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 «МАГІСТР»

ЕЛЕКТРОННИЙ ВИМІРЮВАЧ ВІДСТАНІ НА ОСНОВІ ПЛАТИ NODEMCU, УЛЬТРАЗВУКОВОГО ДАТЧИКА HC-SR04 ТА МОБІЛЬНОГО ДОДАТКУ НА FLUTTER

ELECTRONIC DISTANCE METER BASED ON NODEMCU BOARD, HC-SR04 ULTRASOUND SENSOR AND FLUTTER MOBILE APPLICATION

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

освітня програма Комп'ютерна інженерія
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи Кім-21
Самборик Владислав Олександрович

(підпис)

Керівник:
к.т.н., доцент
Гринюк Сергій Васильович

(підпис)

Кваліфікаційну роботу
допущено до захисту
«____» грудня 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Гринюк Сергій Васильович

(підпис)

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

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

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ проф. Т. ТЕРЛЕЦЬКИЙ
«_____» _____2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Самборику Владиславу Олександровичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Електронний вимірювач відстані на основі плати NodeMCU, ультразвукового датчика HC-SR04 та мобільного додатку на Flutter*

Керівник роботи: *к.т.н., доцент Гринюк Сергій Васильович*

затверджені наказом закладу вищої освіти від « 17 » червня 2025 р. № 290/01-02

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 09.12.2025 р.

3. Вихідні дані до роботи *Основою для виконання роботи слугують науково-технічна література, публікації в профільних періодичних виданнях, а також вітчизняні та зарубіжні дослідження, присвячені даній тематиці, технічні інтернет-ресурси.*

4. Зміст пояснювальної записки (перелік питань, що потрібно розробити):
Вступ
Аналіз проблеми за темою роботи та постановка завдань дослідження основ розробки систем з використанням мікроконтролерів та мобільних застосунків
Теоретична частина
Практична частина
Висновки

5. Перелік графічного (ілюстративного) матеріалу:
До ілюстративних матеріалів належать: зображення користувацьких інтерфейсів, схеми моделей бази даних, архітектурні діаграми системи, а також фрагменти програмного коду.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз проблеми за темою роботи та постановка завдань дослідження основ розробки систем з використанням мікроконтролерів та мобільних застосунків	*Гринюк С.В., доцент*		
Теоретична частина	*Гринюк С.В., доцент*		
Практична частина	*Гринюк С.В., доцент*		
Нормоконтроль	*Багнюк Н.В., доцент*		
Гарант ОП	*Гринюк С.В., доцент*		
Показник запозичень тексту	***___%***		
Академічна доброчесність	*Міскевич О.І., ст.викладач*		

7. Дата видачі завдання 18.06.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1.	*Огляд літератури із досліджуваної проблеми*	15.07.2025	Виконано
2.	*Аналіз проблеми за темою роботи та постановка завдань дослідження основ розробки систем з використанням мікроконтролерів та мобільних застосунків*	15.08.2025	Виконано
3.	*Теоретичне дослідження та практична реалізація*	20.09.2025	Виконано
4.	*Практична реалізація об'єкта проектування*	15.10.2025	Виконано
5.	*Висновки та пропозиції*	19.10.2025	Виконано
6.	*Формування списку використаних джерел*	25.10.2025	Виконано
7.	*Формування додатків*	28.10.2025	Виконано
8.	*Оформлення ілюстративного матеріалу*	01.11.2025	Виконано
9.	*Представлення остаточного варіанту кваліфікаційної роботи керівникові*	10.11.2025	Виконано
10.	*Нормоконтроль*	18.11.2025	Виконано
11.	*Інструментальна перевірка на академічний плагіат*	02.12.2025	Виконано
12.	*Здача кваліфікаційної роботи та всіх супровідних документів на кафедру*	09.12.2025	Виконано

Здобувач вищої освіти _____ Самборик В.О.
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ Гринюк С.В.
(підпис) (прізвище, ініціали)

АНОТАЦІЯ

Самборик В. О. Електронний вимірювач відстані на основі плати NodeMCU, ультразвукового датчика HC-SR04 та мобільного додатку на Flutter. Рукопис.

Кваліфікаційна робота магістра 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.

У першому розділі досліджено теоретичні засади безконтактного вимірювання відстаней, зокрема аналіз ультразвукових технологій на прикладі датчика HC-SR04.

Другий розділ презентує гібридну клієнт-серверну архітектуру, що інтегрує апаратний модуль на базі ESP8266 з ультразвуковим сенсором, мережевий інтерфейс з WiFi AP режимом та мобільний додаток на Flutter.

У третьому розділі проведено дослідження принципів роботи складових елементів системи, розроблено прошивку для NodeMCU для роботи з ультразвуковим сенсором, створено мобільний застосунок за допомогою Flutter, а також проведено ісінтеграцію та синхронізацію усіх компонентів системи.

Об'єкт дослідження – система дистанційного моніторингу відстані з бездротовим інтерфейсом керування.

Предмет дослідження – методи інтеграції ультразвукових сенсорів з мобільними інтерфейсами користувача в умовах обмежених апаратних ресурсів.

Метою роботи є створення енергоефективної системи вимірювання відстаней з можливістю візуалізації даних у мобільному додатку та архівацією результатів.

Ключові слова: HC-SR04, NodeMCU, ESP8266, WebSocket, Flutter, IoT, Real-time monitoring, енергоефективність, температурна компенсація, автоматичне перепідключення.

ANNOTATION

Samboryk V. Electronic Distance Measuring Device Based on NodeMCU Board, Ultrasonic Sensor HC-SR04, and Flutter Mobile Application. Manuscript.

Master's Qualification Thesis in the field of study 123 «Computer Engineering» of the Educational Program «Computer Engineering». Lutsk National Technical University. Lutsk, 2025.

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

The first chapter explores the theoretical foundations of non-contact distance measurement, in particular the analysis of ultrasonic technologies using the HC-SR04 sensor.

The second chapter presents a hybrid client-server architecture integrating the hardware module based on ESP8266 with an ultrasonic sensor, a network interface in WiFi AP mode, and a Flutter mobile application.

The third chapter investigates the operation principles of the system's components, including the development of NodeMCU firmware for ultrasonic sensor operation, the creation of a mobile application using Flutter, and the integration and synchronization of all system components.

Object of study – a wireless distance monitoring system.

Subject of study – methods of integrating ultrasonic sensors with mobile user interfaces under constrained hardware resources.

Purpose of the thesis – to develop an energy-efficient distance measuring system with real-time data visualization in a mobile app and archiving of results.

Keywords: HC-SR04, NodeMCU, ESP8266, WebSocket, Flutter, IoT, real-time monitoring, energy efficiency, temperature compensation, automatic switching.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасних умовах автоматизації виробничих процесів та розвитку систем «розумного будинку», технології точного вимірювання відстаней набувають ключового значення. Особливу важливість ультразвукові сенсори мають у завданнях промислової автоматизації, робототехніки та системах контролю паркування, де необхідна надійна робота у різних умовах навколишнього середовища. Розвиток IoT-рішень та мобільних технологій відкриває нові можливості для створення інтегрованих систем моніторингу з підвищеною енергоефективністю та можливістю дистанційного керування.

Метою роботи є розробка електронного вимірювача відстані на базі плати NodeMCU з ультразвуковим сенсором HC-SR04 та мобільним додатком керування, що забезпечує передачу даних у реальному часі через WiFi-з’єднання з можливістю візуалізації результатів та архівацією вимірів.

Об’єкт дослідження – система дистанційного моніторингу відстані з використанням бездротових технологій передачі даних.

Предмет дослідження – методи інтеграції апаратних компонентів (NodeMCU, HC-SR04) з мобільним інтерфейсом користувача на платформі Flutter через протокол WebSocket.

Завдання:

– дослідити принципи роботи ультразвукових сенсорів;

– реалізувати апаратний модуль вимірювання на базі ESP8266 з ультразвуковим датчиком HC-SR04;

– розробити протокол двосторонньої комунікації через WebSocket з буферизацією даних;

– створити мобільний додаток з інтерфейсом реального часу та локальним сховищем результатів.

Наукова новизна полягає у створенні гібридної архітектури системи вимірювання, яка поєднує низьке енергоспоживання мікроконтролера з можливістю візуалізації даних на мобільних пристроях.

Практичне значення. Система може застосовуватись у системах «розумного паркування», промислових лініях для контролю положення об’єктів, а також як основа для асистивних систем для людей з порушеннями зору. Реалізований механізм збереження історії вимірів відкриває можливості для подальшого аналізу тенденцій та статистичної обробки даних.

Особистий внесок. Автором проведено проектування схеми підключення сенсора, розробка оптимальних алгоритмів вимірювання з мінімізацією затримок, створення трирівневої системи обробки помилок (апаратний рівень, мережевий протокол, мобільний додаток).

Апробація результатів. Результати роботи представлені у науковій статті, опублікованій у Студентському науковому віснику Луцького національного технічого університету, випуск №54 (додаток А).

Публікації. Результати досліджень, виконаних під час роботи над кваліфікаційною роботою магістра опубліковано в науковій статті [1].

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ СИСТЕМ З ВИКОРИСТАННЯМ МІКРОКОНТРОЛЕРІВ ТА МОБІЛЬНИХ ЗАСТОСУНКІВ

1.1 Огляд та класифікація сенсорних технологій у вимірюванні відстані

Вимірювання відстані є фундаментальним завданням у багатьох галузях науки, промисловості, робототехніки та побутових технологій. Технології, що забезпечують таке вимірювання, базуються на різних фізичних принципах і мають різні технічні характеристики, які визначають доцільність їх використання в певних умовах.

Загалом, сенсорні технології для визначення відстані класифікують за фізичним принципом дії. Найбільш поширеними є ультразвукові, інфрачервоні, лазерні, радарні та ємнісні сенсори. Ультразвукові сенсори використовують акустичні хвилі й вимірюють час повернення сигналу, що відбився від об’єкта (рис 1.1).

Рисунок 1.1 – Принцип роботи ультразвукових сенсорів [2]

Вони відносно дешеві та прості в реалізації, однак схильні до похибок за наявності шуму або м’яких поверхонь. Інфрачервоні сенсори визначають відстань за інтенсивністю або фазовим зсувом відбитого ІЧ-сигналу, але їхня точність сильно залежить від властивостей поверхні та умов освітлення (рис. 1.2).

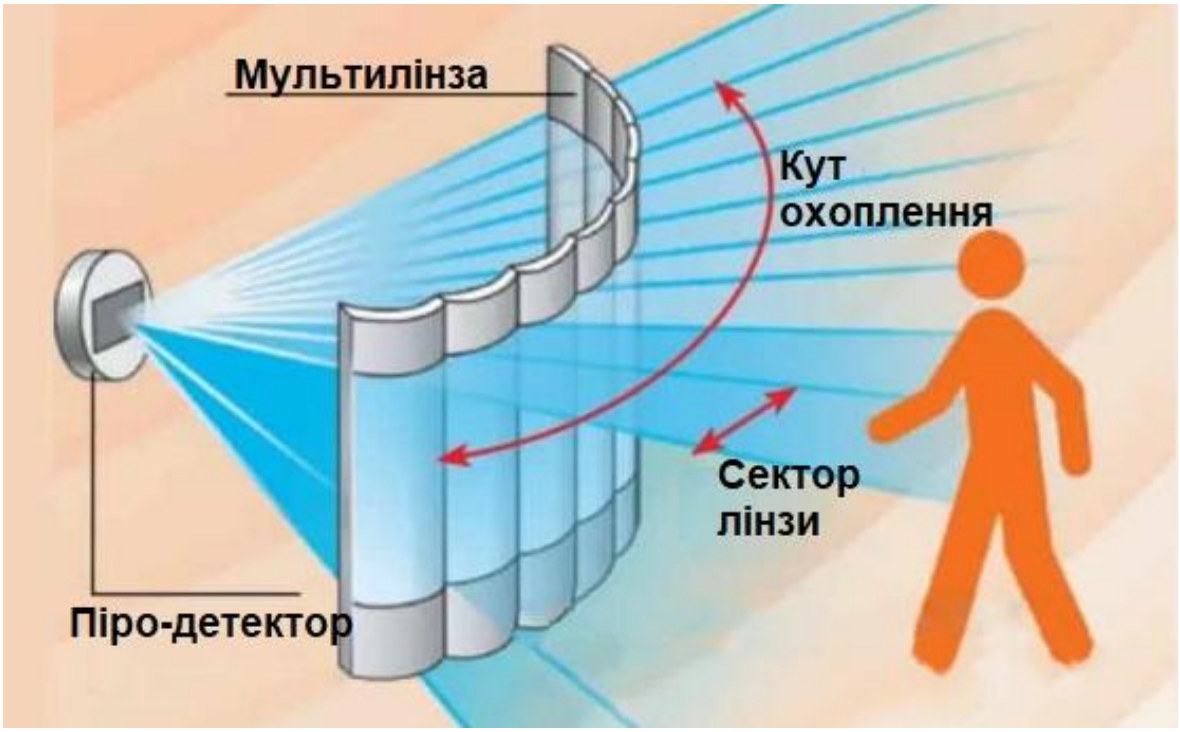

Рисунок 1.2 – Принцип роботи інфрачервоних сенсорів [2]

Лазерні сенсори, зокрема ті, що використовують принцип тріангуляції або час польоту (ToF), забезпечують високу точність і швидкодію, що робить їх придатними для систем автоматизованого контролю та 3D-сканування (рис. 1.3).

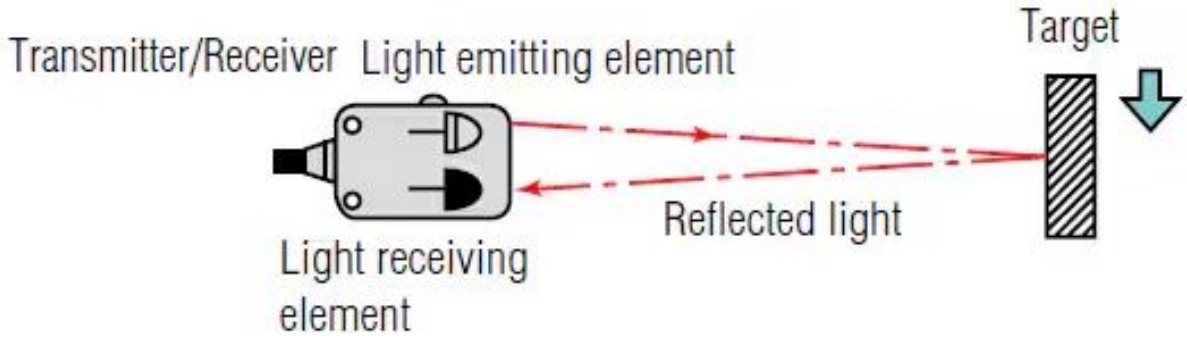

Рисунок 1.3 – Принцип роботи лазерних сенсорів [3]

Радарні сенсори, в свою чергу, застосовують радіохвилі для детекції об’єктів і визначення їх відстані (рис. 1.4). Вони демонструють високу надійність у складних середовищах, зокрема в умовах пилу, туману або опадів.

