi, Bluetooth та MQTT відкриває нові можливості для створення комплексних систем, які по максимуму використовують переваги кожного з цих технологій. Проте важливо знати специфіку завдань, оскільки кожен протокол має свої сильні та слабкі сторони. Wi-Fi ідеально підходить для завдань, що потребують високої швидкості обробки великих обсягів даних, але може бути енергозатратним для деяких пристроїв. Bluetooth забезпечує надійний зв’язок на коротких відстанях з мінімальними енергетичними витратами, але не завжди підходить для ситуацій, де потрібна потужна передача даних. MQTT, завдяки своїй легкій архітектурі, дозволяє зменшити мережевий трафік і забезпечує швидкий обмін інформацією між сотнями пристроїв.

Окрім основних протоколів, важливо зазначити, що розвиток IoT спрямований на інтеграцію штучного інтелекту в процес обміну даними. Подальше вдосконалення стандартів протоколів дозволить пристроям не лише збирати та передавати інформацію, а й аналізувати її в реальному часі та приймати автономні рішення. Це створить нові можливості для оптимізації управління пристроями та підвищення їхньої безпеки.

Інтеграція з мікроконтролерами, такими як ESP8266, займає важливе місце в системах IoT. Мікроконтролери серії ESP відрізняються від інших доступною ціною, компактними розмірами та вбудованою можливістю підключення до Wi-Fi, що робить їх ідеальним вибором для смарт проєктів. ESP8266 дозволяє швидко створювати прототипи розумних пристроїв, поєднуючи високошвидкісний Wi-Fi з можливостями MQTT для надійного обміну повідомленнями. Крім того, завдяки відкритій архітектурі, мікроконтролер забезпечує локальну обробку даних, що допомагає зменшити затримки та знизити навантаження на централізовані сервери. Об'єднання ESP8266 в IoT-системи дозволяє створювати гібридні мережі, де потужність Wi-Fi забезпечує високошвидкісний зв'язок, а простота використання протоколів, таких як MQTT, сприяє ефективному управлінню інформаційними потоками. Такі рішення знаходять застосування в розумних містах, системах автоматизації домашнього господарства, промисловому моніторингу та інших сферах, де важливі оперативність і точність передачі даних.

1.5 Інтеграція Smart-систем з хмарними платформами

Інтеграція смарт-систем з хмарними платформами є складним процесом, що дозволяє з'єднати розумні пристрої та сенсори з централізованою хмарною інфраструктурою. Це створює динамічну екосистему для збору, обробки та аналізу даних у реальному часі. Інтеграція реалізується за допомогою надійних комунікаційних протоколів, таких як MQTT, HTTP або CoAP, які забезпечують швидку та стабільну передачу даних від пристроїв до хмарних серверів. Особливе місце займають API, що дозволяють різним компонентам системи без перешкод обмінюватися структурованою інформацією. Крім того, контейнеризація та мікросервісна архітектура сприяють оперативному масштабуванню та гнучкому розподілу обчислювальних ресурсів.

Розберемо кожен протокол і визначимо які є плюси їх використання.

Протокол MQTT відзначається своєю простотою та низькими витратами, що робить його ідеальним для пристроїв з обмеженими ресурсами. Завдяки моделі публікації та підписки, система може ефективно масштабуватися і забезпечувати надійну доставку повідомлень, навіть за умов нестабільного інтернет-з'єднання.

HTTP є загальноприйнятим стандартом для веб-комунікацій. Він забезпечує зручну інтеграцію різних сервісів завдяки підтримці RESTful API, що гарантує високу сумісність з наявними системами. Завдяки своїй простоті, HTTP стає незамінним інструментом для обміну даними між веб-додатками, серверами та іншими компонентами сучасних мереж.

Протокол CoAP розроблений спеціально для пристроїв з обмеженими ресурсами. Використання UDP замість TCP суттєво знижує накладні витрати, що сприяє швидшому обміну даними. При цьому система залишається сумісною з REST-підходом, а впровадження DTLS забезпечує додатковий рівень безпеки, відповідаючи високим вимогам до енергоефективності.

Переваги хмарних інтеграцій у смарт-системи є надзвичайно різноманітними. По-перше, масштабованість хмарних платформ дає можливість організаціям збільшувати обсяги оброблюваної інформації без потреби в значних інвестиціях у фізичне обладнання. Це дозволяє без труднощів додавати нові пристрої та розширювати мережу смарт-систем, навіть у разі швидкого зростання обсягу даних. По-друге, використання централізованої обробки дозволяє отримувати актуальні дані в режимі онлайн, що сприяє швидкому прийняттю управлінських рішень та прогнозуванню можливих недоліків або аварійних ситуацій у системі. По-третє, безпека є однією з основних переваг інтеграції. Використання сучасних методів шифрування, багаторівневої аутентифікації та постійного моніторингу забезпечує високий рівень захисту даних. Навіть у разі локальних збоїв інформація автоматично синхронізується з хмарними сервісами, що гарантує безперервність роботи в умовах непередбачуваних аварій. Крім того, ефективна централізація даних дозволяє

створювати системи резервного копіювання та відновлення для критично важливих процесів.

Яскравий приклад використання такої системи у сфері розумних міст мережа сенсорів розташовується по всьому місту для моніторингу дорожнього руху, рівня шуму, якості повітря та енергоспоживання. Дані з цих пристроїв передаються в хмарне середовище, де за допомогою алгоритмів аналізу інформації в реальному часі розробляються сценарії для оптимізації роботи світлофорів, планування транспортних маршрутів та швидкого реагування на надзвичайні ситуації. Цей підхід сприяє зменшенню заторів, покращенню екологічних показників і підвищенню якості життя мешканців міста. У промисловій сфері інтеграція смарт-систем з хмарними платформами сприяє автоматизації виробничих процесів. Сенсори, встановлені на обладнанні, постійно відстежують ключові параметри, такі як температура, вібрації та швидкість роботи машин. Зібрані дані передаються на хмарні сервери, де їх аналізують за допомогою штучного інтелекту. Виявлення відхилень у реальному часі дозволяє своєчасно виявляти потенційні несправності, прогнозувати потребу в технічному обслуговуванні та зменшувати простої виробництва. Це, в свою чергу, підвищує ефективність праці, знижує витрати та забезпечує стабільність виробничого процесу.

Інтегровані рішення активно застосовуються в системах управління будівлями. Сучасні житлові та комерційні об’єкти оснащуються сенсорами, які контролюють температуру, вологість, освітленість і безпеку. Дані з цих пристроїв передаються на хмарні платформи, де їх аналіз дозволяє автоматизувати регулювання кондиціонерів, систем опалення та вентиляції. Це сприяє оптимізації енергоспоживання, підтримці комфорту для мешканців будинку і забезпеченню швидкої реакції на потенційні загрози, такі як пожежі або несанкціонований доступ.

Інтеграція смарт-систем з хмарними платформами відкриває численні можливості, проте також стикається з певними викликами. Найважливішими з них є питання кібербезпеки, сумісності різних систем та забезпечення високої

якості зв'язку в умовах зростаючого обсягу даних. Додатково, необхідність постійних інвестицій у розвиток інфраструктури, навчання персоналу та впровадження новітніх технологій створює додаткові труднощі для багатьох організацій.

Проте безперервний розвиток технологій штучного інтелекту і використання нових стандартів безпеки створюють можливості для ще більш ефективної та комплексної інтеграції смарт-систем з хмарними платформами. Це сприятиме формуванню адаптивного, стійкого, інноваційно та технічного середовища, яке відповідатиме майбутнім потребам суспільства.

Інтеграція смарт-систем з хмарними платформами є важливим чинником модернізації сучасних технологічних рішень у різних сферах, таких як урбаністика, промисловість і медицина. Завдяки своїй масштабованості, високій ефективності управління даними та безпеці, інтегровані системи значно покращують якість життя. Проте виклики, пов'язані з технічною сумісністю, кібербезпекою та інвестиційними витратами, спонукають до розвитку нових технологічних рішень і сприяють інноваційному прогресу в майбутньому.

РОЗДІЛ 2
АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ СМАРТ-СИСТЕМИ

2.1 Вибір мікроконтролера для апаратної реалізації

Вибір мікроконтролера для реалізації розробки смарт-системи є важливим етапом, оскільки мікроконтролер в смарт-системах є ключовим фактором в роботі пристрою. Серед численних доступних платформ таких як, ESP8266-01, Raspberry Pi Pico та ESP32. ESP8266-01 виявився найкращим вибором завдяки своїм компактним розмірам, низькому споживанню енергії та вбудованим можливостям бездротового зв'язку через Wi-Fi. Це дозволяє пристрою швидко підключатися до серверів, хмарних платформ або локальних мереж, забезпечуючи обмін даними в реальному часі та можливість віддаленого управління.

Щодо програмування ESP8266-01 то з інтеграцією з Arduino IDE мікроконтролер забезпечує швидку роботу створення, компіляцію та завантаження програмного забезпечення, що суттєво спрощує процес розробки та налагодження. Велика кількість готових бібліотек полегшують написання коду та адаптацію системи до конкретних завдань, а також забезпечують просте підключення необхідних периферійних пристроїв, таких як дисплеї та сенсори.

Цей аспект особливо цінний для створення смарт-систем, де потрібна точність і швидке відображення інформації.

ESP8266-01 має кілька важливих переваг проти конкурентів, ці переваги роблять його ідеальним вибором для смарт-системи. Цей мікроконтролер відзначається високою енергоефективністю, що забезпечує тривалу стабільну роботу пристрою без перерв і тим самим хорошою інтеграцією з акумуляторами. Однією з його основних особливостей є вбудований Wi-Fi модуль, який дозволяє швидко передавати дані та здійснювати віддалене керування системою як через телефон так і через сайти.

Інтеграція вбудованого Wi-Fi розширює функціональність пристрою. Це дає пристрою можливості в режимі реального часу отримувати та відправляти

дані, здійснювати дистанційне оновлення програмного забезпечення, а також інтегрувати систему в мережеві рішення та IoT.

Ця особливість усуває потребу в підключенні зовнішніх модулів для бездротової комунікації, зменшуючи ризик апаратних збоїв і роблячи пристрій компактним.

Великим плюсом в використанні ESP8266-01 є те що він дозволяє підключати різноманітні периферійні пристрої, такі як дисплеї, сенсори, датчики та інші модулі, завдяки наявним великим вибором бібліотек. Це спрощує розширення функціональності системи та інтеграцію нових можливостей без необхідності в додаткових компонентах та робить систему стабільнішою і зменшує ризик помилок в роботі.

ESP8266-01 вирізняється своєю низькою ціною, що робить його економічно вигідним вибором. Поєднуючи доступну ціну з широкою доступністю на ринку. Це особливо важливо тому, що є оптимальний баланс між високою функціональністю та мінімальними витратами. Завдяки своїй конкурентоспроможній ціні, ESP8266-01 дозволяє створювати ефективні та стабільні рішення, навіть з обмеженим бюджетом. У таблиці 2.1 наведено зіставлення найбільш поширених мікроконтролерів та їх основні характеристики.

Таблиця 2.1 – Порівняння мікроконтролерів для реалізації апаратної частини [16]

Параметр	ESP8266-01	ESP32	Raspberry Pi Pico	Arduino Uno
Процесор	Tensilica L106	Dual-core Tensilica Xtensa LX6	RP2040	ATmega328P
Пам’ять	1МБ	4 МБ	2 МБ	32 КБ
Кількість GPIO	2	34	26	14
Підтримка Wi–Fi	Так	Так	Ні	Ні

Продовження таблиці 2.1

Параметр	ESP8266-01	ESP32	Raspberry Pi Pico	Arduino Uno
Енергоспоживання	Низьке	Середнє	Низьке	Середнє
Вартість	Низька	Низька	Середня	Висока

2.2 Вибір та обґрунтування компонентів для апаратної реалізації

У розробці смарт-системи важливо знати не лише функціональні можливості пристрою, а також його енергоефективність, масштабованість та економічність впровадження. Успішна апаратна реалізація базується на ретельно обраних компонентах, які дозволяють зменшити витрати ресурсів, забезпечуючи при цьому високу продуктивність і надійність. Серед компонентів значиме місце займає мікроконтролер з інтегрованим бездротовим модулем зв'язку ESP8266-01. Зображення Зовнішнього вигляду ESP8266-01 наведені на рисунку 2.1.

Схема розміщення виводів, характеристика мікроконтролера та схема розташування компонентів зображені на рисунку 2.2, 2.3 та 2.4.

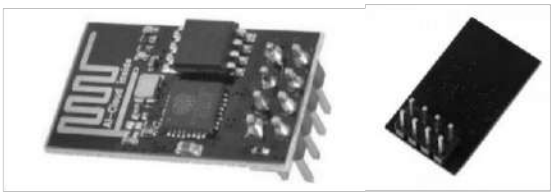


Рисунок 2.1 – Мікроконтролер ESP8266-01 вигляд зверху та знизу [17]

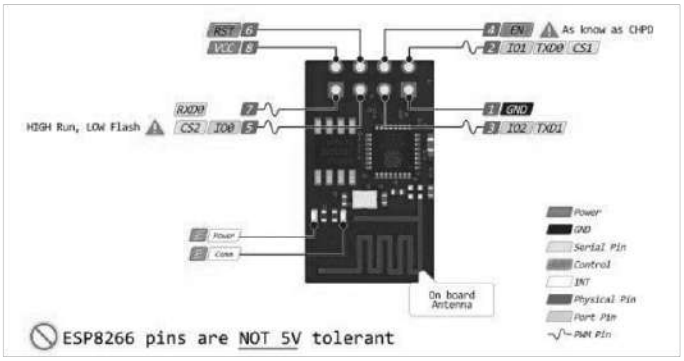
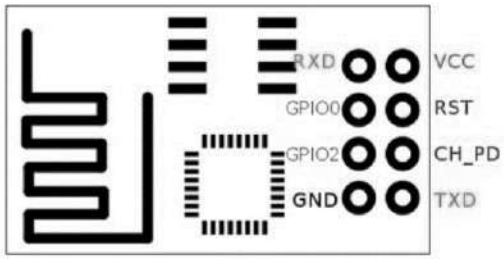


Рисунок 2.2 – Мікроконтролер ESP8266-01 схема розміщення виводів [17]



Label	Signal
VCC	3.3V (3.6V max) supply voltage
GND	Ground
TXD	Transmit Data (3.3V level)
RXD	Receive Data (3.3V level!)
CH_PD	Chip Power down: (LOW = power down active)
GPIO0	General Purpose I / O 0
GPIO2	General Purpose I / O 2
RST	Reset (reset = LOW active)

Рисунок 2.3 – Мікроконтролер ESP8266-01 характеристика [17]

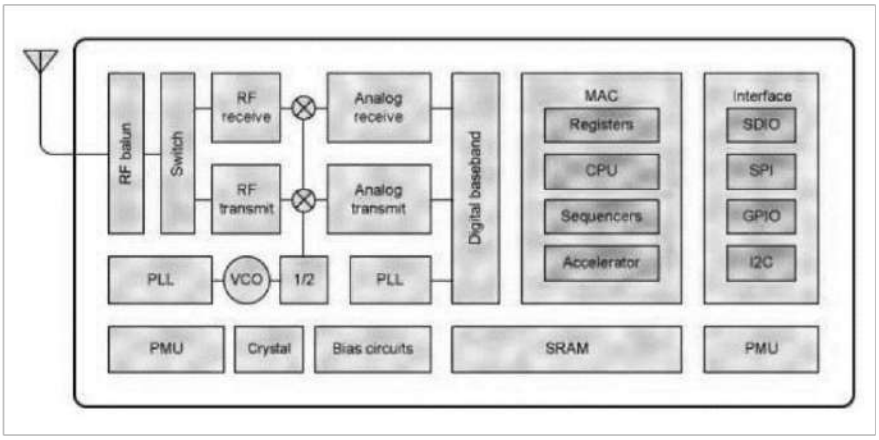


Рисунок 2.4 – Мікроконтролер ESP8266-01 розташування компонентів [17]

Перш за все ESP8266-01 не просто мікроконтролер, а універсальна платформа, що відкриває безмежні можливості у створенні інноваційних IoT-рішень. Конструкція цього мікроконтролера об’єднує у собі всі необхідні модулі для обробки даних та зв’язку, що дозволяє зосередитися на творчих завданнях, не витрачаючи час на вирішення інтеграційних проблем таких як модуль Wifi, які часто виникають при використанні окремих компонентів.