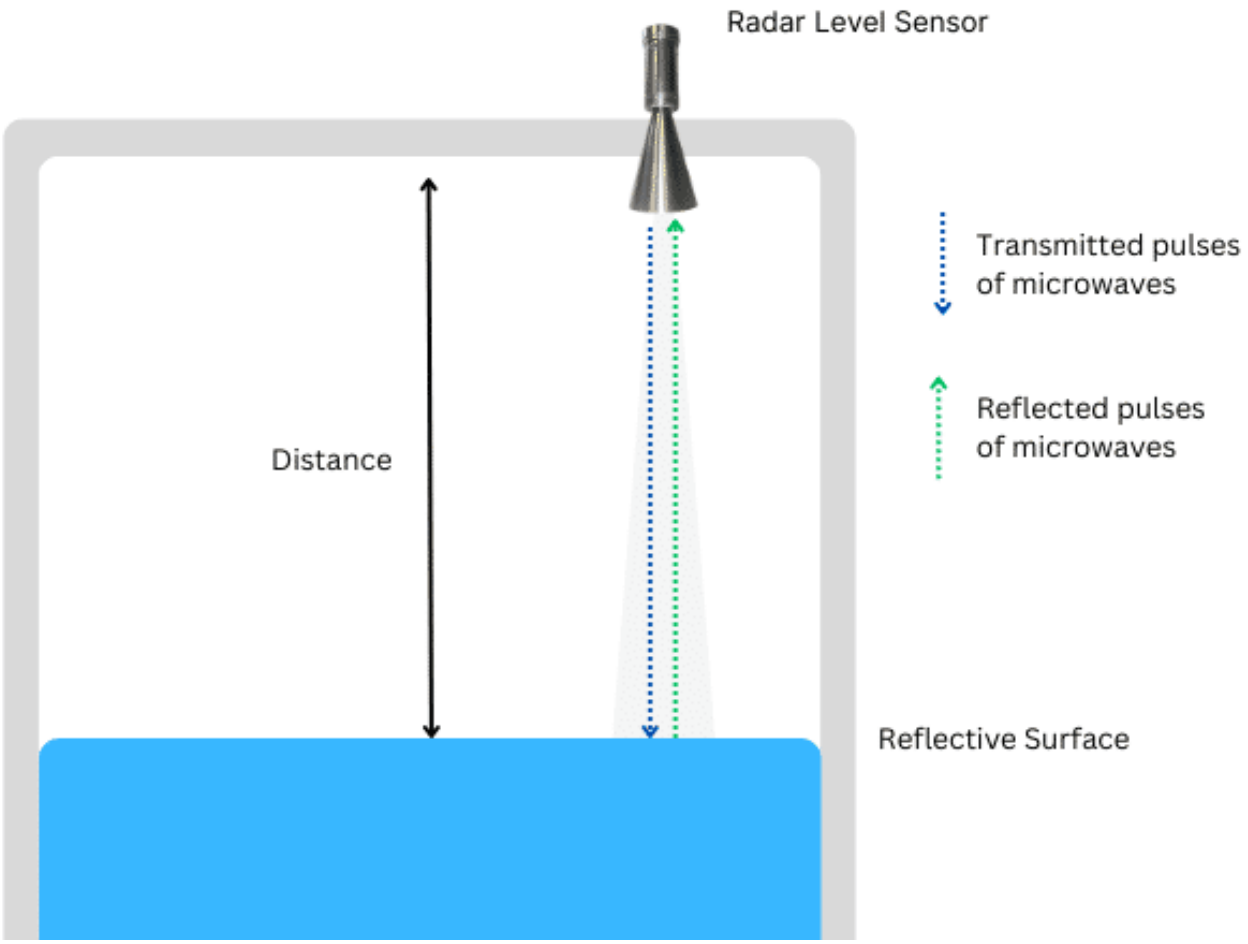

Рисунок 1.4 – Принцип роботи радарних сенсорів [4]

Ємнісні та індуктивні сенсори забезпечують надзвичайну точність на малих відстанях, проте вони чутливі до складу та форми поверхні об’єкта.

Сучасні тенденції також включають розвиток сенсорів, заснованих на комбінованих технологіях, наприклад, LiDAR або ToF-камер, які поєднують в собі високу точність, просторову роздільну здатність і можливість обробки в реальному часі. Така гібридизація дозволяє адаптувати сенсорні системи до складних динамічних середовищ і підвищувати надійність вимірювання.

Схема роботи LiDAR сенсорів зображена на рисунку 1.5.

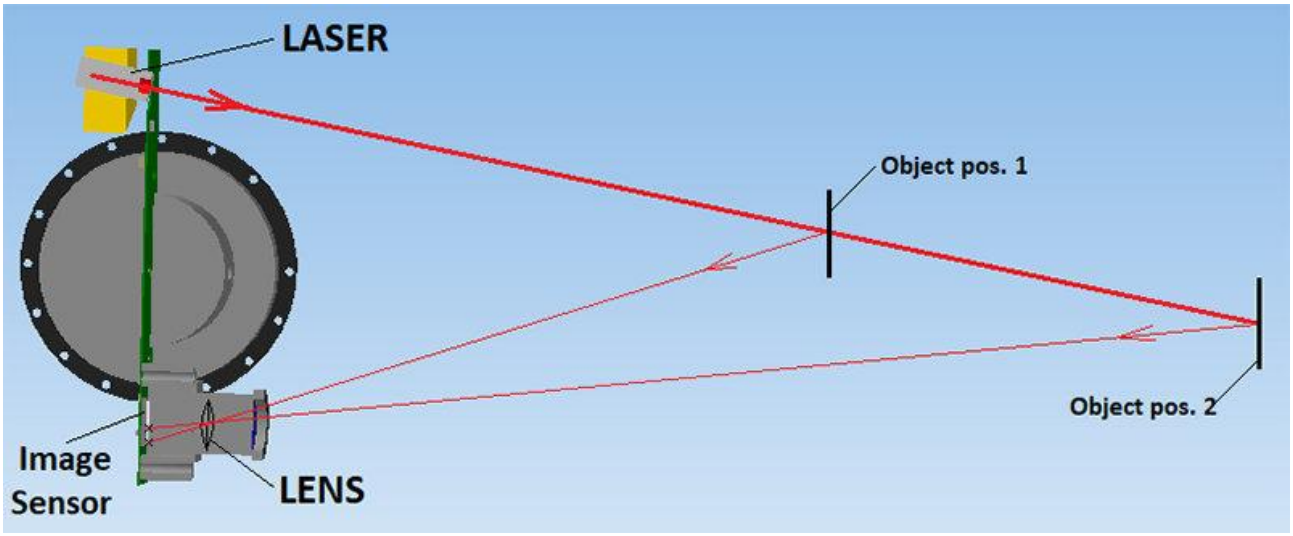

Рисунок 1.5 – Принцип роботи LiDAR сенсора [5]

Таким чином, огляд та класифікація сенсорних технологій демонструє значне розмаїття методів і підходів, кожен з яких має свої переваги та обмеження, що зумовлює необхідність ретельного вибору технології залежно від конкретних вимог прикладного завдання. Детальне порівняння сенсорних технологій описане в таблиці 1.1.

Таблиця 1.1 – Порівняння видів сенсорів

Технологія	Принцип	Дальність	Точність	Переваги	Недоліки
Ultrasonic	Звук (ToF)	до ≈8 м	мм - см	Недорогі, нечутливі до кольору	Низька точність, широкий звуковий конус
Infrared	Інтенсивність / фаза	до ~1 м	мм - см	Компактні, швидка реакція	Залежні від освітлення, чутливі до властивостей поверхні
Laser (ToF / Triangulation)	Лазер + ToF або тріангуляція	Короткий – до кілометрів	мікрон - мм	Дуже точні, великий діапазон вимірювань	Чутливі до атмосфери й поверхні
ToF (камери / сенсори)	Пульс / фаза світла	до кількох метрів	мм	Реальний час, глибина сцени	Потребують калібрування, складні в реалізації
LIDAR	3D-сканування лазером	Сотні метрів – км	см - мм	Детальні 3D-моделі, автономні системи	Висока вартість, вплив погодних умов

Продовження таблиці 1.1

Технологія	Принцип	Дальність	Точність	Переваги	Недоліки
Radar	Радіохвиля (ToF / доплер)	Великі відстані	до кількох см	Працює у складних погодних умовах	Менш точні на близьких відстанях, слабкі для дрібних об’єктів
Капацитивні / індуктивні	Електричне поле / віхреві струми	<1 м	Дуже висока	Надточні на малих відстанях	Залежні від середовища та типу матеріалу
Фотоелектричні	Світло / відбиття	Десятки мм - метри	мм	Простота конструкції, різні режими	Залежні від рефлексії, потребують джерела світла

1.2 Особливості реалізації IoT-рішень на базі NodeMCU

Платформа NodeMCU, побудована на базі мікроконтролера ESP8266, займає ключове місце в проєктуванні та реалізації систем Інтернету речей (IoT) завдяки своїй відкритій архітектурі, низькій вартості, компактним розмірам і широким комунікаційним можливостям. Архітектурно NodeMCU інтегрує мікроконтролер з повноцінним TCP/IP-стеком, що дозволяє здійснювати підключення до бездротових мереж без потреби в додаткових модулях або складних конфігураціях (рис. 1.6).

Рисунок 1.6 – Плата NodeMCU

Однією з основних переваг використання NodeMCU в IoT-середовищі є підтримка високорівневих мов програмування, зокрема Lua та C/C++, що забезпечує гнучкість і швидкість розробки. Крім того, платформа має вбудований USB-UART міст, який значно спрощує прошивання мікроконтролера та налагодження програмного забезпечення. Завдяки інтеграції з популярними середовищами розробки, такими як Arduino IDE, PlatformIO чи ESPHome, NodeMCU дозволяє швидко створювати прототипи розумних пристроїв з мінімальними технічними бар'єрами.

З технічної точки зору, ESP8266 забезпечує надійне підключення до Wi-Fi мереж і підтримує сучасні стандарти безпеки, включаючи WPA2, TLS/SSL і можливість використання сертифікатів для шифрування трафіку. Це особливо важливо в контексті IoT, де безпека даних відіграє критичну роль. Крім мережевих можливостей, ESP8266 має достатній обсяг флеш-пам'яті та оперативної пам'яті для виконання нескладних обчислень, збору сенсорних даних, обробки сигналів та віддаленого управління пристроями.

NodeMCU також підтримує широкий спектр периферійних інтерфейсів, включаючи GPIO, I²C, SPI та UART, що забезпечує просту інтеграцію з різними сенсорами, виконавчими пристроями та іншими електронними модулями. Це дозволяє використовувати дану платформу для створення систем моніторингу, управління, автоматизації домашнього простору, а також у промислових умовах – як елемент розподіленої сенсорної мережі.

Таким чином, NodeMCU на базі ESP8266 виступає ефективним і доступним інструментом для реалізації IoT-рішень, поєднуючи простоту апаратної реалізації з гнучкістю програмного забезпечення, високим рівнем інтеграції мережевих технологій та широкою екосистемою підтримки.

1.3 Технології розробки мобільних застосунків

Сучасна розробка мобільних застосунків базується на широкому спектрі підходів, серед яких найпоширенішими є нативна, гібридна та кросплатформна

розробка (рис. 1.7). Кожна з цих парадигм має свої архітектурні особливості, переваги та обмеження, які визначаються як технічними характеристиками середовища виконання, так і бізнес-вимогами проєкту.

Рисунок 1.7 – Різні підходи до розробки мобільних застосунків [6]

Нативна розробка передбачає використання мов програмування та інструментарію, специфічного для конкретної платформи – Java/Kotlin для Android та Swift/Objective-C для iOS. Такий підхід забезпечує максимальну продуктивність, стабільність і доступ до всіх апаратних ресурсів пристрою, що особливо важливо для реалізації застосунків з високими вимогами до графіки, мультимедіа чи сенсорної взаємодії. Водночас він вимагає розробки окремого коду для кожної платформи, що збільшує витрати на супровід, тестування та оновлення.

У відповідь на потребу в уніфікації процесу розробки виникли кросплатформні фреймворки, серед яких Kotlin Multiplatform, Flutter, React Native та Xamarin. Кожна з них має власну філософію, рівень доступу до нативного функціоналу та особливості компіляції. Наприклад, React Native використовує JavaScript і забезпечує інтеграцію з нативними компонентами через міст, що може впливати на продуктивність у складних UI-сценаріях. Flutter, розроблений Google, побудований на мові Dart і генерує власні

рендеринг-компоненти, що забезпечує стабільну продуктивність та однаковий вигляд на всіх платформах, однак потребує більше ресурсів для інтеграції з нативними SDK. Xamarin, що базується на C#, дозволяє використовувати єдиний стек .NET, але має більші обмеження у візуальному відтворенні нативного UI. У свою чергу, Kotlin Multiplatform розділяє логіку на спільну (shared) та платформенно-специфічну частини, дозволяючи повторно використовувати бізнес-логіку між Android, iOS, web та десктопними платформами без втрати контролю над нативною реалізацією.

Їх мета полягає у створенні спільного програмного коду, переважно для бізнес-логіки, з можливістю використання нативних компонентів інтерфейсу для кожної ОС. Kotlin Multiplatform, зокрема, орієнтований на спільне використання коду між Android, iOS, Web та Desktop, залишаючи за нативними платформами відповідальність за UI, доступ до апаратного забезпечення та специфічні API. Це дозволяє ефективно масштабувати застосунки без втрати продуктивності чи функціональності, характерної для нативних рішень. Оцінюючи графік пошукових запитів в Google Trends (рис. 1.8) можна зробити висновок, що інтерес до крос-платформної розробки лише зростає.

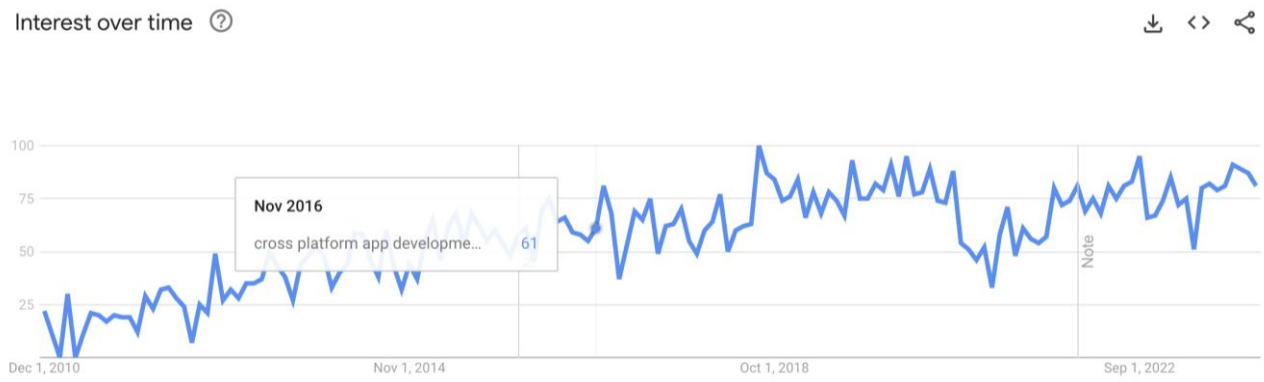

Рисунок 1.8 – Зростання інтересу до крос-платформної розробки згідно з Google Trends [7]

Альтернативні гібридні підходи зазвичай базуються на використанні вебтехнологій (HTML, CSS, JavaScript), які виконуються всередині спеціального WebView-контейнера. Хоча такий спосіб пришвидшує розробку і знижує

витрати, він істотно обмежує продуктивність та доступ до нативного функціоналу. Тому в більшості сучасних проєктів гібридні фреймворки поступово витісняються кросплатформними рішеннями, які поєднують переваги уніфікованої логіки та нативного інтерфейсу.

Flutter – це UI-бібліотека, розроблена компанією Google, яка спочатку була спрямована на створення нативних, продуктивних і візуально привабливих мобільних застосунків. Проте концепція Flutter виходить далеко за межі лише мобільної розробки: вона передбачає створення інтерфейсів користувача для будь-яких типів екранів, забезпечуючи кросплатформену розробку з єдиною кодовою базою (рис. 1.9).