Також ESP8266-01 не лише виконує складні обчислення, а й забезпечує ефективну комунікацію між пристроями, що є ключовим аспектом для схожих розумних пристроїв. Завдяки підтримці стандартів IEEE 802.11 b/g/n, ця

платформа відкриває нові можливості для створення динамічних мереж, де кожен пристрій може одночасно виконувати функції як отримувача, так і передавача даних. Така гнучкість у мікроконтролері дозволяє реалізовувати рішення, які адаптуються до потреб різних застосувань – від домашньої автоматизації до великих промислових систем, де безперервність зв'язку є критично важливою.

Універсальність ESP8266-01 дозволяє використовувати його як модуль, який легко інтегрується в будь-яку систему, компактні розміри пристрою забезпечують високу продуктивність навіть в умовах обмеженого простору або ресурсів. Здатність працювати з мінімальним споживанням енергії та можливість функціонування в різних режимах – від повномасштабних обчислень до активної комунікації – роблять його ідеальним вибором для любих проектів.

Варто зазначити що за свою ціну ESP8266-01, пропонує високий рівень функціональності та надійності. Це робить його вигідним вибором як для прототипування, так і для масового виробництва. Використання ESP8266-01 підтверджує, що це ефективна стратегія, яка забезпечує стабільну роботу системи з мінімальними ресурсами, при цьому підтримуючи високі стандарти якості та інноваційності в сфері IoT-технологій.

2.3 Вибір акумулятора

Щоб, забезпечити максимальну портативність пристрою в смарт-системі, було прийнято рішення поставити невеликий акумулятор Li-Ion 18650 ємністю 1500 mAh. який забезпечує ідеальний баланс між компактністю пристрою та його автономністю.

Таке рішення обумовлено тим, що така ємність гарантує ідеальний баланс між компактністю пристрою та його автономністю.

Завдяки високій енергетичній щільності, акумулятор здатен забезпечувати достатнє живлення навіть у невеликих розмірах.

Використання цього акумулятора, який працює в парі з функціями енергозбереження, системі вдається працювати стабільно навіть у режимах з підвищеним навантаженням.

Стабільне постачання напруги забезпечує безперебійну роботу мікроконтролеру та інших елементів, що є важливим для підтримки якості та ефективності функціонування всієї смарт-системи. Крім того, компактний акумулятор дозволяє зручно розміщувати всі компоненти пристрою, що покращує загальну ергономіку та мобільність смарт-системи.

Можливість використання режимів сну та оптимізації процесів обробки даних дозволяє зменшити споживання енергії до мінімуму і зробити максимальну тривалу роботу смарт-системи. Акумулятор Li-Ion 18650 зображений на рисунку 2.5.

Рисунок 2.5 – Акумулятор Li-Ion [18]

2.4 Вибір дисплею

Після детального аналізу доступних технологій дисплеїв таких як IPS, LCD та OLED дисплеїв було прийнято рішення зупинитися саме на технології OLED. У процесі вибору враховувалися низка ключових критеріїв, зокрема висока якість зображення, енергоефективність, компактність та сумісність.

Найважливішою характеристикою технології OLED є те, що кожен піксель функціонує як окреме джерело світла. Це забезпечує вражаючу контрастність, дозволяючи отримувати глибокі чорні відтінки поряд із насиченими кольорами, що робить таке зображення яскравим і динамічним порівнюючи з IPS матрицями. Відсутність окремої підсвітки, властивої традиційним LCD-матрицям, значно знижує енергоспоживання яке максимально важливе в смарт-системі коли потрібна максимально довга робота акумулятора, кожен піксель активується лише за потреби, створюючи ефект «розумного» освітлення екрану та сприяючи економії заряду батареї.

Компактний OLED-дисплей ідеально підходить для смарт-системи в якій потрібна компактність і хороша видимість, такий дисплей ідеально підходить для обмеженого простору. Його тонка конструкція та мінімалістичний дизайн роблять цю технологію надзвичайно привабливою для компактних пристроїв. Адаптивність цієї технології дозволяє безперешкодно інтегрувати її з іншими компонентами, створюючи системи, в яких кожен елемент працює в гармонії, забезпечуючи бездоганний користувацький досвід.

В смарт-системі використовується OLED-дисплей з роздільною здатністю 128×64 пікселів який зображений на рисунку 2.6 він поєднує високу якість зображення з енергоефективністю. Такий екран гарантує чітке та яскраве відображення інформації. В таблиці 2.2 описані характеристики дисплею.

Рисунок 2.6 – OLED Дисплей 128x64 [19]

Таблиця 2.2 – Характеристика OLED дисплею [20]

Тип дисплея	OLED
Контролер	SSD1306
Дозвіл	128x64
Діагональ дисплея	0,96"
Інтерфейс	IIC (I2C)
робоча напруга	3.3-5 В
Розміри	27x27 мм
Роз'єм	4-pin
Товщина	11 мм
Ширина	27 мм
Висота	27 мм

2.5 Контролер заряду

Найкращим варіантом для безпечного та ефективного заряджання акумулятора Li-Ion 18650 в проєкті є зарядний модуль TP4056 Mini-USB з функцією захисту акумулятора.

Цей модуль виявився найкращим вибором серед інших модулів, він вирізняється інноваційним принципом роботи: спочатку він функціонує за алгоритмом постійного струму, а після досягнення заданого рівня переходить у режим постійної напруги. Така комбінація режимів забезпечує оптимальне живлення акумулятора, значно підвищуючи його ефективність і продовжуючи термін служби.

Його ефективні функції зображені на рисунку 2.7

Захист від перезарядки: Автоматичне обмеження струму зупиняє заряджання після досягнення встановленої напруги, що запобігає перегріванню та пошкодженню акумулятора.

Захист від надмірного розрядження: Механізм відключення дозволяє уникнути ситуацій, коли акумулятор розряджається нижче критично допустимого рівня, зберігаючи його життєздатність.

Захист від короткого замикання і перевантаження: Інтелектуальна система моніторингу стежить за електричними параметрами, що допомагає уникнути небезпечних ситуацій при ненормальних схемах підключення або збоїв у роботі.

Індикатор стану заряджання: Багато варіантів модуля оснащені світловими індикаторами, які інформують користувача про поточний стан заряджання та діагностують роботу пристрою.

Рисунок 2.7 – Основні функції роботи модуля TP4056 [21]

Зарядний модуль TP4056 дозволяє використовувати стандартний 5В струм з Mini-USB роз'єму, що забезпечує його сумісність з майже всіма джерелами живлення, такими як адаптери, Power Bank або USB-порти комп'ютера. Завдяки своїй простій структурі він легко вписується в смарт-системи, забезпечуючи безпечний процес зарядки та запобігаючи можливим аварійним ситуаціям завдяки вбудованим захисним механізмам. Зарядний модуль TP4056 зображений на рисунку 2.8.

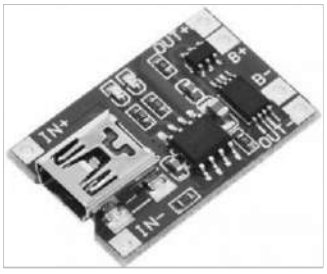


Рисунок 2.8 – Зарядний модуль TP4056 Mini-USB з функцією захисту акумулятора [22]

2.6 Схемотехніка: розробка схеми підключення компонентів

Для реалізації прототипу смарт-системи відображення інформації було перебрано перелік компонентів та обрано найоптимальніші варіанти.

В якості мікроконтролеру системи було обрано ESP8266-01 який є найкращим вибором між своїх конкурентів. Його плюси це невелика ціна, маленькі розміри та вбудований Wi- fi модуль.

Для відображення інформації було обрано OLED дисплей 128x64. Його плюсами є чітке зображення та невелика ціна.

Для автономності системи було обрано акумулятор Li-Ion 18650. Тому що для переносимості пристрою потрібен невеликий але місткий акумулятор.

Для реалізації віртуального стенду була використана програма Fritzing. В Fritzing були використані макетна плата та набір з'єднувальних проводів типу тато-тато Готовий тестовий стенд зображений на рисунку 2.9. Для створення стенду були використанні такі компоненти:

- ESP8266-01;
- дисплей 128x64;
- акумулятор;
- кнопки SMD.

Компоненти підключаються до ESP8266-01 наступним чином:

- червоний провід підключається до макетної дошки;
- жовтий провід підключається до дисплею;
- фіолетовий провід підключається до кнопки;
- сірий провід підключається до кнопки;
- чорний провід підключається до макетної дошки;
- зелений провід пікдлючається до дисплею.

Компоненти які підключаються до дисплею:

- жовтий провід пікдлючається до ESP8266-01;
- зелений провід пікдлючається до ESP8266-01;

– червоний провід пікдлючається до макетної дошки;

– чорний провід підключається до макетної дошки;

Компоненти які підключаються до акумулятора:

– чорний провід підключається до макетної дошки;

– червоний провід підключається до макетної дошки.

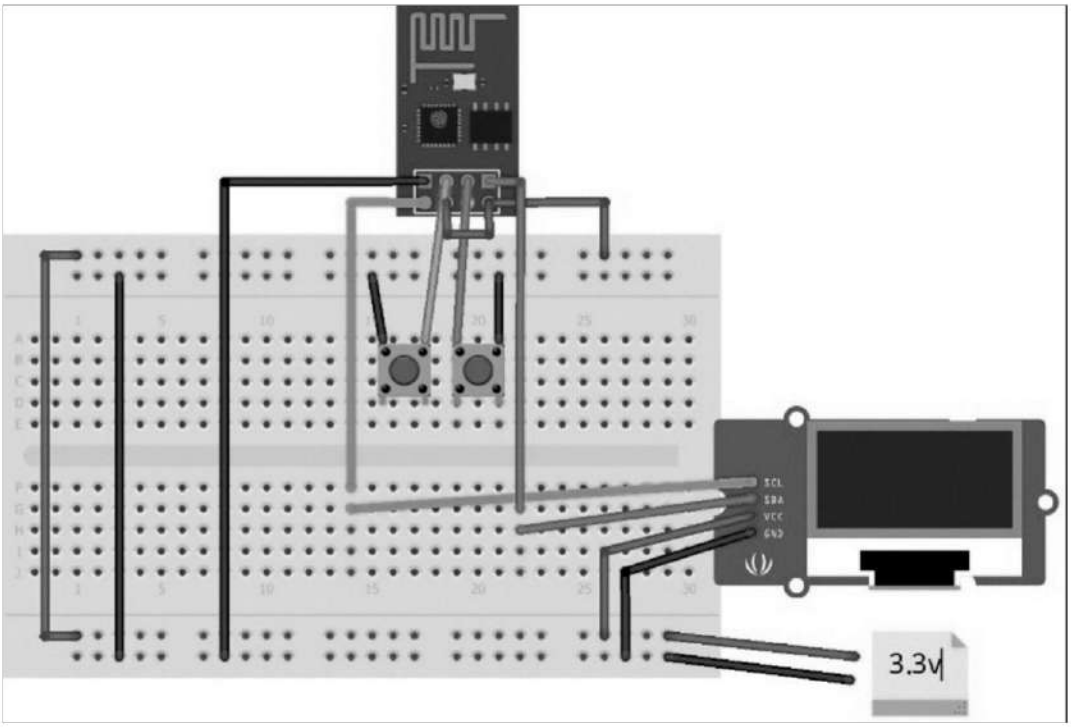


Рисунок 2.9 – Схема підключення компонентів в програмі frizzing

В зібраній системі всі частини були спаяні та вкладені в корпус, живлення системи відбувається через зарядний модуль TP4056.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СМАРТ-СИСТЕМИ

3.1 Розробка прототипу апаратної частини

Розробка прототипу апаратної частини смарт-системи є дуже важливим розділом для перевірки працездатності вибраних компонентів системи та валідації схемотехнічних рішень. На етапі розробки здійснюється фізичне збирання компонентів відповідно до розробленої схеми підключення, тестується взаємодія між модулями та налагоджуються основні функції пристрою.

Основними елементами смарт-системи відображення інформації є мікроконтролер серії ESP8266-01 він відповідає за збір даних, виконання алгоритмів управління та забезпечення комунікації із зовнішніми пристроями, акумулятор Lipo джерело живлення для системи, що забезпечує автономну роботу для портативності пристрою, OLED дисплей для чіткого виводу інформації, та зарядний модуль TP4056 він забезпечує безпечне та стабільне заряджання акумулятора. Тестування та з'єднання всіх компонентів здійснюється через макетну плату та з'єднувальних проводів. Таке рішення дозволяє швидко впевнитись чи всі компоненти пристрою готові до збору в корпус. В таблиці 3.1 написані всі компоненти які будуть використовуватись для прототипу смарт-системи.

Таблиця 3.1 – Всі компоненти які використовується для прототипу

Компонент	Призначення
ESP8266-01	Мікроконтролер, забезпечує обробку даних, реалізацію алгоритмів та бездротову комунікацію
Акумулятор 1500 mAh	Джерело автономного живлення для роботи системи
OLED дисплей 128×64	Відображення інформації в режимі реального часу
Зарядний модуль TP4056	Забезпечує ефективне та безпечне заряджання акумулятора за допомогою алгоритмів контролю над струмом та напругою

Продовження таблиці 3.1

Компонент	Призначення
Макетна плата (breadboard)	Використовується для швидкого з'єднання всіх компонентів системи
З'єднувальні проводи	Підключення компонентів до ESP8266-01
Кнопки	Кнопки вкл/викл та для переміщення.

Після виготовлення фізичного прототипу смарт-системи проводиться тестування кожного компонента. Перевіряється функціональність ESP8266-01 при встановленні бездротового з'єднання, точність відображення інформації на OLED-дисплеї, а також ефективність роботи зарядного модуля TP4056 в комбінації з акумулятором ємністю 1500 mAh.

Після успішного тестування всіх компонентів, відбувається тестування готового прототипу в реальних умовах.

3.2 Програмування мікроконтролера ESP: середовище та бібліотеки

Програмування мікроконтролера це надзвичайно важливий розділ в проектуванні та створенні смарт-системи відображення інформації, для реалізації програмного коду буде використовуватись середовище програмування Arduino IDE.

Програмне середовище Arduino IDE використовується завдяки зрозумілій структурі та широкому спектру можливостей є підтримка численних бібліотек, прикладів коду та модулів. Зручний графічний інтерфейс дозволяє швидко компілювати програмний код його завантаження на плату відбувається майже миттєво і без жодних труднощів, що суттєво скорочує час, необхідний для встановлення та тестування.