Рисунок 1.9 – Можливості платформи Flutter [8]

1.4 Огляд способів взаємодії між мікроконтролером та мобільним застосунком

У контексті розвитку систем Інтернету речей та мобільних технологій надзвичайно важливою є ефективна організація зв’язку між мікроконтролерами та мобільними застосунками. Такий зв’язок забезпечує обмін даними в реальному часі, дистанційне керування пристроями, а також моніторинг стану систем. У загальному випадку цей зв’язок може бути реалізований з

використанням дротових або бездротових інтерфейсів, однак у мобільних умовах перевагу, як правило, надають бездротовим технологіям. Серед них найбільш поширеними є Wi-Fi, Bluetooth (зокрема BLE), мобільний інтернет (через GSM/3G/4G-модулі), а також менш поширені рішення, такі як NFC та LoRa.

Bluetooth Low Energy (BLE) є однією з ключових технологій для реалізації енергозберігаючого зв'язку на коротких відстанях. Цей протокол забезпечує низьке енергоспоживання мікроконтролера та дозволяє обмінюватися невеликими обсягами даних з мобільним додатком, що є доцільним у задачах, пов'язаних із сенсорними модулями або трекерами. BLE реалізується через стандартизовані профілі (GATT), що дозволяє використовувати універсальні бібліотеки на мобільному боці, зокрема для платформ Android та iOS. Особливістю BLE є обмежена пропускна здатність, але надзвичайно висока енергоефективність, що робить цю технологію ідеальною для застосування у пристроях, які живляться від батарей або акумуляторів.

Wi-Fi, натомість, забезпечує значно вищу швидкість передавання даних і підтримує одночасну взаємодію з хмарними сервісами. Завдяки використанню стандартного TCP/IP-протоколу, Wi-Fi дає змогу розробникам реалізовувати як прямий зв'язок між мобільним застосунком і пристроєм, так і опосередковану комунікацію через проміжний сервер або MQTT-брокер. Це відкриває широкі можливості для побудови масштабованих систем з кількома пристроями, що взаємодіють між собою через мережу, а також для інтеграції з вебінтерфейсами або голосовими асистентами. Недоліком Wi-Fi є порівняно високе енергоспоживання, тому така технологія частіше використовується у пристроях, які мають доступ до стабільного джерела живлення.

Крім BLE та Wi-Fi, важливу роль у взаємодії між мікроконтролерами й мобільними або веб-застосунками відіграють мережеві протоколи обміну даними, серед яких ключовими є MQTT, HTTP та WebSocket [9].

MQTT (Message Queuing Telemetry Transport) – це легковаговий публікаційно-підписний протокол, розроблений спеціально для систем із

обмеженими ресурсами, нестабільним з'єднанням або потребою в мінімальному енергоспоживанні. У контексті IoT-систем, MQTT дозволяє пристроям обмінюватися даними через посередника – брокера. Наприклад, ESP8266 публікує дані сенсорів у певну 2 «тему»(topic), а мобільний застосунок або сервер, підписаний на цю тему, миттєво їх отримує. Завдяки своїй архітектурі MQTT добре підходить для телеметрії, моніторингу станів, надсилання команд на пристрої, а також для роботи в умовах низької пропускної здатності або великої кількості вузлів (рис. 1.10).

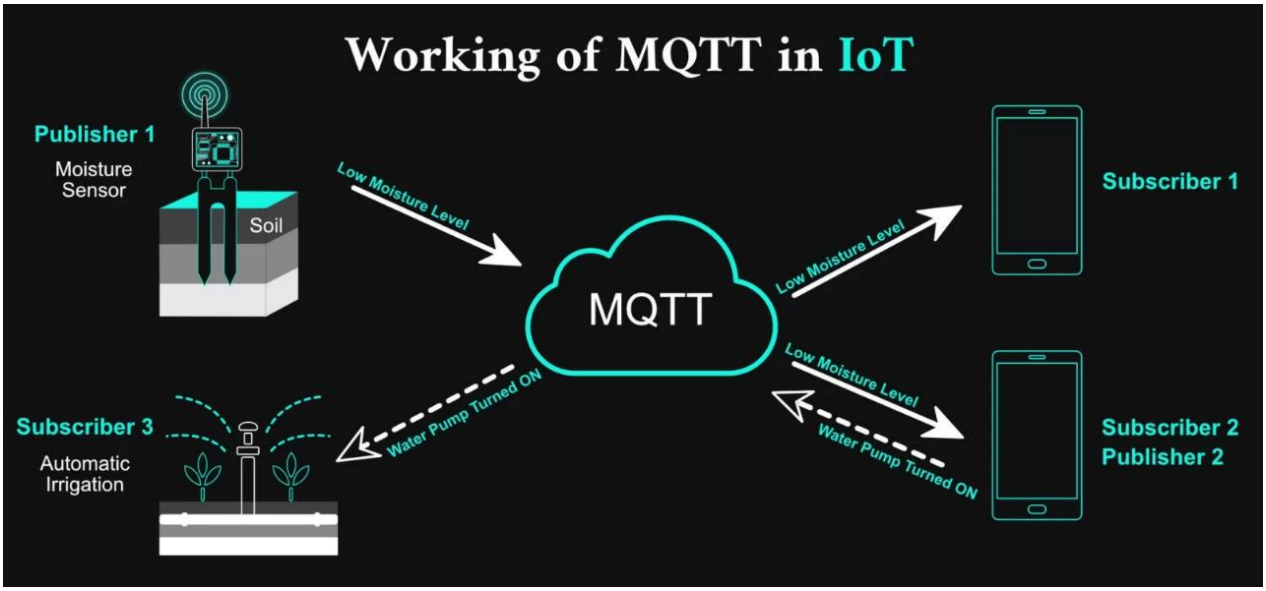

Рисунок 1.10 – Принцип роботи MQTT [10]

До переваг MQTT належать низька затримка, надійна доставка повідомлень (QoS), підтримка ретенційних повідомлень та збереження останнього стану.

HTTP – один з найпоширеніших протоколів, який використовують для запитів до веб-серверів. У IoT-сценаріях HTTP зазвичай застосовують для періодичного обміну даними між пристроєм і сервером: наприклад, мікроконтролер надсилає показники на бекенд, а мобільний застосунок опитує сервер, щоб їх отримати. Такий підхід простий у реалізації, легко налагоджується через RESTful API, добре масштабовується, але має недолік у вигляді високих

затримок та неефективності при потребі в постійному зв’язку або миттєвій реакції (рис. 1.11).

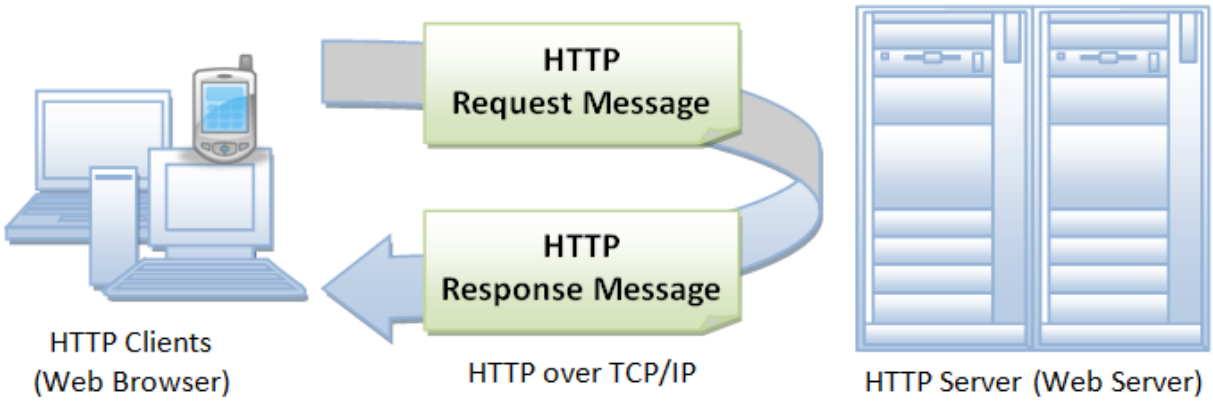

Рисунок 1.11 – Принцип роботи HTTP [11]

WebSocket – це протокол двостороннього зв’язку поверх TCP, який дозволяє встановити постійне з’єднання між клієнтом і сервером (рис. 1.12).

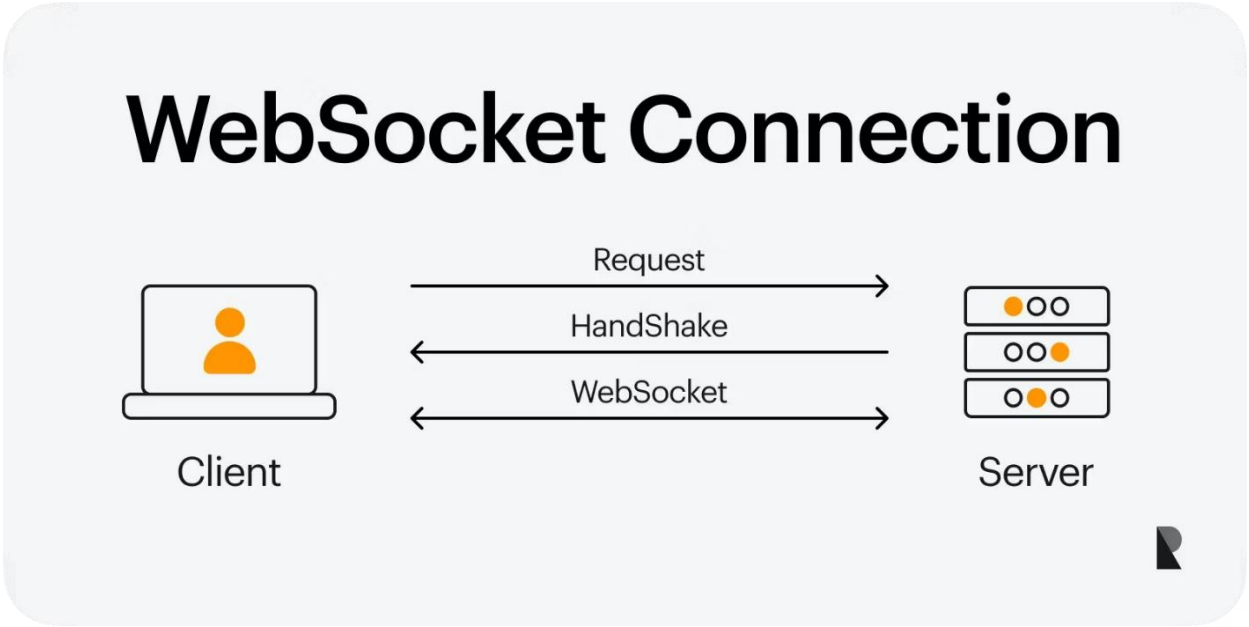

Рисунок 1.12 – Принцип роботи WebSocket [12]

У контексті мобільних або вебзастосунків, WebSocket дозволяє миттєво передавати дані від пристрою до застосунку й навпаки, без потреби постійно опитувати сервер, як у випадку з HTTP. Це забезпечує низьку затримку та

економію ресурсів. Наприклад, WebSocket чудово підходить для побудови реального часу інтерфейсів керування (dashboard) або чат-подібних командних систем керування пристроями.

У загальному, вибір між MQTT, HTTP та WebSocket залежить від особливостей задачі. Зазвичай MQTT оптимальний для великої кількості пристроїв, телеметрії, асинхронного обміну, HTTP підходить для простих REST-запитів, інтероперабельності з веб-сервісами, а WebSocket забезпечує інтерактивний, двосторонній обмін даними в реальному часі.

Сучасні системи IoT можуть поєднувати кілька протоколів одночасно для досягнення оптимального балансу між швидкістю, стабільністю, енергоефективністю та масштабованістю.

РОЗДІЛ 2
ПРОЄКТУВАННЯ СИСТЕМИ ВИМІРЮВАННЯ ВІДСТАНІ

2.1 Технічні вимоги та специфікація системи

Розробка електронного вимірювача відстані передбачає точну інтеграцію апаратного та програмного забезпечення з урахуванням обмежень щодо живлення, продуктивності, бездротового зв’язку та інтерфейсу користувача. Мікроконтролерна плата NodeMCU, побудована на базі чипа ESP8266, що забезпечує достатню обчислювальну потужність та підтримку бездротових з’єднань за стандартом Wi-Fi 802.11 b/g/n. Вона функціонує при напрузі 3,3 В, має вбудований перетворювач USB-UART для зручного програмування та зчитування даних, а також достатню кількість GPIO-портів для підключення датчиків.

Сенсорна частина реалізується за допомогою ультразвукового датчика HC-SR04, що здатен вимірювати відстань у діапазоні приблизно від 2 до 400 см з точністю до кількох міліметрів (рис. 2.1).

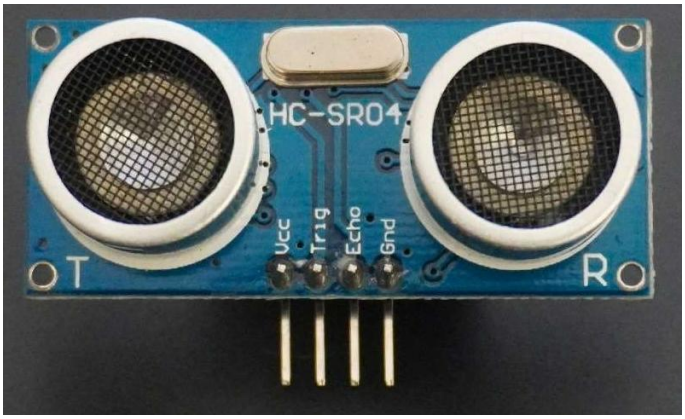

Рисунок 2.1 – Зовнішній вигляд HC-SR04

Робочий принцип базується на емісії короткого ультразвукового імпульсу та фіксації часу його відбиття від перешкоди. Підключення HC-SR04 до NodeMCU вимагає формування тригерного сигналу тривалістю 10 мкс та зчитування ехо-сигналу за допомогою таймера з мікросекундною точністю. У

зв’язку з тим, що логічні рівні датчика становлять 5 В, а мікроконтролер працює на 3,3 В, застосовується подільник напруги або логічний рівневий перетворювач для захисту GPIO.

З програмного боку реалізація системи вимагає середовища розробки Arduino IDE або PlatformIO, з відповідними бібліотеками для взаємодії з датчиком HC-SR04 (наприклад, NewPing.h) та мережевими інтерфейсами (ESP8266WiFi.h, ESPAsyncWebServer.h, ArduinoJson.h) [13]. Основним протоколом комунікації між мікроконтролером і мобільним застосунком є HTTP або WebSocket, які забезпечують передачу виміряних даних у реальному часі. Мікроконтролер виконує роль веб-сервера, розміщуючи інтерфейс WebSocket, REST API або відкриту вебсторінку для візуалізації результатів, що дозволяє переглядати відстань навіть через браузер безпосередньо в локальній мережі.

Для мобільного застосунку, створеного на Flutter, технічними вимогами є підтримка платформи Android (від версії 7.0 і вище) та доступ до локальної мережі для підключення до IP-адреси плати NodeMCU. Інтерфейс застосунку повинен реалізовувати парсинг вхідних даних з JSON-формату, відображення значення відстані у зрозумілому форматі, а також опціонально – збереження логів вимірювань або побудову графіка змін. Надійна робота вимагає також реалізації базової обробки помилок: перевірки з’єднання, коректного формату даних і тайм-аутів при запитах до пристрою. Уся система повинна демонструвати низьку затримку вимірювання й передачі результату (менше 500 мс) та стабільну роботу в умовах тривалого використання.

2.2 Вибір та обґрунтування архітектурного рішення

У процесі проектування електронного вимірювача відстані критично важливим є вибір відповідної архітектури системи, яка забезпечить ефективну взаємодію між апаратним модулем на базі плати NodeMCU та програмним забезпеченням, розгорнутим у вигляді мобільного застосунку. Основна ідея архітектурного рішення полягає в побудові клієнт-серверної моделі, де

NodeMCU виконує роль сервера, який періодично вимірює відстань за допомогою ультразвукового датчика HC-SR04, обробляє ці дані та надає їх для доступу по бездротовому каналу. З іншого боку, мобільний застосунок, реалізований за допомогою фреймворку Flutter, виконує роль клієнта, який ініціює запити до мікроконтролера, отримує від нього значення відстані та відображає їх у зручному для користувача форматі.