Програмування смарт–годинника виконується в середовищі Arduino IDE яке зображене на рисунку 3.1

Рисунок 3.1 – Середовище розробки ARDUINO IDE

Однією з головних переваг Arduino IDE є його вбудована система управління бібліотеками. Менеджер бібліотек дозволяє швидко знаходити, встановлювати та оновлювати потрібні модулі, що спрощує інтеграцію сторонніх компонентів і забезпечує сумісність з актуальними версіями програмного забезпечення.

Спрощений синтаксис, заснований на C/C++, робить цю платформу зручною як для новачків, так і для розробників, які давно вже використовують цю програму. Завдяки зрозумілому інтерфейсу початківці швидко освоюють основи програмування мікроконтролерів, тоді як професіонали можуть ефективно реалізовувати складні алгоритми управління пристроєм.

Програмне забезпечення системи відображення інформації створено за модульним принципом, цей принцип дозволяє розділити функціонал на кілька окремих блоків. Перший блок відповідає за те як проходить ініціалізація апаратних компонентів: налаштовується OLED–дисплей для відображення інформації, апаратні кнопки для управління режимами, а також система моніторингу заряду акумулятора, яка реалізується за допомогою контролера заряду та вбудованої функції зчитування аналогових значень. Такий підхід забезпечує стабільну роботу кожного компонента системи, що відповідає за

свою частину логіки – від відображення часу до реагування на натискання кнопок.

Використання менеджера бібліотек в Arduino IDE має значні переваги. Цей інструмент дозволяє швидко встановлювати та оновлювати необхідні модулі, що знижує ризик технічних помилок, пов'язаних із неправильною інсталяцією. Регулярне оновлення бібліотек не лише підвищує продуктивність програми, але й забезпечує сумісність із новими версіями середовища розробки. Цей критерій є важливим для інтеграції з Google API для отримання додаткових сервісів.

Для реалізації всіх зазначених задач використовуються спеціалізовані бібліотеки які наведені у таблиці 3.2.

Таблиця 3.2 – Бібліотеки Arduino IDE які використовуються

Назва бібліотеки	Призначення
Adafruit_SSD1306	бібліотека для роботи з OLED-дисплеєм
Time.h	бібліотека для синхронізації часу
Wire.h	функції для спрощення роботи з I^2C
ESP8266WiFi.h	надає функціонал для роботи з Wi-Fi
DNSServer.h	дозволяє мікроконтролеру функціонувати як DNS сервер.
ESP8266WebServer.h	функціонування як веб-сервер.
WiFiManager.h	використовуєтья для того щоб спростити підключення пристрою до Wi-Fi мережі.
WiFiManager.h	використовуєтья для того щоб спростити підключення пристрою до Wi-Fi мережі.
ESP8266WiFi.h	надає функціонал для роботи з Wi-Fi
WiFiClientSecure.h	основна функція це встановлювати захищені з'єднання через TLS/SSL розширює можливості ESP8266WiFi.h

Продовження таблиці 3.2

Назва бібліотеки	Призначення
JsonStreamingParser.h	використовується для обробки великих JSON-об'єктів
Adafruit_GFX.h	є графічною бібліотекою для дисплеїв
logo.h	Використовується для зображення ярликів на екрані

3.3 Налаштування зв'язку з мобільним додатком

Для налаштування зв'язку зі смарт-системою використовується мобільний додаток, MIT App Inventor він використовується як інструмент для створення мобільних додатків за допомогою візуального програмування, з інтеграцією Google API для отримання актуальних сповіщень з погодою. Головна мета полягає в передачі інформації.

MIT App Inventor зображений на рисунку 3.2

Рисунок 3.2 – Мобільний додаток смарт-системи

При розробці мобільного додатку особливу увагу дають стандартним компонентам App Inventor, доповненим спеціальними розширеннями для інтеграції з Google API. Додаток отримує доступ до необхідних даних через авторизоване з'єднання до мережі Wi-Fi, використовуючи сучасні протоколи безпеки, такі як OAuth 2.0. Це гарантує захист інформаційного обміну між сервером і додатком.

Передача сповіщень від мобільного додатку до пристрою здійснюється за допомогою захищеного стандартизованого протоколу зв'язку. В смарт-системі використовується мікроконтролер ESP-01 з вбудованим модулем за допомогою якого, пристрій може підключатися через Wi-Fi і отримувати необхідні дані в режимі реального часу, використовуючи хмарні сервіси Google.

Використання MIT App Inventor дозволяє швидко створювати інтуїтивно зрозумілі інтерфейси завдяки підтримці у візуальному програмуванню. У цьому середовищі кожен елемент, від налаштувань сповіщень до обробки подій, інтегрований з логікою роботи смарт-системи. Це значно прискорює процес розробки, забезпечує гнучкість у налаштуваннях і дозволяє адаптувати функціональність під конкретні потреби користувачів. Крім того, можливість оперативного тестування та внесення змін сприяє швидкому переходу від прототипу до готового продукту без зайвих ускладнень.

3.4 Реалізація алгоритмів збору та відображення інформації

В етапі збору використовуються мікроконтролер ESP8266-01, USB перехідник для програмування і налагодження для мікроконтролера ESP8266-01, OLED дисплей 0.96 128x64 для відображенні інформації, Акумулятор Li-Ion 18650 на 1500 Mah для портативної роботи смарт-системи, Зарядний модуль TP4056 Mini-USB для заряджання акумулятора, Кнопки типу SMD для включення виключення системи та переходу між екранами, та програматор FT232RL для програмування інших компонентів системи.

Мікроконтролер ESP8266-01 має 8 виходів, а саме:

– VCC: живлення напругою від 2,5 до 3,6 В;
– GND: «земля»;
– TXD: передача даних (UART);
– RXD: прийом даних (UART);
– CH_PD: переводить модуль в енергозберігаючий режим;
– GPIO0: дозволяє перевести пристрій у режим програмування;
– GPIO2: порт введення та виведення;
– RST: апаратне скидання.

Після написання і компілювання коду в Arduino IDE вводимо необхідні дані для того щоб мікроконтролер підключився до мережі Wi-Fi, завантажуємо його на мікроконтролер через перехідник, бачимо що загорівся червоний індикатор значить що все працює правильно,

Для програмування мікроконтроллера ESP8266-01 використовуємо програматор FT232RL, який зображений на фото 3.2.

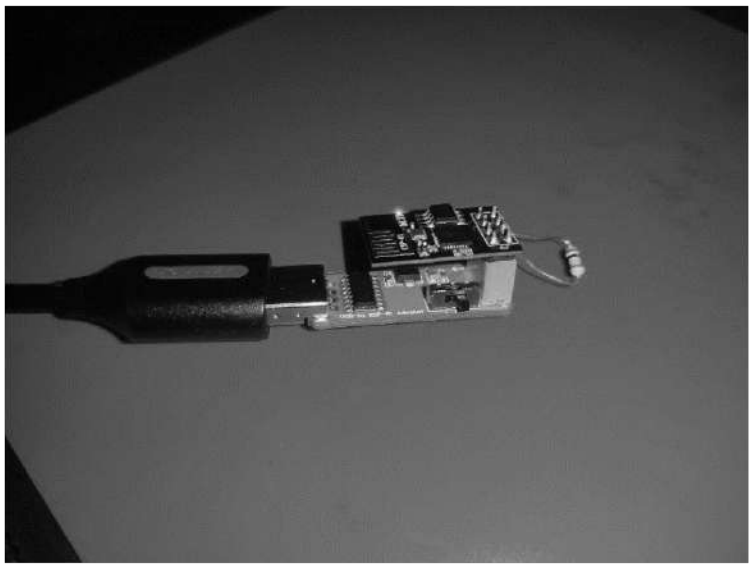


Рисунок 3.2 – ESP8266-01 з програматором FT232RL

Після програмування, для швидкого підключення використовуємо макетну плату, під'єднуємо до контактів плати ESP8266 (мінус -> GND), (плюс -> 3v3, EN).

Дисплей використовуємо SSD1306, до нього також під'єднуємо (мінус -> GND), (плюс -> VCC), а також контакти для передачі інформації з контролера (SCL -> RX) та (SDA -> TX).

Дві кнопки під'єднуємо до контактів IO0 та IO2. Макетний вигляд під'єднаних елементів зображений на фото 3.3

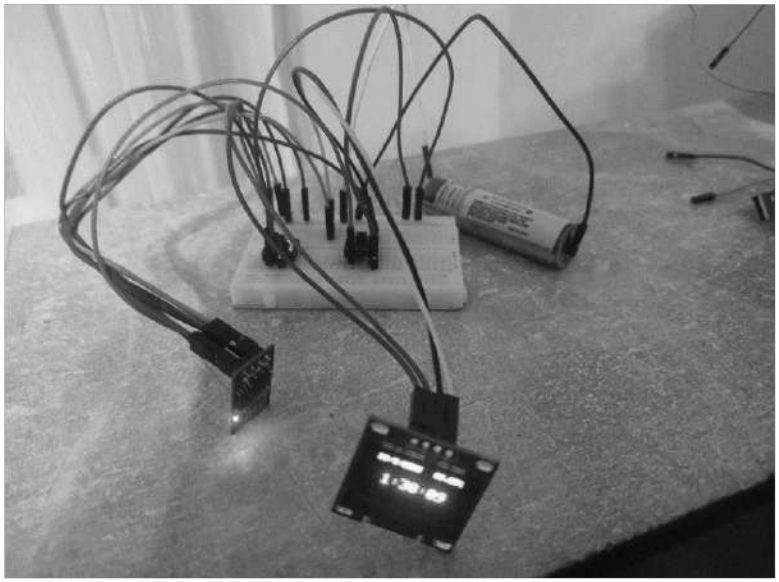


Рисунок 3.3 – Макет Смарт-системи

3.5 Тестування роботи смарт-системи

Після важливого етапу розробки апаратної частини та не менш важливого етапу створення програмного забезпечення наступим важливим кроком є тестування готової смарт-системи відображення інформації. Цей процес є необхідним для перевірки роботи пристрою, виявленням недоліків та оптимізації роботи системи для досягнення максимального комфорту при роботі з пристроєм.

Для тестування смарт-системи проводились необхідні перевірки роботи кнопок, екрану та акумулятора. Після успішного проведення цих компонентів почався етап повного тестування пристрою почався він кнопкою вмикання пристрою, після першого вмикання пристрій напише що створив точку доступу для того щоб ми змогли з'єднати пристрій з нашим телефоном, знаходимо її

через телефон в параметрах, Wi-Fi називається «ESP Watch» підключаємся до неї і вводим пароль до точки доступу Wi-Fi, після успішного з'єднання часи запам'ятають пристрій.

Головним екраном після включення буде час/дата та цифри які показують наскільки хороше з'єднання з Wi-Fi, наступними вікнами є вікно погоди на сьогоднішній день, в цьому вікні показуються температура, хмарність, вологість, атмосферний тиск та швидкість вітру, в наступних двох вікнах показується погодна на наступні два дні. Вікно 5 показує який зараз курс гривні до долара та євро. 6 вікно відповідає за IP-пристрою це вікно потрібно для з'єднання пристрою з смарт-системою при втраті з'єднання з Wi-Fi пристрій буде підтримувати дані та оновлювати їх з телефону.

Загалом, розроблена смарт-система відображення інформації показала свою функціональність, надійність та відповідність технічним вимогам, які були встановлені на початковому етапі проєкту. Додаткові можливості для оптимізації, такі як впровадження енергозберігаючого режиму та створення більшого функціоналу для роботи, створюють перспективи для подальшого вдосконалення.

ВИСНОВКИ

У процесі розробки було реалізовано прототип смарт-системи відображення інформації на базі мікроконтролера ESP. Смарт-система – це сучасна інтегрована платформа, що об'єднує фізичні пристрої з цифровими процесами, забезпечуючи швидку та точну передачу даних користувачу.

Під час створення прототипу смарт-системи було реалізовано інтеграцію мікроконтролера з іншими елементами, через розроблену електричну схему, яка враховує специфіку роботи всіх компонентів: мікроконтролера, дисплея, акумулятора та контролера заряду. Всі компоненти були з’єднанні через макетну плату. Це рішення дозволило швидко перевірити їхню взаємодію та налаштувати більш ефективну схему з'єднань в корпусі.

Розроблено алгоритм для збору інформації в режимі реального часу для для відображення результатів на пристрої, щляхом написання і компілювання коду в Arduino IDE, введенню необхідних даних для того щоб мікроконтролер підключився до мережі Wi-Fi.

Було досліджено методи оптимізації автономності системи пристрою для її портативності. Можливості мікроконтролера ESP8266-01, завдяки вбудованому Wi-Fi модулю, цей пристрій забезпечує можливості дистанційного керування, обміну даними в реальному часі. Інтеграція з Arduino IDE, дає широкий вибір бібліотек і високу енергоефективність та автономність пристрою.

Було запропоноване рішення для синхронізації роботи смарт-системи з мобільними пристроями шляхом створення мобільного додатку на платформі MIT App Inventor, що інтегрується з Google API. Через яку можна підключатись до пристрою і отримувати необхідні дані через неї.

Результати проведеного тестування в реальних умовах показали, що пристрій стабільно працює, помилок не виявлено, пристрій демонструє хорошу енергоефективність.

На основі виконаного дослідження можна зробити висновок, що розроблена смарт-система є ефективна і може бути використана для персонального використання, має хороші перспективи для подальшого розвитку у рамках рішень для Інтернету речей (IoT).

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Никитюк А, Поліщук Л. Інтеграція Smart-систем з хмарними платформами. Програмне та апаратне забезпечення в інформаційних технологіях: матеріали Міжнар. наук.-практ. конф. молодих вчених та студентів (6 травня 2025 р). / відп. ред. Т.В. Терлецький. Вип. 1. Луцьк: ЛНТУ, 2025. С. 134-136.

2. Смарт-технології як хобі: що це, види, компоненти. *Hobitera.* URL: https://hobitera.com/c-smart-tekhnologhiji (дата звернення: 18.02.2025).

3. Navigating the Smart City *Domains. Blogs.* URL: https://www.vvdntech.com/en-us/blog/navigating-the-smart-city-domains-a-comprehensive-guide/ (дата звернення: 19.02.2025).

4. Застосування мікроконтролерів *Distancionka.* URL: https://salo.li/0971396 (дата звернення: 19.02.2025).

5. Different Applications of Microcontroller *VLSIFacts.* URL: https://vlsifacts.com/different-applications-microcontroller/ (дата звернення: 25.02.2025).

6. Арифметико-логічний пристрій. *StudFiles.* URL: https://studfile.net/preview/7075014/page:29/ (дата звернення: 26.02.2025).

7. Microcontroller in the automotive sector. *Evision* URL: https://evision-webshop.eu/blog/microcontroller-in-the-automotive-sector (дата звернення: 26.02.2025).

8. Мікроконтролер ATMEGA. *Радіо Магазин.* URL: https://radio-magazin.com.ua/atmega8-16pu-mikrokontroller-8-bit-avr-16mgts-8kb-flash-dip-28(дата звернення: 28.02.2025).

9. MSP430 Texas Instruments *Mouser Electronic.* URL: https://salo.li/aDf99B0 (дата звернення: 28.02.2025).

10. ESP32-C3 Type-C. *Arduino в Україні.* URL: https://surl.li/phfxxc (дата звернення: 28.02.2025).

11. Посібник з мікроконтролерів ESP. *Technology Limited.* URL: https://ua.ariat-tech.com/blog/ESP32-Microcontroller-Guide.html (дата звернення: 05.03.2025).