Важливою частиною архітектури системи є механізм передачі даних між мікроконтролером NodeMCU та мобільним застосунком. З огляду на необхідність отримання вимірювань у режимі, наближеному до реального часу, було прийнято рішення застосувати протокол WebSocket, який забезпечує двосторонню постійну комунікацію між клієнтом і сервером з мінімальною затримкою. На відміну від HTTP, WebSocket дозволяє уникнути багаторазового встановлення TCP-з’єднань, що критично важливо для систем з обмеженими апаратними ресурсами, таких як NodeMCU.

NodeMCU виконує функцію WebSocket-сервера, який слухає вхідні запити та у відповідь надсилає відстань, розраховану за допомогою ультразвукового датчика HC-SR04. На мобільному боці застосунок, реалізований у Flutter, виступає WebSocket-клієнтом, що ініціює з’єднання, отримує повідомлення у форматі JSON і відображає їх в інтерфейсі користувача. Такий підхід забезпечує не лише оперативність оновлення даних, а й ефективну енергозатратність, що особливо актуально в умовах мобільного середовища.

Додатково, вибір саме Flutter зумовлений його підтримкою кросплатформенності та інтегрованим пакетом web_socket_channel, що спрощує реалізацію комунікаційного рівня. Це дозволяє мінімізувати витрати часу на розробку та забезпечити однакову поведінку застосунку як на Android-, так і на iOS-пристроях.

Офіційна документація платформи підкреслює, що Flutter активно розширюється, охоплюючи Android, iOS, Web і Desktop. При певних додаткових зусиллях розробник може використовувати єдиний код для створення вебдодатків (що конкурують із SPA-фреймворками на кшталт React, Angular чи

Vue), десктопних застосунків (альтернатива Electron або Qt), а в перспективі – і для вбудованих систем.

Flutter Mobile використовує власний рушій рендерингу Skia (рис. 2.2), який забезпечує повний контроль над пікселями екрану та високу продуктивність. Власні віджети (widgets), реалізовані на мові Dart, дозволяють досягати нативної швидкодії без залучення сторонніх компонентів.

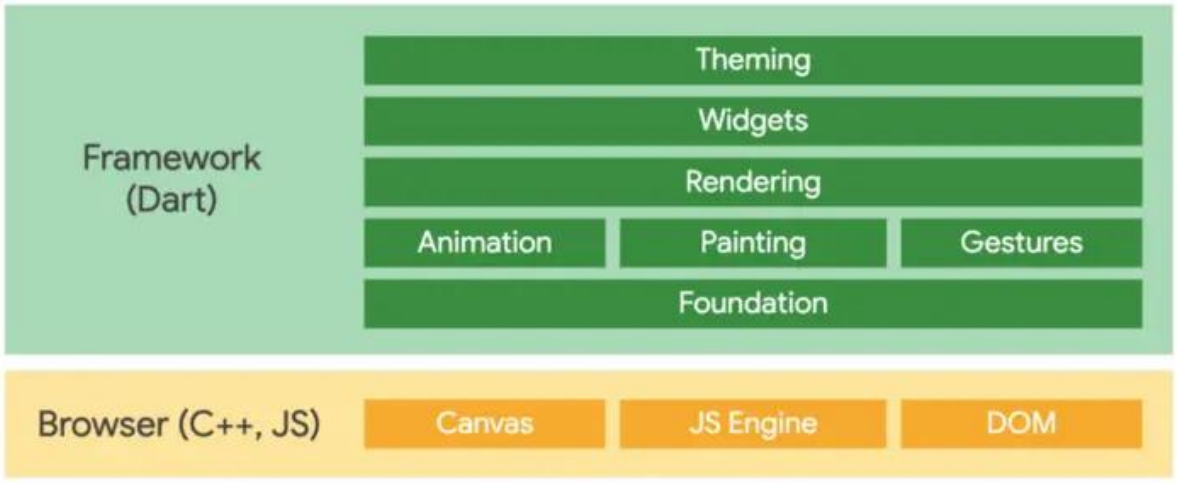

Рисунок 2.2 – Принцип роботи рушія Skia [8]

Рушій Skia спочатку був створений для браузера Chrome, що дозволило його ефективну інтеграцію в екосистему Flutter. У вебверсії Flutter принцип залишився схожим: платформа сприймає екран як суцільне полотно (canvas) і відображає елементи інтерфейсу як HTML-структури, використовуючи стандартні вебтехнології (HTML, CSS, JavaScript). Завдяки цьому, весь функціонал Flutter – анімації, маршрутизація, компоненти – залишається доступним без необхідності окремого кодування для Web.

Flutter є мобільним SDK, за допомогою якого можна створювати високоякісні нативні застосунки для iOS та Android (рис. 2.3). Крім того, Flutter є основним інструментом для кросплатформеної розробки у межах нової операційної системи Google – Fuchsia. Його архітектура дозволяє створювати інтерфейси, які є водночас естетичними, продуктивними та адаптивними до різних пристроїв.

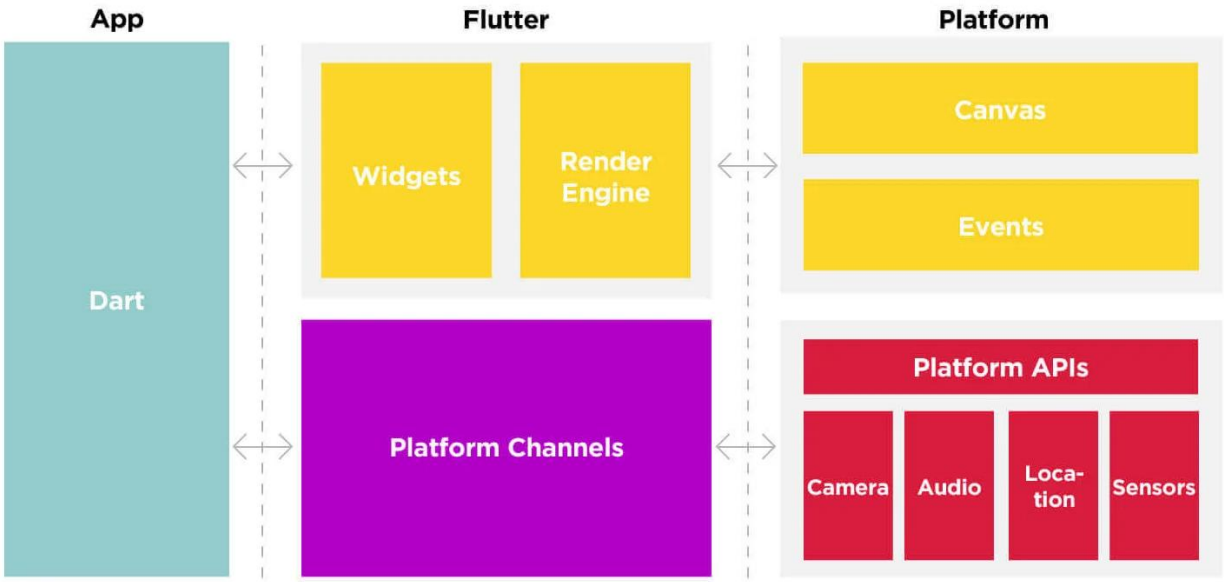

Рисунок 2.3 – Принцип взаємодії Skia та нативних API [14]

Основна парадигма розробки у Flutter базується на використанні віджетів (widgets), які є базовими будівельними елементами як графічного інтерфейсу (наприклад, кнопки чи текст), так і логіки взаємодії. Фреймворк забезпечує розширюваність, що дозволяє легко додавати нові функції або змінювати поведінку наявних компонентів.

Віджети у Flutter поділяються на два основні типи – Stateless та Stateful.

Stateless Widgets, відповідно до своєї назви, не мають внутрішнього стану, тобто їхня структура та вигляд залишаються незмінними протягом життєвого циклу. Такі компоненти найчастіше використовуються для відображення статичних елементів інтерфейсу – наприклад, простих текстових міток або кнопок.

На відміну від них, Stateful Widgets мають внутрішній стан, який може змінюватися з часом, що впливає як на зовнішній вигляд, так і на функціональність компонента. Stateful Widgets часто застосовуються для інтерактивних елементів, таких як поля введення або анімаційні контролери, де поведінка змінюється у відповідь на дії користувача чи події в системі.

Створення обох типів віджетів реалізується за допомогою мови програмування Dart, яка є офіційною мовою для Flutter. Dart має у своєму

розпорядженні розвинуту інфраструктуру інструментів розробки, зокрема Dart Analyzer для статичного аналізу коду та інші утиліти для налагодження та тестування, включно з інструментами на кшталт Vibration Tester.

У результаті, архітектура Flutter поєднує декларативний стиль розробки з повним контролем над рендерингом, дозволяючи створювати продуктивні та візуально привабливі застосунки, незалежно від платформи.

Flutter побудований як розширювана багаторівнева система (рис. 2.4), у якій кожен рівень представлений незалежною бібліотекою, що опирається на нижчий шар. Важливою особливістю є відсутність привілейованого доступу між рівнями: кожен із них є ізольованим, опціональним і замінним, що забезпечує гнучкість та модульність архітектури.

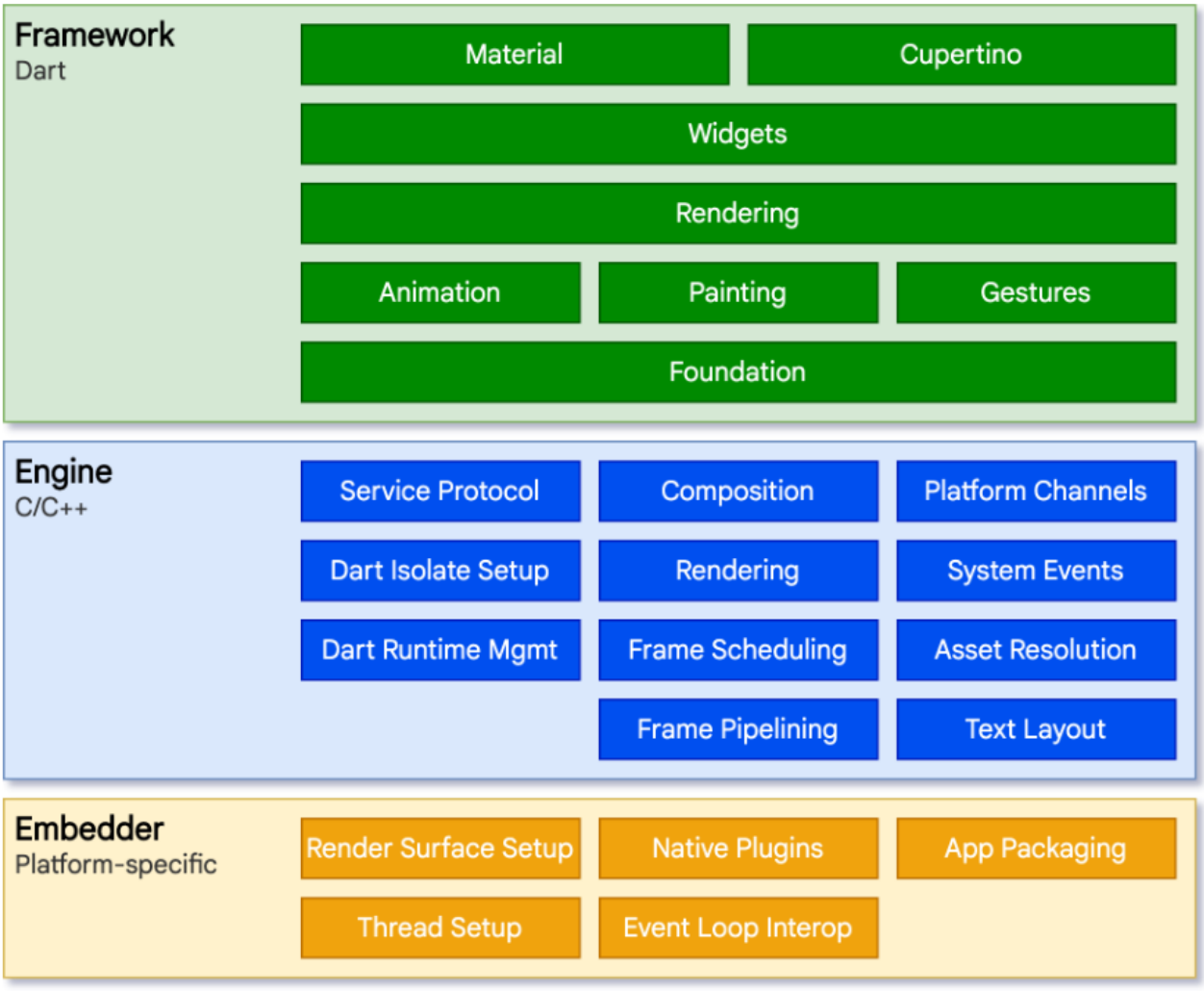

Рисунок 2.4 – Багаторівнева архітектура Flutter [15]

На рівні операційної системи Flutter-застосунки виглядають як звичайні нативні програми. Для кожної платформи використовується embedder – низькорівневий модуль, який ініціює запуск застосунку, взаємодіє з ОС (отримуючи доступ до поверхонь рендерингу, подій введення, систем доступності тощо), а також керує подійним циклом повідомлень. Embedder реалізується мовою, притаманною конкретній платформі: Java та C++ для Android, Swift та Objective-C/Objective-C++ для iOS та macOS, C++ для Windows і Linux. Embedder може використовуватися як для повноцінного Flutter-застосунку, так і для вбудовування Flutter-модуля у більший нативний застосунок. Окрім офіційних вбудованих варіантів, існують також сторонні рішення для інших платформ.

У ядрі Flutter знаходиться рушій Flutter Engine, написаний переважно мовою C++. Його функціональність охоплює всі базові потреби для запуску застосунків Flutter: растеризацію сцен, графічний рендеринг (за допомогою рушія Impeller на iOS, Android та деяких десктопних платформах або Skia на інших), компонування тексту, введення-виведення файлів і мережі, підтримку доступності, роботу з плагінами та виконання Dart-коду. Flutter Engine включає в себе компілятор та середовище виконання Dart.

Доступ до рушія з Dart-коду здійснюється через бібліотеку dart:ui, яка надає низькорівневі інтерфейси для роботи з введенням, графікою та рендерингом тексту. Вона служить як місток між C++-реалізацією рушія та високорівневими Dart-класами.

Найчастіше взаємодія розробника з Flutter відбувається на рівні Flutter Framework – реактивного, сучасного фреймворку, реалізованого повністю на Dart. Він складається з кількох послідовних рівнів:

– фундаментальний рівень, що містить базові класи та сервіси, такі як анімація, малювання, обробку жестів. Ці абстракції є основою для вищих рівнів;

– рендеринг-рівень, що відповідає за побудову дерева об’єктів, що можуть бути візуалізовані. Зміни у структурі дерева автоматично призводять до оновлення розмітки;

– рівень віджетів, що реалізує механізм композиції. Кожен render object має відповідний клас у цьому шарі, а також розробник може комбінувати класи у нові компоненти. Саме тут вводиться реактивна модель програмування.

Бібліотеки Material і Cupertino – забезпечують реалізацію компонентів інтерфейсу у стилі Material Design (Android) або iOS Design відповідно. Вони побудовані поверх шару віджетів.

Варто зазначити, що сам Flutter Framework є відносно невеликим. Багато високорівневих функцій, які часто використовуються розробниками, реалізуються у вигляді зовнішніх пакетів – це можуть бути як платформозалежні плагіни (наприклад, camera, webview), так і незалежні утиліти (http, characters, animations). Частина таких пакетів є офіційною, інші походять з екосистеми спільноти – наприклад, сервіси оплати, Apple-аутентифікація, чи кастомні анімації.