12. ESP modules. *Espressif Systems.* URL: https://www.espressif.com/en/products/modules (дата звернення: 05.03.2025).

13. ESP 8266 vs ESP 32. *The IoT Academy Blogs.* URL: https://www.theiotacademy.co/blog/esp-8266-vs-esp-32/ (дата звернення: 12.03.2025).

14. BLE протокол –. *IT Master - електроніка та програмування.* URL: https://itmaster.biz.ua/directory/standarts/ble.html (дата звернення: 12.03.2025).

15. Solutions T. D. Wi-Fi, Bluetooth, and MQTT. *Embedded-System.* URL: https://embedded-systems.techdefencesolutions.com/2024/06/wi-fi-bluetooth-and-mqtt-essential.html (дата звернення: 12.03.2025).

16. Raspberry pi pico compare with arduino uno, esp32, esp8266. *MrElectroUino.* URL: https://surl.lu/jwwojh (дата звернення: 18.03.2025).

17. Wi-Fi модуль ESP8266 версія ESP-01. *Arduino в Україні.* URL: https://surl.li/jpmyxz (дата звернення: 19.03.2025).

18. Акумулятор 18650 1500 mAh. *Bestbattery* URL: https://bestbattery.com.ua/ua/li_ion_1850_ua/li_ion_18650_ua/18650_protected_ua/35e_protected_ua (дата звернення: 28.05.2025).

19. OLED дисплей 0.96 128x64. *Arduino в Україні.* URL: https://arduino.ua/prod2911-oled-displei-0-96-i2c-128x64-jelto-sinii-ssd1315 (дата звернення: 21.03.2025).

20. OLED дисплей 0.96 128x64. *Arduino в Україні.* URL: https://surl.li/snwjfe (дата звернення: 25.03.2025).

21. Зарядний модуль TP4056 Mini-USB. *Arduino в Україні.* URL: https://surl.li/ehcntf (дата звернення: 27.03.2025).

22. Зарядний модуль TP4056 Mini-USB. *Arduino в Україні.* URL: https://surl.lu/ggfrza (дата звернення: 27.03.2025).