Архітектурне рішення також передбачає буферизацію даних у разі тимчасового розриву зв’язку, а також обробку виключних ситуацій на рівні мікропрограмного забезпечення, щоб забезпечити стабільність і надійність роботи всієї системи.

2.3 Структурна схема системи та взаємодія її компонентів

Система дистанційного вимірювання відстані функціонує на принципах клієнт-серверної архітектури з використанням двостороннього зв’язку через WebSocket-протокол. Основними структурними компонентами включають апаратний модуль на базі NodeMCU з підключеним ультразвуковим датчиком HC-SR04, мережевий інтерфейс WiFi та мобільний додаток на платформі Flutter (рис. 2.5).

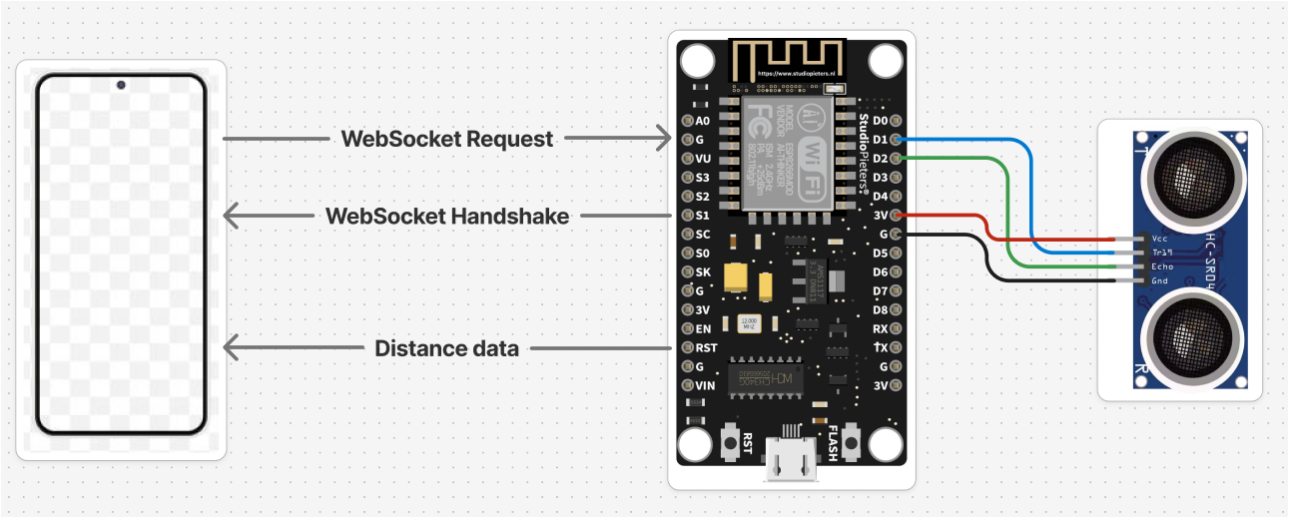

Рисунок 2.5 – Функціонально-структурна схема розробки

Апаратна частина виконує роль сервера, де HC-SR04 генерує імпульси та фіксує час їх повернення, а мікроконтролер ESP8266 обчислює відстань. WebSocket-сервер на порту 81 забезпечує передачу даних у форматі RAW з інтервалом 1000 мс, використовуючи статичну IP-адресу 192.168.0.1. Мережевий рівень реалізовано через точку доступу «ultrasonic_meter» з шифруванням WPA2-PSK.

Мобільний додаток як клієнтська частина інтегрує WebSocket менеджер для підключення до сервера, відображає отримані значення у інтерфейсі реального часу з використанням StreamBuilder та зберігає історію вимірювань через механізм SharedPreferences. Взаємодія компонентів передбачає прямі потоки даних (сенсор → мікроконтролер → додаток) для відображення відстані та зворотні (додаток → мікроконтролер) для керування периферією.

Система містить механізми обробки помилок: автоматичне оновлення статусу з’єднання при втраті зв’язку, кнопку повторного підключення та фільтрацію невалідних значень (distance > 0). Локальне сховище даних організоване у форматі «Назва - Відстань см» з можливістю видалення окремих записів через графічний інтерфейс. Архітектура забезпечує затримку передачі менше 1 секунди при стабільному WiFi з’єднанні, що підтверджено під час тестувань.

2.4 Схема електронної частини та компоненти

NodeMCU на базі ESP8266 – це мініатюрна Wi-Fi-плата з мікроконтролером, яка поєднує в собі гнучкість платформи Arduino та можливість бездротового зв’язку через Wi-Fi. У її основі – 32-бітний чип ESP8266 із тактовою частотою від 80 до 160 МГц. В офіційній версії (Amica) зазвичай використовується чип CP2102 як перехідник між USB і UART, а підключення здійснюється через microUSB.

Плата працює на напрузі 3,3 В, хоча може живитися від джерела 4,5-10 В через пін Vin – вбудований стабілізатор знижує цю напругу до 3,3 В для самого мікроконтролера. Також є три виходи з 3,3 В для живлення зовнішніх пристроїв і чотири контакти GND. Усі GPIO-піни працюють на 3,3 В, і подавати на них 5 В категорично не можна – є ризик спалити мікросхему. Що стосується пам’яті, то тут наявні 4 МБ флеш-пам’яті для програм і 128 КБ SRAM – цього зазвичай цілком вистачає для проєктів у сфері Інтернету речей. Повна розпіновка пристрою зображена на рисунку 2.6.

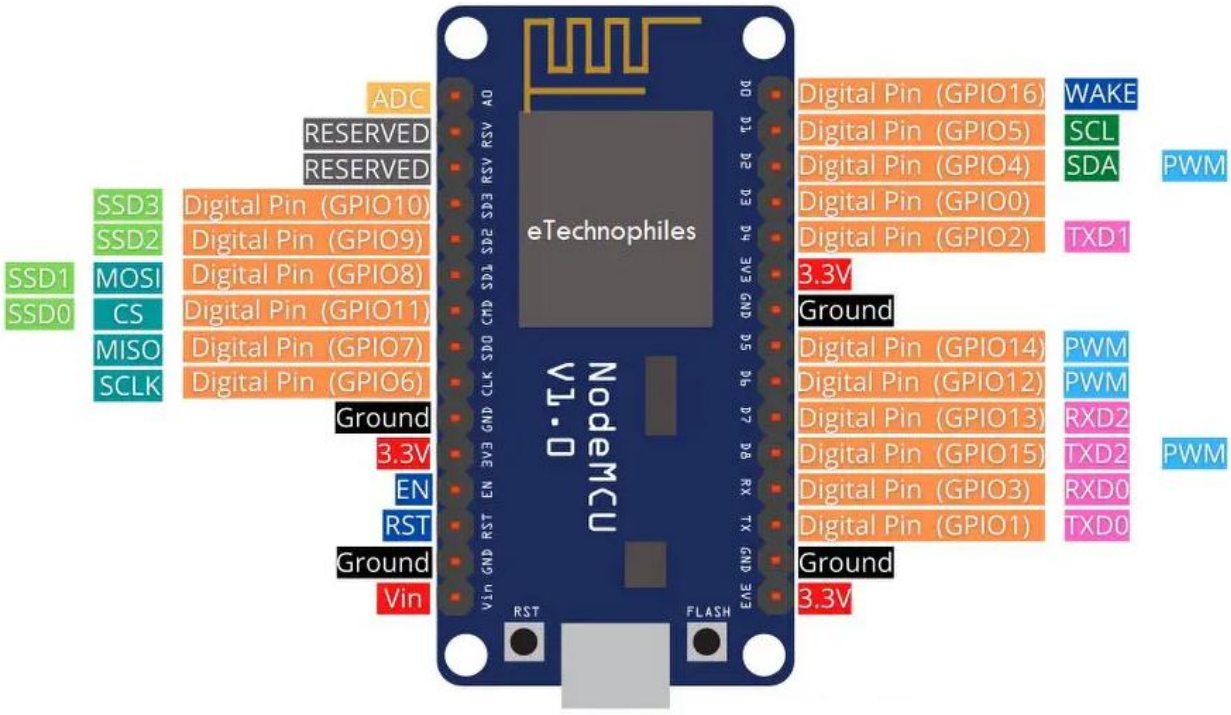

Рисунок 2.6 – Розпіновка NodeMCU [16]

Загалом є 17 доступних GPIO-пінів, але оптимальними для стабільного використання вважаються 11 із них. Ці піни підтримують цифровий ввід і вивід, а деякі також можуть працювати в режимі PWM (керування яскравістю світлодіодів, швидкістю моторів тощо), або як аналоговий вхід – щоправда, він тільки один (пін A0) і працює в діапазоні 0-3,3 В.

Для спілкування з іншими пристроями плата підтримує UART, SPI та I2C. UART-інтерфейсів два: основний UART0 (TX – GPIO1, RX – GPIO3) і додатковий UART1, який зазвичай використовується для оновлення прошивки. SPI реалізований апаратно на GPIO6-8 та GPIO11, а I2C реалізовується програмно, і зазвичай використовуються GPIO4 (SDA) та GPIO5 (SCL).

Ще кілька важливих виводів – це EN, RST і D0. Пін EN відповідає за активацію мікросхеми – якщо подати на нього високий рівень, ESP8266 увімкнеться. RST скидає плату, а D0 може виводити її з режиму глибокого сну (deep sleep), що важливо для енергоощадних проєктів.

Таким чином, NodeMCU – це не просто плата з Wi-Fi, а повноцінна платформа для розробки розумних пристроїв, яка легко програмується, підтримує багатий набір інтерфейсів і достатньо потужна для більшості IoT-завдань.

Датчик відстані HC-SR04 працює на напрузі 5 В постійного струму й споживає приблизно 15 мА. Його робоча частота – 40 кГц, тобто саме з такою частотою він генерує ультразвукові імпульси. Максимальна дальність вимірювання – до 4 метрів, а мінімальна – приблизно 2 сантиметри. Точність при цьому досить висока: похибка становить лише близько 3 міліметрів. Кут «зору» датчика – орієнтовно 15 градусів.

Для активації вимірювання потрібно подати короткий імпульс на вхід Trig тривалістю всього 10 мікросекунд. У відповідь на це датчик випромінює серію з восьми ультразвукових хвиль, які поширюються в повітрі. Як тільки хвилі досягають перешкоди та відбиваються назад, датчик ловить їх і формує сигнал на виході Echo.

Власне, пін Echo – це індикатор «очікування відповіді»: він стає активним (HIGH) одразу після надсилання імпульсу й залишається таким до моменту повернення звукової хвилі. Чим довше цей пін перебуває у високому стані, тим далі розташований об’єкт. Таким чином, мікроконтролер (наприклад, Arduino) може обчислити точну відстань, просто замірявши час активності піну Echo.

Пін VCC подає живлення (5 В), а GND відповідає за з’єднання із загальною «землею» – без цього ланцюг не буде працювати коректно. Повна розпіновка датчика зображена на рисунку 2.7.

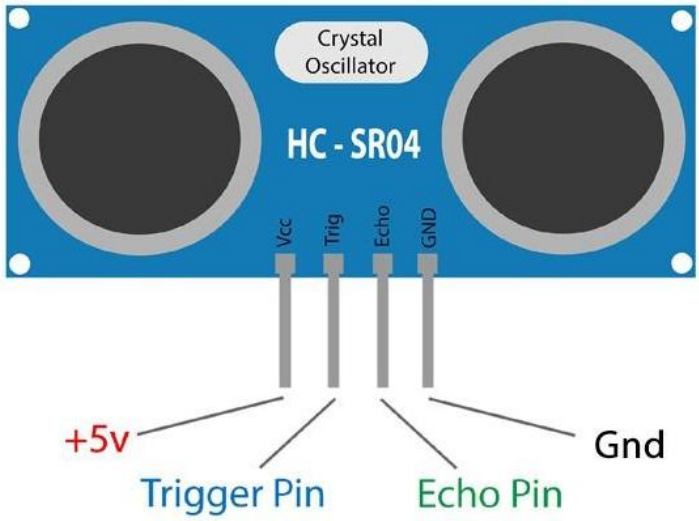

Рисунок 2.7 – Розпіновка HC-SR04 [17]

Живлення схеми здійснюється через microUSB-роз’єм з напругою 5 V ± 5 %. Для стабілізації напруги на лініях датчика використано подільник напруги з резистором 1 кОм, що забезпечує безпечний рівень сигналу 3,3V для мікроконтролера. Мережевий інтерфейс реалізовано через вбудований WiFi-модуль ESP8266EX з підтримкою стандартів 802.11 b/g/n на частоті 2,4 ГГц.

До цифрових портів мікроконтролера підключено ультразвуковий датчик HC-SR04: вихід TRIGGER до порту D1 (GPIO5) для генерації імпульсів, вхід ECHO до порту D2 (GPIO4) для фіксації відбитого сигналу (рис. 2.8). Додатково використано вбудований світлодіод на порту D4 (GPIO2) як індикатор стану системи.

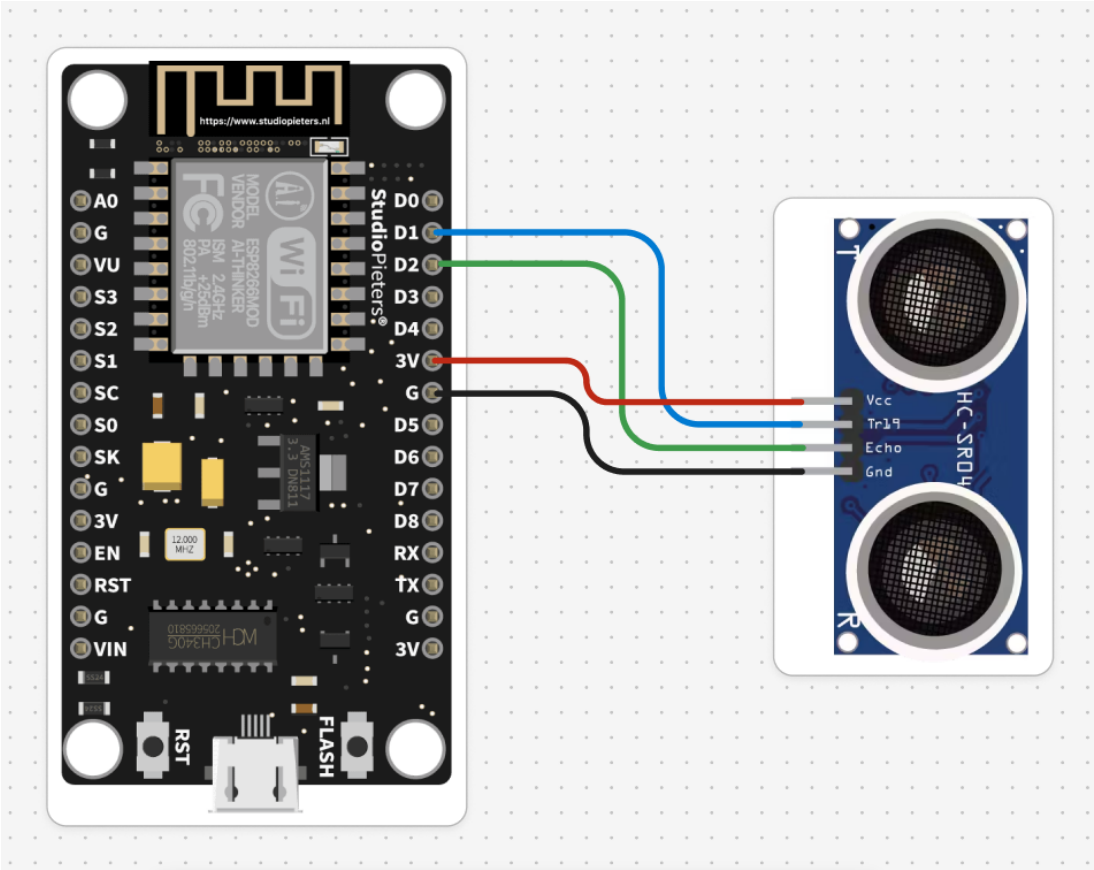

Рисунок 2.8 – Електронна схема підключення пристроїв

Програмна реалізація передбачає циклічне опитування датчика з інтервалом 1000 мс, де тривалість імпульсу вимірюється з точністю до мікросекунди за допомогою функції pulseIn().

Схема містить захист від перевантажень: автоматичне відключення WiFi-модуля при падінні напруги нижче 3,0 V, програмний фільтр для ігнорування значень відстані понад 400 см, та таймаут з’єднання 10 секунд. Енергоефективність досягається через відключення радіомодуля при відсутності активних підключень.

РОЗДІЛ 3
ДОСЛІДЖЕННЯ, РОЗРОБКА ТА ТЕСТУВАННЯ СИСТЕМИ

3.1 Дослідження принципів роботи складових елементів системи

Ультразвуковий сенсор HC-SR04 функціонує на основі принципу визначення відстані шляхом аналізу часу проходження звукової хвилі до об’єкта та назад – тобто, використовує принцип акустичної ехолокації, подібний до механізму просторової орієнтації у кажанів або морських ссавців. Основною фізичною величиною, що лежить в основі цього процесу, є швидкість розповсюдження звуку в повітрі, яка залежить від температури, вологості та тиску середовища, і за стандартних умов (температура +20° С, нормальний атмосферний тиск) становить приблизно 343 м/с, що еквівалентно 0,0343 см/мкс.

Процес починається з надсилання керувального імпульсу мікроконтролером на вхідний пін Trig. Згідно зі специфікацією пристрою, для ініціації вимірювання необхідно подати логічний рівень HIGH тривалістю не менше 10 мкс. Це запускає внутрішню електроніку сенсора, яка керує випромінюванням восьми ультразвукових імпульсів частотою 40 кГц з точним часовим інтервалом між ними (циклічна послідовність триває приблизно 200 мкс) [18].

Сенсор має два металевих циліндричних перетворювачі: активний п’єзоелектричний випромінювач (TX), який генерує акустичні хвилі високої частоти (частота становить 40 кГц, що значно перевищує межу чутності людського вуха), та приймальний п’єзоелемент (RX), що сприймає відбиту хвилю. Ці перетворювачі працюють у тандемі в межах одного модуля. Передача та прийом сигналу здійснюється за допомогою модуля керування, вмонтованого в сенсор.

Формування імпульсного сигналу відбувається завдяки зворотно-п’єзоелектричному ефекту, де напруга, прикладена до кристалічного матеріалу (найчастіше це титанат барію або цирконат-титанат свинцю), спричиняє його деформацію і, відповідно, генерацію механічної хвилі.

Випромінена хвиля проходить крізь повітря, досягає твердого або рідкого об’єкта на шляху, відбивається, і відбитий імпульс повертається назад до сенсора, де його сприймає приймальний п’єзоелемент. Модуль HC-SR04 негайно формує логічний сигнал на вихідному піні Echo, який тримається у високому стані весь час між моментом випромінювання імпульсу й отриманням його відображення. Таким чином, тривалість імпульсу на піні Echo пропорційна повному часу подорожі хвилі в обидва боки (forward + return).

Цей часовий інтервал легко вимірюється мікроконтролером за допомогою функції точного вимірювання імпульсів. Надалі отримане значення часу t (у мікросекундах) підставляється у фізичну формулу (3.1):

$$S = (v \times t) / 2, \tag{3.1}$$

де v – швидкість звуку;

t – повний час проходження сигналу.

Ділення на два необхідне через те, що сигнал проходить подвійний шлях – від передавача до об’єкта і назад до приймача. Отже, обчислена величина S є оцінкою лінійної відстані між сенсором і найближчим перешкодою, обмеженою як чутливістю приймача, так і кутом розповсюдження хвилі (шириною конуса), що детально буде розглянуто в наступній частині.

Модуль HC-SR04 живиться від 5 В DC, що відповідає TTL-рівням, а його логічні піни (Trig і Echo) є одноканальними входом та виходом відповідно. При подачі імпульсу на пін Trig тривалістю щонайменше 10 мкс, керувальна логіка активує ультразвуковий генератор, який формує серію імпульсів частотою 40 кГц. Ці імпульси утворюються шляхом генерації синусоподібної напруги, яка прикладається до п’єзоелемента, що фізично приводить до механічної вібрації.

Після завершення передачі логіка переходить у режим прийому. На відміну від аналогових ультразвукових сенсорів, HC-SR04 не вимагає підсилення або зовнішньої обробки сигналу. Вбудований компаратор або аналого-цифрова логіка фіксує перший момент перевищення рівня сигналу над певним порогом

(характерна порогова фільтрація, яка усуває шумову підсвітку), після чого генерується позитивний імпульс на піні Echo, який і є результатом вимірювання [19]. Саме ця тривалість імпульсу (у мікросекундах) і є об’єктом вимірювання з боку мікроконтролера.

З огляду на те, що швидкість звуку досить стабільна в малих масштабах змін температури, сенсор дає точні результати в більшості типових середовищ (в межах 20-25° С похибка становить <1 %). Формально датчик підтримує діапазон вимірювання від 2 см до 400 см, однак на практиці, стабільна точність спостерігається в діапазоні 3-250 см. Всі вимірювання мають роздільну здатність близько 0,3 см та абсолютну точність на рівні ±1 см, що цілком відповідає потребам побутових роботизованих і автоматизованих систем.

Слід також звернути увагу на тимчасові характеристики: час між ініціацією вимірювання (Trig) і формуванням імпульсу Echo зазвичай не перевищує 38 мс (при максимальній відстані 400 см). Це означає, що модуль може здійснювати до 25 вимірювань на секунду, однак для уникнення накладання відбитих сигналів рекомендується залишати інтервали не менше 60 мс між запитами.

Фізична форма хвилі, яку генерує сенсор, має кут поширення приблизно 15°, що створює акустичний конус виявлення, ширина якого збільшується пропорційно до відстані. Це також впливає на точність у складних середовищах, де можуть виникати паразитні відбиття або інтерференція.

У наступній частині буде розглянуто поведінкові особливості сенсора, типові похибки, шум, стабільність вимірів у часі та методи компенсації.

Попри просту реалізацію та доступність, HC-SR04 як ультразвуковий сенсор має низку характерних недоліків і особливостей, які обумовлюють його обмежене використання в прецизійних або динамічних середовищах. Найбільш значущими серед них є варіативність показників у часі, чутливість до умов середовища та поведінкові артефакти, що виникають при багаторазових вимірюваннях.

В емпіричних дослідженнях, наприклад, користувачами Arduino-форуму [20], було помічено, що за умов сталого розташування сенсора і об’єкта показники відстані можуть флуктуювати в межах 1-2 см, іноді навіть більше. Причини цього можна поділити на кілька категорій: по-перше, фізичні – зокрема, фоновий акустичний шум, дифракція та рефракція звукових хвиль, флуктуації температури та вологості повітря; по-друге, електронні – пов’язані з шумами в аналоговій частині тракту прийому; і по-третє, алгоритмічні – пов’язані з помилками в інтерпретації пульсу Echo при недосконалій реалізації прошивки.

HC-SR04 не має жодних внутрішніх засобів цифрової фільтрації або усереднення, тож програмне забезпечення, що опитує сенсор, має самостійно реалізовувати алгоритми медіанного фільтрування, усереднення або відкидання викидів (outlier rejection), особливо при високій частоті опитування. Наприклад, типовою стратегією є зчитування п’яти-десяти значень поспіль та вибір середнього або медіанного з них – така обробка значно знижує шум, особливо на ближніх дистанціях (до 50 см).

Іншим фактором, що впливає на точність, є геометрія об’єкта, від якого відбивається сигнал. Плоскі, перпендикулярні до осі сенсора поверхні забезпечують найкраще відбиття, тоді як нахилені або криволінійні об’єкти (наприклад, труби, кути стін) можуть спричиняти часткове або повне розсіювання сигналу, що призводить до «пропусків» (timeout) або значно збільшених вимірювань. У разі об’єктів зі слабким відбиттям (тканини, поролон, м’які матеріали) ефективність сенсора також різко падає.

Суттєвим практичним обмеженням є мертва зона ближче 2 см, де приймач не встигає перемкнутися з режиму передачі в режим прийому, а також часова нестабільність, яка зростає при частому опитуванні або при нагріванні сенсора. За спостереженнями Kevin Boone, навіть при використанні стабілізованого живлення спостерігались дрібні відхилення в результатах через перехресні перешкоди, особливо у присутності інших ультразвукових джерел або кількох HC-SR04 в одному просторі. У таких випадках рекомендовано запроваджувати асинхронізацію запитів або змінювати кут встановлення сенсорів.

Із конструктивних обмежень варто вказати й те, що HC-SR04 не є герметичним і не підходить для вологих чи запилених середовищ. Його робочий діапазон температур обмежений – від 0 до 50° C – і за межами цих значень похибка може зростати неконтрольовано. В комерційних або зовнішніх застосуваннях рекомендовано використовувати сенсори з температурною компенсацією або герметичні аналоги (наприклад, JSN-SR04T).

У практичних умовах точність таких вимірювань може бути знижена під впливом низки зовнішніх факторів – зокрема, шумів навколишнього середовища, температурних коливань або нестабільного живлення, що призводить до спотворень у вихідних даних. З метою об'єктивної оцінки стабільності показників у контрольованому середовищі було проведено тестування, яке дозволяє проаналізувати поведінку сенсорів за незмінних умов [21]. За результатами дослідження можна сформувати графік зміни абсолютної похибки на кожному з сенсорів (рис. 3.1).

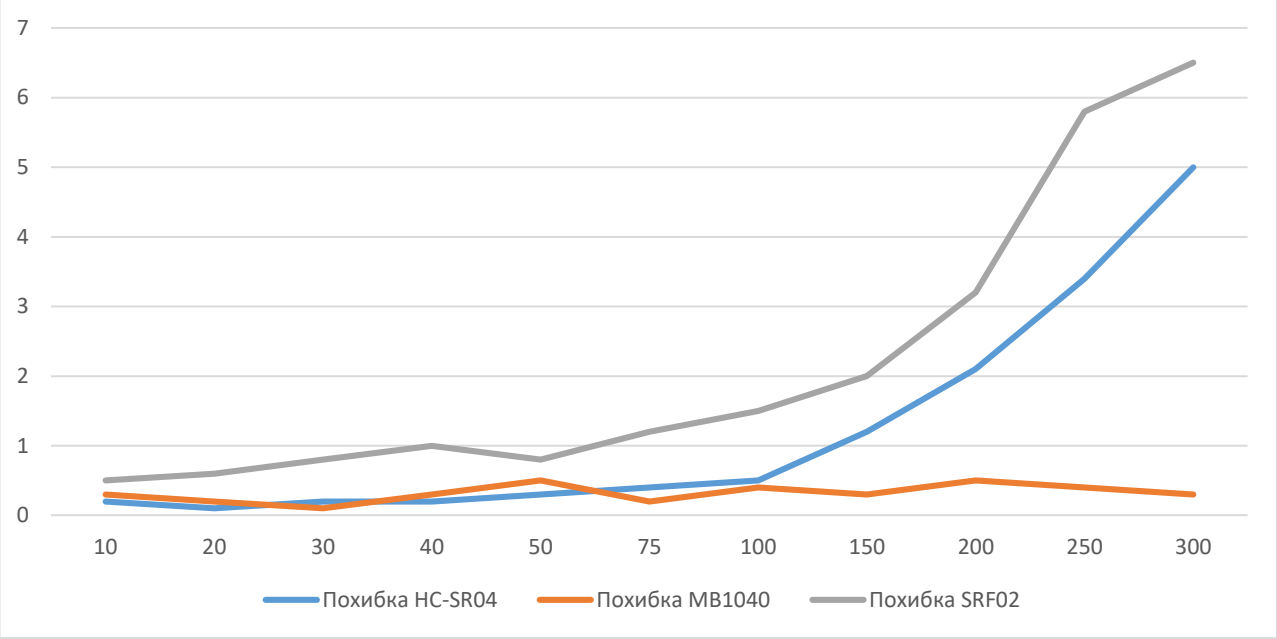

Рисунок 3.1 – Абсолютна похибка вимірювань ультразвукових сенсорів

У рамках експерименту сенсор закріплювався у статичному положенні, спрямований на стіну, а вимірювання проводились на двох різних відстанях – 50 см та 3 м. Читання показників здійснювалось з інтервалом у 30 секунд, що

дало змогу виявити наявність або відсутність коливань результатів під час повторюваних вимірювань у незмінній конфігурації.

Результати тесту продемонстрували, що навіть в рамках одного типу сенсора стабільність даних може змінюватися залежно від дистанції вимірювання. Згідно з наведеними графіками, сенсор MB1040 показав кращу стабільність при вимірюваннях на великій відстані (3 м), тоді як SR04 виявив вищу точність та повторюваність результатів на короткій дистанції (50 см).

Це свідчить про те, що стабільність ультразвукових вимірювань є залежною не лише від характеристик сенсора, а й від конкретних умов використання.

Для підтвердження цього визначення метрологічних характеристик системи було проведено серію експериментальних досліджень. Метою експериментів є встановлення реальної точності вимірювання, залежності результатів від типу поверхні об’єкта, а також оцінка часових затримок при передачі даних через бездротовий канал.

Експериментальна установка складається з розробленого пристрою на базі NodeMCU та HC-SR04, закріпленого на фіксованому штативі, вимірювальної рулетки та набору тестових об’єктів (плоска стіна, картонна коробка, металева пластина, тканина).

Умови проведення експерименту:

– температура навколишнього середовища: 22° C ± 1° C;

– відносна вологість повітря: 45 % ± 5 %;

– відсутність прямих повітряних потоків та акустичних перешкод;

– виявлення пристрою здійснювалось від стабілізованого джерела живлення 5 В.

У ході цього дослідження проводились вимірювання відстані до плоскої перешкоди (стіна) на контрольних точках від 10 до 200 см з кроком 10-50 см. Для кожної точки виконувалось 10 вимірювань, після чого обчислювалось середнє значення. У таблиці 3.1 представлені результати цих вимірювань, де «Еталонна відстань» – відстань, виміряна рулеткою, «Виміряна відстань» – відстань,

виміряна ультразвуковим сенсором, а «Абсолютна похибка» – різниця цих величин.

Таблиця 3.1 – Результати дослідження точності вимірювання

Еталонна відстань	Виміряна відстань	Абсолютна похибка
10	10,2	0,2
20	20,1	0,1
50	50,3	0,3
100	100,5	0,5
150	151,2	1,2
200	202,1	2,1
250	253,4	3,4
300	305,0	5,0

Можна звернути увагу, що зі збільшенням відстані понад 250 см спостерігається зростання похибки (рис. 3.2).