ДОДАТКИ

Додаток А

Код пристрою

```
#include <Adafruit_SSD1306.h>
#include <ESP8266HTTPClient.h>
#include <Wire.h>
#include <WiFiClientSecure.h>
#include <ESP8266WiFi.h>
#include <DNSServer.h>
#include <ESP8266WebServer.h>
#include <WiFiManager.h>
#include <time.h>
#include <JsonStreamingParser.h>
#include <Adafruit_GFX.h>
#include <ArduinoJson.h>
#include "logo.h"

// Конфігурація API
const char* OPENWEATHER_API_KEY = "3762bcad39d27da54f738d1a0ce1046d";
const char* CITY_ID = "702569"; // Київ за замовчуванням
struct CurrencyRates {
  float usd;
  float eur;
};
CurrencyRates rates = {0, 0};
// Налаштування дисплею
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
// Іконки
const unsigned char sunny_icon[] PROGMEM = {
  0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0xC0, 0xC0, 0xC0, 0xC0, 0x80,
  0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};
const unsigned char cloudy_icon[] PROGMEM = {
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0xC0, 0xE0, 0xF0, 0xF8,
  0xF8, 0xFC, 0xFC, 0xFE, 0xFE, 0xFE, 0xFE, 0xFC, 0xFC, 0xF8, 0xF0, 0x00
};
const unsigned char rain_icon[] PROGMEM = {
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0xC0, 0xE0, 0xF0, 0xF8,
  0xF8, 0xFC, 0xFC, 0xFE, 0xFE, 0xFE, 0xFE, 0xFC, 0xFC, 0xF8, 0xF0, 0x00
};
const unsigned char snow_icon[] PROGMEM = {
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0xC0, 0xE0, 0xF0, 0xF8,
  0xF8, 0xFC, 0xFC, 0xFE, 0xFE, 0xFE, 0xFE, 0xFC, 0xFC, 0xF8, 0xF0, 0x00
};
// Структура для даних погоди
struct WeatherData {
  int temp;
  const unsigned char* icon;
  String condition;
  int humidity;
  int pressure;
  float windSpeed;
};
// Глобальні змінні
WiFiServer server(80);
WiFiClientSecure secureClient;
int timezone = 2 * 3600; // UTC+2 для України
int dst = 1; // Літній час
WeatherData currentWeather = {0, sunny_icon, "Loading...", 0, 0, 0};
```

```
WeatherData forecast[3] = {
  {0, sunny_icon, "Loading...", 0, 0, 0},
  {0, cloudy_icon, "Loading...", 0, 0, 0},
  {0, rain_icon, "Loading...", 0, 0, 0}
};
long api_lasttime = 0;
long clocklast = 0;
long weather_lasttime = 0;
long subs = 0;
long viewcount = 0;
long comments = 0;

String serorclient = "Client";
int showdisplay = 1;
int notif = 0;
int val = 0;

InstagramUserStats instaResponse1, instaResponse2;
bool apiCallSuccess = false;
// Функції
const unsigned char* getWeatherIcon(String condition) {
  condition.toLowerCase();
  if (condition.indexOf("clear") >= 0 || condition.indexOf("sunny") >= 0) {
    return sunny_icon;
  } else if (condition.indexOf("cloud") >= 0) {
    return cloudy_icon;
  } else if (condition.indexOf("rain") >= 0 || condition.indexOf("drizzle")
>= 0) {
    return rain_icon;
  } else if (condition.indexOf("snow") >= 0) {
    return snow_icon;
  }
  return cloudy_icon;
}
bool getCurrencyRates() {
  BearSSL::WiFiClientSecure client;
  client.setInsecure(); // Вимкнення перевірки SSL сертифіката

  if (!client.connect("api.privatbank.ua", 443)) {
    Serial.println("Connection to PrivatBank failed");
    return false;
  }

  String                    request                    =                    "GET
http://api.privatbank.ua/p24api/pubinfo?json&exchange&coursid=5
HTTP/1.1\r\n"
                 "Host: api.privatbank.ua\r\n"
                 "Connection: close\r\n\r\n";

  client.print(request);

  unsigned long timeout = millis();
  while (!client.available() && millis() - timeout < 10000) {
    delay(10);
  }

  if (!client.available()) {
    Serial.println("PrivatBank response timeout");
    client.stop();
    return false;
  }

  // Пропускаємо HTTP заголовки
```

```
  while (client.available()) {
    String line = client.readStringUntil('\n');
    if (line == "\r") break;
  }

  // Читаємо JSON тіло відповіді
  String payload = client.readString();
  client.stop();

  DynamicJsonDocument doc(512);
  DeserializationError error = deserializeJson(doc, payload);

  if (error) {
    Serial.print("JSON parsing failed: ");
    Serial.println(error.c_str());
    return false;
  }

  // Отримуємо курси валют
  for (JsonObject rate : doc.as<JsonArray>()) {
String currency = rate["ccy"].as<String>();
    if (currency == "USD") rates.usd = rate["buy"].as<float>();
    else if (currency == "EUR") rates.eur = rate["buy"].as<float>();
  }

  Serial.printf("Rates    updated:    USD=%.2f,    EUR=%.2f\n",    rates.usd,
rates.eur);
  return true;
}

bool fetchWeatherData() {
  WiFiClient client;
  String   url   =   "http://api.openweathermap.org/data/2.5/weather?id="   +
String(CITY_ID) +
               "&units=metric&appid=" + OPENWEATHER_API_KEY;

  if (!client.connect("api.openweathermap.org", 80)) {
    return false;
  display.clearDisplay();
  display.setTextSize(1);
  display.setCursor(0, 0);
  display.println("clientconnect: error");
  }

  client.print(String("GET ") + url + " HTTP/1.1\r\n" +
               "Host: api.openweathermap.org\r\n" +
               "Connection: close\r\n\r\n");

  unsigned long timeout = millis();
  while (client.available() == 0) {
    if (millis() - timeout > 5000) {
      client.stop();
      return false;
        display.clearDisplay();
        display.setTextSize(1);
        display.setCursor(0, 0);
        display.println("clientavailable: error");
    }
  }

  while (client.available()) {
    String line = client.readStringUntil('\r');
    if (line == "\n") break;
```

```
  }

  String payload = client.readString();
  DynamicJsonDocument doc(1024);
  DeserializationError error = deserializeJson(doc, payload);

  if (error) return false;

  currentWeather.temp = doc["main"]["temp"];
  currentWeather.humidity = doc["main"]["humidity"];
  currentWeather.pressure = doc["main"]["pressure"];
  currentWeather.condition = doc["weather"][0]["main"].as<String>();
  currentWeather.windSpeed = doc["wind"]["speed"];
  currentWeather.icon = getWeatherIcon(currentWeather.condition);

  for (int i = 0; i < 3; i++) {
    forecast[i].temp = currentWeather.temp + random(-3, 3);
    forecast[i].humidity = currentWeather.humidity + random(-10, 10);
    forecast[i].pressure = currentWeather.pressure + random(-5, 5);
    forecast[i].windSpeed = currentWeather.windSpeed + random(-2, 2);
    forecast[i].condition = currentWeather.condition;
    forecast[i].icon = currentWeather.icon;
  }

  return true;
}

void showInstagramStats() {
  display.clearDisplay();
  display.setTextSize(1);

  display.setCursor(0, 0);
  display.println("Exchange rates:");
  display.drawLine(0, 10, 128, 10, WHITE);

  display.setCursor(0, 20);
  display.print("USD: ");
  display.print(rates.usd, 2);
  display.println(" UAH");

  display.setCursor(0, 35);
  display.print("EUR: ");
  display.print(rates.eur, 2);
  display.println(" UAH");

  display.display();
}

void showWeather(int dayOffset) {
  display.clearDisplay();

  if (dayOffset == 0) {
    display.drawBitmap(0, 0, currentWeather.icon, 24, 24, WHITE);
    display.setCursor(30, 5);
    display.setTextSize(1);
    display.print("Now: ");
    display.print(currentWeather.temp);
    display.print("°C ");
    display.println(currentWeather.condition);

    display.setCursor(30, 20);
    display.print("Hum: ");
    display.print(currentWeather.humidity);
```

```
    display.print("%");

    display.setCursor(30, 35);
    display.print("Pres: ");
    display.print(currentWeather.pressure);
    display.print("hPa");

    display.setCursor(30, 50);
    display.print("Wind: ");
    display.print(currentWeather.windSpeed);
    display.print("m/s");
  } else {
    int dayIndex = dayOffset - 1;
    if (dayIndex >= 0 && dayIndex < 3) {
      display.drawBitmap(0, 0, forecast[dayIndex].icon, 24, 24, WHITE);
      display.setCursor(30, 5);
      display.setTextSize(1);
      display.print("Day ");
      display.print(dayOffset);
      display.print(": ");
      display.print(forecast[dayIndex].temp);
      display.print("°C ");
      display.println(forecast[dayIndex].condition);

      display.setCursor(30, 20);
      display.print("Hum: ");
      display.print(forecast[dayIndex].humidity);
      display.print("%");

      display.setCursor(30, 35);
      display.print("Pres: ");
      display.print(forecast[dayIndex].
pressure);
      display.print("hPa");
      display.setCursor(30, 50);
      display.print("Wind: ");
      display.print(forecast[dayIndex].windSpeed);
      display.print("m/s");
    }
  }

  display.display();
}
void updateDisplay() {
  switch (showdisplay) {
    case 1: // Годинник
      if (millis() - clocklast >= 1000) {
        display.clearDisplay();
        display.setTextColor(WHITE, BLACK);
        time_t now = time(nullptr);
        struct tm* timeinfo = localtime(&now);
        // Верхній рядок
        display.setCursor(10, 3);
        display.setTextSize(1);
        display.print(timeinfo->tm_mday);
        display.print("/");
        display.print(timeinfo->tm_mon + 1);
        display.print("/");
        display.print(timeinfo->tm_year + 1900);
        int rssi = WiFi.RSSI();
        display.print("   ");
        if (rssi < -92) display.print("1%");
        else if (rssi > -21) display.print("100%");
```

```
        else                          display.print(round((-0.0154*rssi*rssi)-
(0.3794*rssi)+98.182)), display.print("%");
        // Годинник
        display.setCursor(23, 27);
        display.setTextSize(2);
        display.print(timeinfo->tm_hour);
        display.print(":");
        if (timeinfo->tm_min < 10) display.print("0");
        display.print(timeinfo->tm_min);
        display.print(":");
        if (timeinfo->tm_sec < 10) display.print("0");
        display.println(timeinfo->tm_sec);

        if (millis() - weather_lasttime > 600000) {
          fetchWeatherData();
          getCurrencyRates();
          weather_lasttime = millis();
        }

        display.display();
        clocklast = millis();
      }
      break;

    case 2: // Погода зараз
      showWeather(0);
      break;

    case 3: // Прогноз погоди на завтра (1 день)
      showWeather(1);
      break;

    case 4: // Прогноз погоди на післязавтра (2 дні)
      showWeather(2);
      break;


    case 5: // Instagram
      showInstagramStats();
      break;

    case 6: // Налаштування
      display.clearDisplay();
      display.drawBitmap(96, 1, setting, 24, 24, WHITE);
      display.setCursor(1, 4);
      display.setTextSize(1);
      display.print(serorclient);
      display.setCursor(1, 12);
      display.print(WiFi.localIP());
      display.display();
      break;
  }
}
void setup() {
  Serial.begin(115200);

  Wire.begin(1, 3);
  Wire.setClock(100000);
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 allocation failed"));
    for (;;);
  }
  // Налаштування кнопок
```

```
  pinMode(0, INPUT_PULLUP);
  pinMode(2, INPUT_PULLUP);
  // Підключення WiFi
  WiFiManager wifiManager;
  display.clearDisplay();
  display.println("Connecting WiFi...");
  display.display();

  wifiManager.autoConnect("EspWatch");

  display.clearDisplay();
  display.println("Connected!");
  display.println(WiFi.localIP());
  display.display();
  delay(1000);
  // Налаштування часу
  configTime(timezone, dst, "pool.ntp.org", "time.nist.gov");
  display.clearDisplay();
  display.println("Syncing time...");
  display.display();

  while (!time(nullptr)) {
    delay(500);
    yield();
  }
  // Отримання початкових даних
  fetchWeatherData();
  getCurrencyRates();

  // Запуск сервера
  server.begin();
  WiFi.mode(WIFI_STA);
  wifi_set_sleep_type(LIGHT_SLEEP_T);
}

void loop() {
  ESP.wdtFeed();

  // Обробка кнопок
  if (digitalRead(0) == LOW && digitalRead(2) == HIGH) {
    showdisplay--;
    if (showdisplay < 1) showdisplay = 6;
    updateDisplay();
    delay(200);
  }
  if (digitalRead(2) == LOW && digitalRead(0) == HIGH) {
    showdisplay++;
    if (showdisplay > 6) showdisplay = 1;
    updateDisplay();
    delay(200);
  }
  // Подвійне натискання
  if (digitalRead(2) == LOW && digitalRead(0) == LOW) {
    if (WiFi.status() != WL_CONNECTED) {
      server.stop();
      WiFiManager wifiManager;
      display.clearDisplay();
      display.setCursor(1, 2);
      display.println("Server started at IP: 192.168.4.1");
      display.display();
wifiManager.autoConnect("EspWatch");
      display.println("connected...yeey :)");
      server.begin();
```

```
  } else {
    serorclient = "Client";
  }
}
updateDisplay();
delay(50);
```