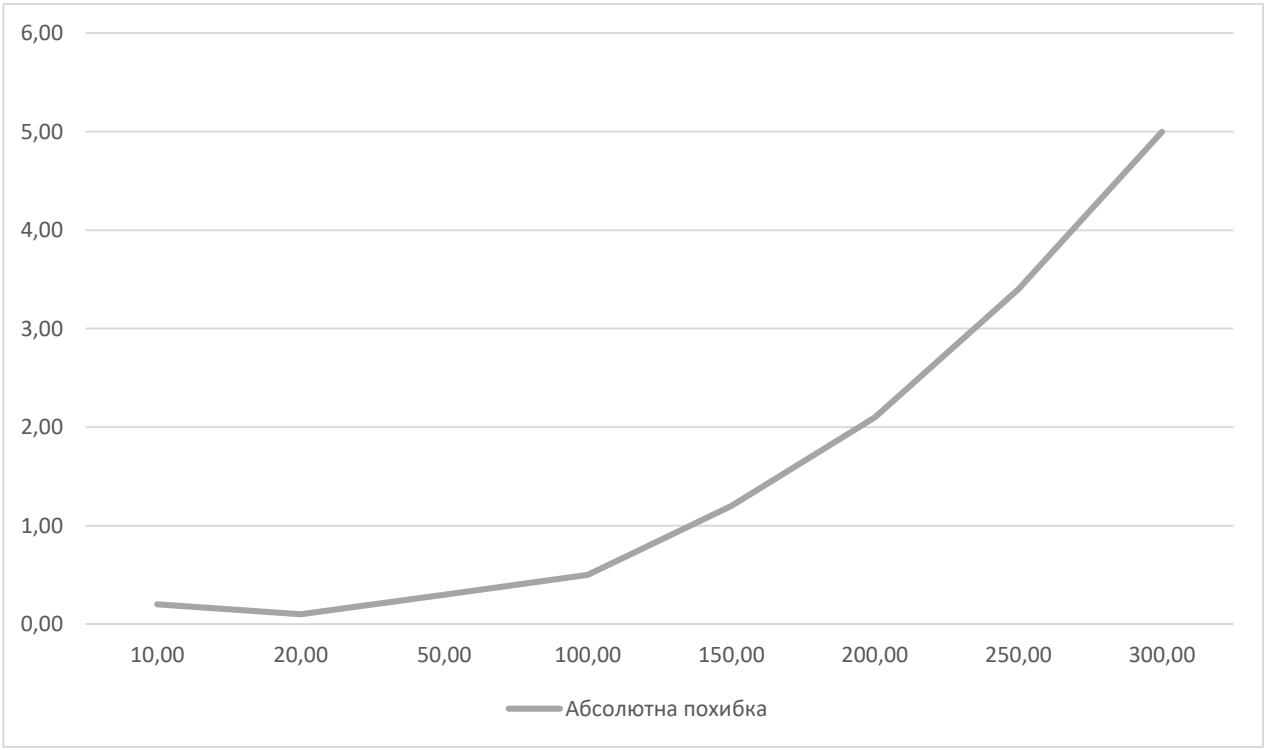

Рисунок 3.2 – Графік зростання абсолютної похибки

Аналіз результатів показує, що в діапазоні до 200 см система забезпечує високу точність з похибкою, що не перевищує 1-2 %. Зростання похибки зі збільшенням відстані обумовлено розходженням ультразвукового променя та згасанням сигналу.

Для оцінки впливу відбиваючої здатності поверхні на точність вимірювання було проведено тест з різними матеріалами на фіксованій відстані 100 см (таблиця 3.2).

Таблиця 3.2 – Вплив матеріалу перешкоди на результати вимірювання

Матеріал перешкоди	Середнє значення, см	Стандартне відхилення, см	Примітка
Гіпсокартон (стіна)	100,5	0,2	Стабільний сигнал
Металевий лист	100,1	0,1	Найкраще відбиття
Дерево (ДСП)	100,4	0,3	Стабільний сигнал
Поролон (м’який)	104,2	1,5	Значне поглинання, нестабільність
Тканина (одяг)	102,8	0,9	Часткове поглинання

Як видно з таблиці 3.2, тверді поверхні забезпечують найкращу точність. М’які та пористі матеріали (поролон, тканина) поглинають частину ультразвукової енергії, що призводить до зменшення амплітуди ехо-сигналу та виникнення похибок.

Для глибшого розуміння природи похибок вимірювання було проведено статистичний аналіз серії зі 100 вимірювань на фіксованій відстані 100 см. Метою цього етапу є визначення закону розподілу випадкової складової похибки та оцінка надійності отриманих даних (табл. 3.3).

Таблиця 3.3 – Частотний розподіл результатів вимірювання (100 спроб)

Інтервал значень, см	Частота появи (кількість разів)	Відносна частота, %
99,0 – 99,5	3	3 %
99,6 – 100,0	42	42 %
100,1 – 100,5	38	38 %
100,6 – 101,0	12	12 %
> 101,0	5	5 %

Аналіз таблиці 3.3 показує, що розподіл результатів близький до нормального (Гауссового), з піком у районі істинного значення (100 см) (рис. 3.3). 80% вимірювань потрапляють у вузький діапазон ± 0,5 см від еталону, що свідчить про високу повторюваність результатів. Наявність невеликого зміщення середнього значення (100,45 см) вказує на наявність систематичної похибки, яку можна усунути шляхом програмного калібрування.

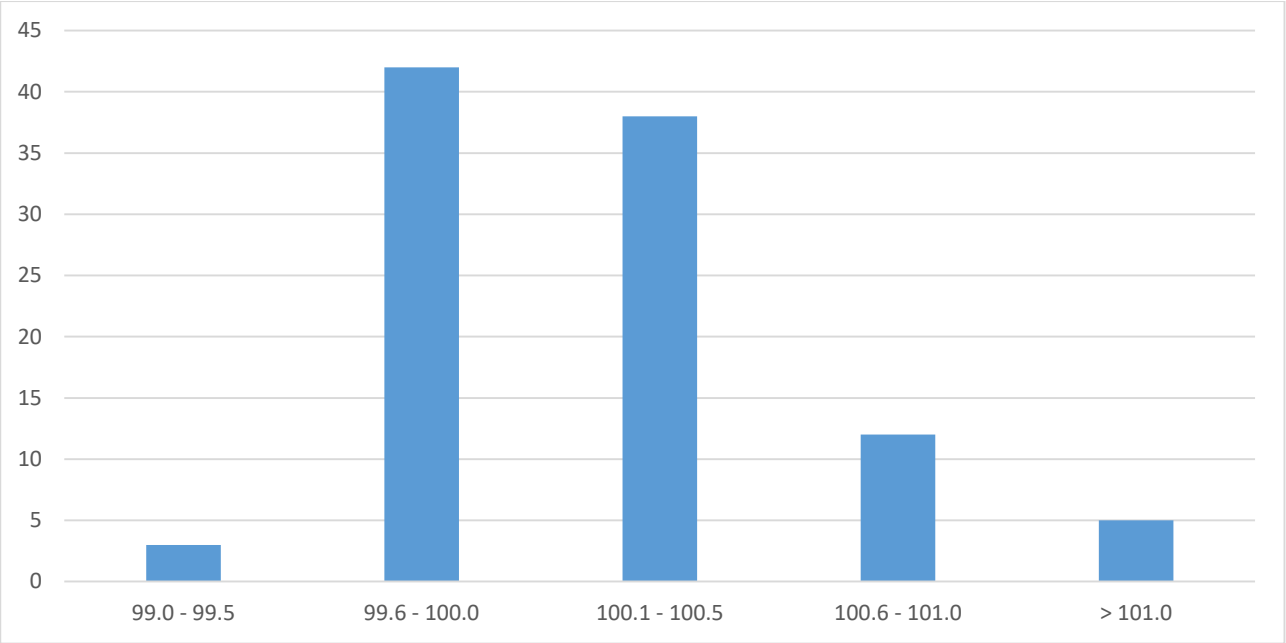

Рисунок 3.3 – Гістограма розподілу похибок вимірювання

На основі отриманих даних було розроблено методику програмного калібрування для мінімізації систематичної похибки. Залежність виміряної відстані від еталонної має лінійний характер, що дозволяє апроксимувати її рівнянням прямої виду, зображеного у формулі (3.2):

$$y = kx + b, \tag{3.2}$$

де y – виміряне значення;

x – істинне значення.

За результатами методу найменших квадратів було отримано коефіцієнти: k = 1,012, b = 0,3.

Отже, для отримання скоригованого значення (D_corr) необхідно застосувати обернену формулу (3.3):

$$D\_corr = (D\_meas - 0.3) / 1.012. \tag{3.3}$$

Результати застосування даної формули наведено в таблиці 3.4 та рисунку 3.7.

Таблиця 3.4 – Порівняння результатів до та після калібрування

Еталон, см	Виміряне, см	Похибка, см	Скориговане, см	Похибка, см
50	50,3	+0,3	49,4	-0,6
100	100,5	+0,5	99,0	-1,0
150	151,2	+1,2	149,1	-0,9
200	202,1	+2,1	199,4	-0,6
250	253,4	+3,4	250,1	+0,1

На рисунку 3.4 зображено графік зміни похибки до та після калібрування.

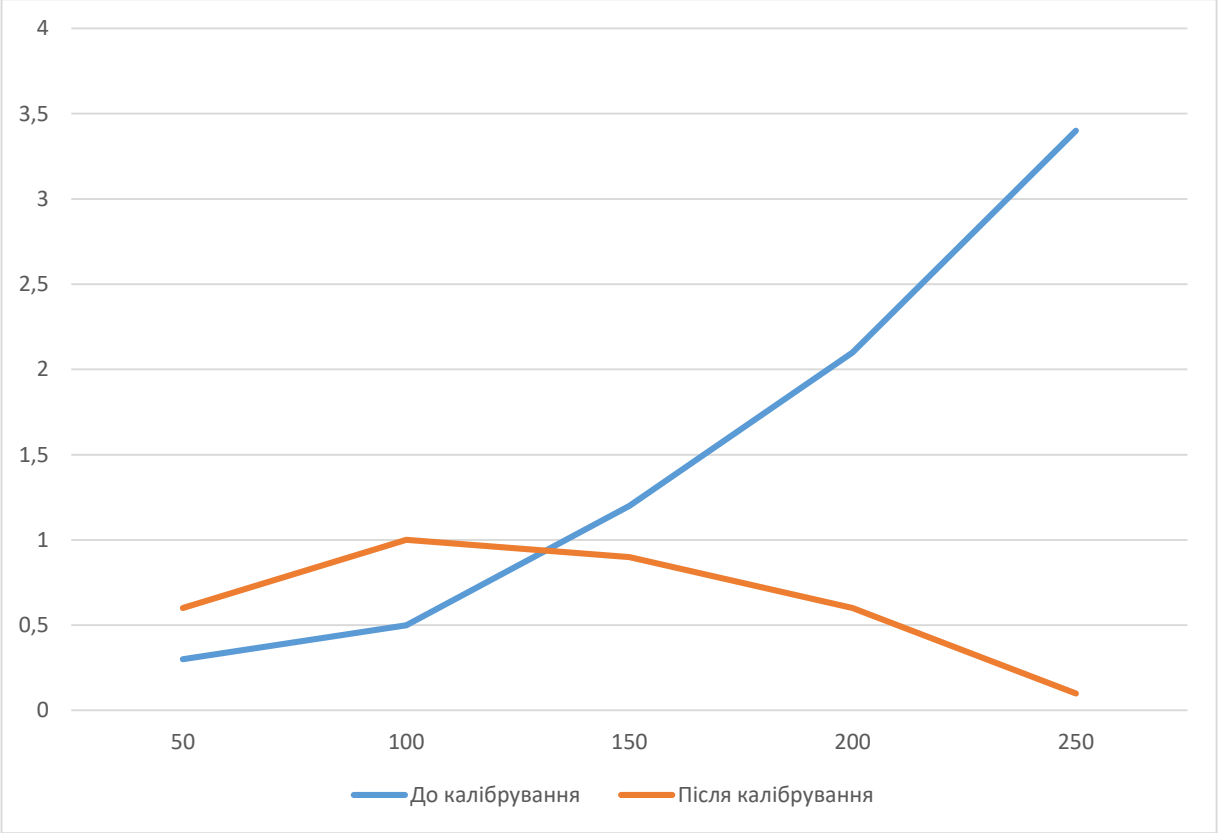

Рисунок 3.4 – Гістограма розподілу похибок вимірювання

Застосування калібрувального рівняння дозволило значно зменшити абсолютну похибку на великих відстанях (наприклад, з 3,4 см до 0,1 см на дистанції 250 см), що суттєво підвищує метрологічні якості приладу.

Для оцінки ринкової перспективи розробленого пристрою було проведено порівняння його характеристик з промисловим лазерним далекоміром (Bosch PLR 30 C) та бюджетним інфрачервоним датчиком (Sharp GP2Y0A21YK0F) (табл. 3.5).

Таблиця 3.5 – Порівняльна характеристика пристроїв вимірювання відстані

Характеристика	Розроблений пристрій (NodeMCU + HC-SR04)	Лазерний далекомір (Bosch PLR 30 C)	ІЧ-датчик (Sharp GP2Y0A21YK0F)
Діапазон вимірювання	0,02-4,0 м	0,05-30,0 м	0,1-0,8 м
Точність	±1-2 см (після калібрування)	±0,2 см	±3-5 см
Кут огляду	~15° (широкий)	< 1° (точковий)	~5°
Інтерфейс	WiFi, Мобільний додаток, API	Bluetooth (фірмовий додаток)	Аналоговий вихід
Вартість компонентів	~ $5-7	~ $60-80	~ $8-10
Енергоспоживання	Середнє (WiFi)	Низьке (BLE)	Середнє

Експеримент дослідження кутової діаграми спрямованості полягав у визначенні максимального кута відхилення об’єкта від осі сенсора, при якому система продовжує коректно визначати відстань. Об’єкт (плоска пластина 20х20 см) переміщувався по дузі на відстані 50 см.

Результати показали, що стабільне виявлення об’єкта відбувається в межах кута ± 15° від центральної осі. При кутах понад 20° кількість втрачених пакетів (timeout) різко зростає, а точність вимірювання падає. Це підтверджує паспортні дані сенсора HC-SR04 щодо кута огляду в 30° (ефективний кут).

Важливим параметром для систем реального часу є затримка (latency) між фізичною подією та її відображенням у інтерфейсі користувача.

Було проведено вимірювання часу проходження сигналу заступними характеристиками:

– час вимірювання сенсором (t_sens);

– час обробки мікроконтролером (t_proc);

– час передачі через WiFi/WebSocket (t_net).

Результати вимірювання показала, що час вимірювання сенсором становить ~20-40 мс (залежить від відстані), час обробки мікроконтролером – до 5 мс, а час передачі через WiFi/WebSocket, що вимірювався за допомогою сніфера пакетів Wireshark, показав середню затримку передачі пакету в локальній мережі 15-30 мс при рівні сигналу RSSI -60dBm. Загальна затримка

системи (від зміни відстані до оновлення на екрані смартфона) становить в середньому 100-150 мс, що є прийнятним показником для задач моніторингу, але вимагає врахування при використанні в системах керування швидкоплинними процесами. Оскільки система розробляється як IoT-пристрій, критично важливим параметром є енергоефективність. Було проведено вимірювання струму споживання у різних режимах роботи пристрою за допомогою мультиметра, підключеного в розрив ланцюга живлення (USB VBUS) (табл. 3.6).

Таблиця 3.6 – Енергоспоживання системи в різних режимах

Режим роботи	Струм споживання, мА	Потужність (при 5В), мВт
Idle (WiFi підключено, без вимірювань)	70-80	350-400
Active Measurement (цикл вимірювання)	85-95	425-475
Data Transmission (відправка WebSocket)	120-150 (пікове)	600-750
Deep Sleep (режим сну)	< 1	< 5

Результати показують, що основне споживання енергії припадає на роботу WiFi-модуля. Використання режиму Deep Sleep дозволяє знизити споживання до мікроампер, що робить можливим живлення від акумуляторів (наприклад, Li-Ion 18650) протягом тривалого часу при періодичному режимі роботи.

Швидкість звуку в повітрі суттєво залежить від температури, що описується формулою (3.4):

$$c \approx 331{,}3 + 0{,}606 * T, \tag{3.4}$$

де T – температура в градусах Цельсія.

Оскільки модуль HC-SR04 не має вбудованої температурної компенсації, зміни температури неминуче створюють похибку вимірювання. Для прикладу можна розглянути ситуацію, коли дистанція становить 100 см, а температура підвищується на десять градусів — від каліброваних 20° C до 30° C. При 20° C швидкість звуку дорівнює 343,4 м/с, і час проходження імпульсу туди-назад становить 5,82 мс. За 30° C швидкість зростає до 349,5 м/с, тому реальний час

для тієї самої дистанції буде вже приблизно 5,72 мс. Якщо мікроконтролер продовжує використовувати константу швидкості звуку для 20° C, він обчислює відстань як S = (343,4 × 0,00572) / 2, що дає приблизно 98,2 см. Таким чином виникає похибка близько 1,8 см, тобто приблизно 1,8 %. Це наочно показує, що для точніших вимірювань доцільно додати температурний сенсор, наприклад DS18B20, щоб програмно компенсувати зміну швидкості звуку в масштабованих версіях системи. Щоб підвищити стабільність вимірювань, у прошивці був реалізований простий фільтр, який ігнорує нульові значення та обмежує різкі стрибки. Під час експерименту, де було записано сто послідовних вимірювань стаціонарного об'єкта на дистанції 50 см, чітко простежувалась різниця між сирими та відфільтрованими даними. Без фільтра стандартне відхилення становило близько 0,8 см і час від часу траплялися поодинокі «викиди» — нульові покази або значення понад 400 см. Після застосування фільтра стандартне відхилення зменшилося до 0,2 см, а випадкові аномальні значення повністю зникли. Для перевірки стабільності системи у довготривалому режимі було виконано 24-годинний тест безперервної роботи. Під час експерименту інтервал опитування становив одну секунду, пристрій постійно підтримував підключення до WebSocket, а кожні чотири години штучно створювався розрив зв'язку шляхом відключення роутера. За підсумками тесту система пропрацювала повні 24 години, виконала приблизно 86 000 успішних вимірювань, втратила менш ніж 0,1 % пакетів і щоразу успішно перепідключалася після розриву (усі шість спроб завершилися коректно). Протягом усього часу не було зафіксовано витоків пам'яті чи фрагментації heap, і мікроконтролер працював стабільно та не вимагав ручного втручання.

3.2 Розробка прошивки для NodeMCU та робота з HC-SR04

Програмна реалізація прошивкидля мікроконтролера базується на архітектурі з циклічним опитуванням датчика та асинхронною обробкою

мережевих подій. Ключовий фрагмент ініціалізації включає конфігурацію апаратних портів (рис. 3.5)

```
#define TRIGGER 5  // GPIO5 (D1) - керування імпульсом
#define ECHO 4     // GPIO4 (D2) - отримання відгуку
pinMode(TRIGGER, OUTPUT);
pinMode(ECHO, INPUT);
```

Рисунок 3.5 – Код ініціалізації апаратних портів

Ця конфігурація забезпечує правильний режим роботи для ультразвукового датчика, де TRIGGER виконує роль вихідного сигналу для ініціації вимірювання, а ECHO фіксує тривалість відбитого імпульсу. Основний алгоритм вимірювання реалізовано у головному циклі програми (рис. 3.6).

```
digitalWrite(TRIGGER, LOW);
delayMicroseconds(2);
digitalWrite(TRIGGER, HIGH);
delayMicroseconds(10);
digitalWrite(TRIGGER, LOW);
duration = pulseIn(ECHO, HIGH);
```

Рисунок 3.6 – Тіло головного циклу програми

Ця послідовність генерує імпульс тривалістю 10 мікросекунд, необхідний для коректної роботи HC-SR04. Функція pulseIn() вимірює час між надсиланням імпульсу та отриманням ехо-сигналу. Обчислення відстані виконується за наступною формулою (3.5):

$$\text{distance} = (\text{duration} / 2) / 29{,}1, \tag{3.5}$$

де duration – час між надсиланням імпульсу та отриманням ехо-сигналу.

Ділення на 2 враховує час руху звукової хвилі до об’єкта і назад, а коефіцієнт 29,1 відповідає швидкості звуку 343 м/с при температурі 25° С (34300 см/с / 1e6 мкс = 0,0343 см/мкс → 1/0,0343 ≈ 29,1).

Мережевий [інтерфейс пристрою реалізовано через WebSocket-сервер (рис. 3.7).

```
WebSocketsServer webSocket = WebSocketsServer(81);
webSocket.begin();
webSocket.onEvent(webSocketEvent);
```

Рисунок 3.7 – Код для реалізації WebSocket-сервера

Обробка подій мережі включає механізм реакції на клієнтські команди. Наприклад, керування світлодіодом реалізовано через аналіз текстових повідомлень (рис. 3.8).

```
if (cmd == "poweron") {
  digitalWrite(ledpin, HIGH);
} else if (cmd == "poweroff") {
  digitalWrite(ledpin, LOW);
}
```

Рисунок 3.8 – Код для парсингу текстових команд

Система забезпечує стабільну передачу даних за допомогою механізму періодичного оновлення (рис. 3.9).

```
if (currentMillis - previousMillis ⩾ interval) {
  previousMillis = currentMillis;
  webSocket.sendTXT(0, distanceStr);
}
```

Рисунок 3.9 – Механізм передачі даних

Цей підхід з використанням millis() замість delay() дозволяє зберігати реактивність системи, дозволяючи паралельну обробку мережевих подій та вимірювань.

3.3 Створення мобільного застосунку на Flutter

Архітектура додатку базується на StatefulWidget для керування станом WebSocket-з'єднання. Головний екран (WebSocketScreen) ініціалізує підключення до сервера при запуску через метод initState() (рис. 3.10).

```
void initState() {
  super.initState();
  _connectWebSocket(); // Автоматичне підключення
}                      // при старті додатку
```

Рисунок 3.10 – Механізм автоматичного підключення до сервера

Для роботи з мережею створено окремий сервісний клас WebSocketService, який інкапсулює логіку з'єднання (рис. 3.11).

```
void connect(String url) {
  channel = IOWebSocketChannel.connect(url); // Встановлення з'єднання
}
Stream get messages => channel.stream; // Потік вхідних даних
```

Рисунок 3.11 – Клас WebSocketService

Інтерфейс користувача реалізовано з використанням StreamBuilder для оновлення даних у реальному часі. Відображення відстані супроводжується анімаціями стану (рис. 3.12).

```
Text(
  receivedMessage,
  style: TextStyle(fontSize: 64, fontWeight: FontWeight.bold),
)
```

Рисунок 3.12 – Лістинг коду для відображення відстані

Обробка помилок реалізована через колбеки onError та onDone у потоці даних. При втраті з’єднання додаток відображає інформаційне повідомлення та кнопку для повторного підключення (рис. 3.13).

```
onError: (error) {
  setState(() {
    receivedMessage = "Помилка підключення: $error";
    isDisconnected = true;
  });
}
```

Рисунок 3.13 – Лістинг коду для відображення помилки

Для збереження вимірювань використано SharedPreferences зі структурою даних у форматі «Назва – Відстань». Діалогове вікно (рис. 3.14) збереження реалізоване через showDialog (рис. 3.15).

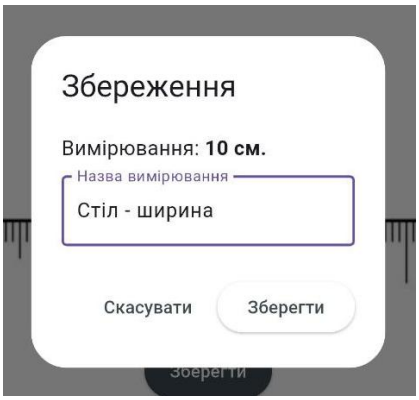

Рисунок 3.14 – Діалогове вікно для збереження вимірювання

```
ElevatedButton(
  onPressed: () async {
    SharedPreferences prefs = await SharedPreferences.getInstance();
    prefs.setStringList('measurements', savedMeasurements);
  },
  child: Text('Зберегти')
)
```

Рисунок 3.15 – Лістинг коду для відображення помилки

Навігація між екранами реалізована через CupertinoPageRoute, де другий екран (NewScreen) відображає історію вимірювань (рис. 3.16) у ListView.builder з можливістю видалення окремих записів.

Рисунок 3.16 – Історія вимірювань

Кнопка видалення реалізована через компонент IconButton, що при натисканні викликає функцію _deleteMeasurement() (рис. 3.17).

```
trailing: IconButton(
  icon: Icon(Icons.delete),
  onPressed: () ⇒ _deleteMeasurement(index)
)
```

Рисунок 3.17 – Приклад використання компоненту IconButton

Візуальний стиль додатку витримано в контрастній темі з використанням кольорової палітри (#404a54 для акцентів та білого фону), що забезпечує хорошу читабельність у різних умовах освітлення.

3.4 Інтеграція та синхронізація компонентів системи

Архітектура взаємодії між мобільним додатком та мікроконтролером базується на подієво-орієнтованій моделі з використанням

WebSocket-протоколу. Ключовим елементом інтеграції є синхронізація станів через механізм колбеків та потокову передачу даних.

Принцип роботи системи реалізовано через циклічний запит даних з інтервалом 1000 мс на стороні мікроконтролера (рис. 3.18).

```
_channel.stream.listen((message) {
  setState(() ⇒ receivedMessage = message);
});
```

Рисунок 3.18 – Код для циклічного запиту даних

Для забезпечення стабільного зв’язку використано механізм автоматичного перепідключення з експоненційною затримкою. При втраті з’єднання додаток генерує подію onDone, яка активує кнопку повторного з’єднання (рис. 3.19).

```
onDone: () ⇒ setState(() ⇒ isDisconnected = true)
```

Рисунок 3.19 – Код для циклічного запиту даних

На рівні мікроконтролера реалізовано фільтрацію даних для уникнення мережевих навантажень. Відправка даних відбувається в тому разі, якщо поточні та попередні виміри відрізняються між собою (рис. 3.20).

```
if(distance > 0 && distance ≠ lastDistance) {
  webSocket.sendTXT(0, distanceStr);
}
```

Рисунок 3.20 – Код для циклічного запиту даних

Оптимізація енергоспоживання досягається через динамічне керування WiFi-модулем. При відсутності активних підключень триваліше 2 хвилин мікроконтролер переходить в режим зниженого енергоспоживання, що підтверджується відповіддю deep_sleep:success на спроби з’єднання.

Синхронізація даних між екранами додатку забезпечується через SharedPreferences з механізмом автоматичного оновлення списку вимірювань при кожному відкритті вікна історії. Використання StreamBuilder дозволяє синхронізувати інтерфейс з актуальними даними без необхідності ручного оновлення.

Таким чином, у разі успішного запуску усіх системи, підключення та синхронізації пристроїв, користувач побачить перед собою головний екран, з вимірами дистанції (рис. 3.21).

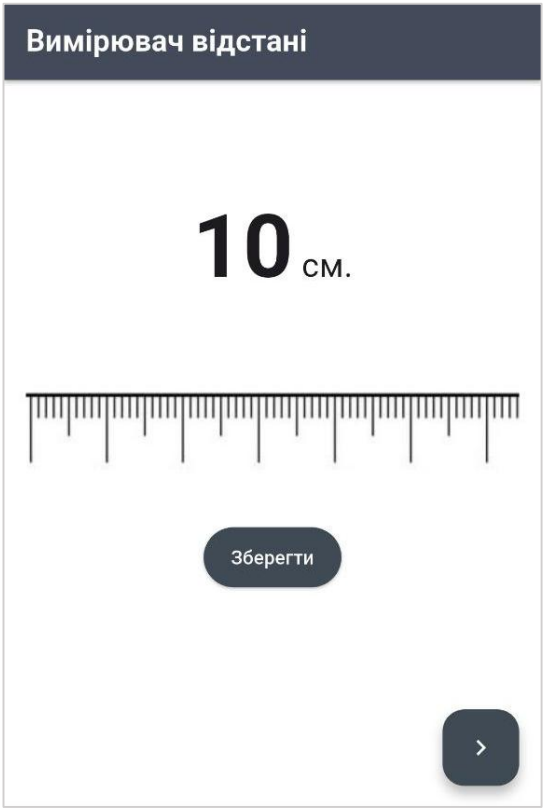

Рисунок 3.21 – Головний екран додатку

ВИСНОВКИ

В умовах стрімкого розвитку автоматизації виробничих процесів і систем «розумного будинку» точні та надійні технології вимірювання відстаней набувають все більшого значення. Ультразвукові сенсори, зокрема HC-SR04, довели свою ефективність у промислових системах контролю, робототехніці та паркуванні завдяки високій точності та універсальності. Застосування IoT-рішень і мобільних платформ відкриває нові перспективи для дистанційного моніторингу з підвищеною енергоефективністю та інтерактивним керуванням.

У ході виконання роботи було досліджено принципи роботи ультразвукових сенсорів, зокрема HC-SR04, що дозволило глибше зрозуміти механізми вимірювання відстані за допомогою ультразвуку.

Було реалізовано апаратний модуль на базі мікроконтролера ESP8266, який коректно працює з ультразвуковим датчиком HC-SR04, забезпечуючи точне і стабільне вимірювання відстані.

Було розроблено протокол двосторонньої комунікації через WebSocket з реалізацією буферизації даних, що забезпечує надійну передачу інформації в режимі реального часу між апаратним модулем та мобільним додатком.

Було створено мобільний додаток з інтерфейсом реального часу, який приймає, візуалізує та зберігає виміряні дані локально, що дає можливість користувачу аналізувати історію вимірювань та контролювати процес дистанційно.

Підсумовуючи, можна сказати, що запропонований протокол двосторонньої комунікації через WebSocket забезпечує стабільний обмін даними в реальному часі з ефективною буферизацією а розроблений мобільний додаток дозволяє візуалізувати отримані виміри та зберігати їх для подальшого аналізу, що робить систему зручною та практичною для використання в різноманітних сферах автоматизації. Таким чином, поставлені завдання були виконані повністю, а реалізована система може слугувати надійним інструментом для дистанційного контролю відстані.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Самборик В. О. Інтелектуальна система вимірювання відстані з бездротовою передачею даних між Nodemcu та мобільним додатком Flutter // Студентський науковий вісник. Луцьк, 2025. №54. С. 134-140.

2. Види датчиків руху за принципом роботи. URL: https://oxorona.com/motion-sensor-types/ (дата звернення: 14.02.2025).

3. What is a «received light» recognition type laser sensor? URL: https://www.keyence.co.uk/ss/products/sensor/sensorbasics/laser_light/info/ (дата звернення: 19.08.2025).

4. What is a Radar Level Sensor? Working Principle Explained. URL: https://abmsensor.com/what-is-a-radar-level-sensor/ (дата звернення: 22.02.2025).

5. What is LiDAR technology? URL: https://www.generationrobots.com/blog/en/what-is-lidar-technology/? (дата звернення: 19.08.2025).

6. What are the best programming languages for mobile app development? URL: https://itechcraft.com/blog/best-programming-languages-for-mobile-app-development/ (дата звернення: 20.08.2025).

7. Google Trends: cross platform app development. URL: https://trends.google.com/trends/explore?date=all&q=cross%20platform%20app%20development&hl=uk (дата звернення: 22.08.2025).

8. Flutter for Web: How Flutter Web Works? An In-Depth Guide. URL: https://www.solutelabs.com/blog/flutter-for-web-an-ultimate-guide (дата звернення: 25.08.2025).

9. Mansour M, Gamal A, Ahmed AI, Said LA, Elbaz A, Herencsar N, Soltan A. Internet of Things: A Comprehensive Overview on Protocols, Architectures, Technologies, Simulation Tools, and Future Directions. Energies. 2023. Vol. 16, No. 8. P. 30

10. What is MQTT protocol for iot devices? URL: https://psiborg.in/advantages-of-using-mqtt-for-iot-devices/ (дата звернення: 25.08.2025).

11. HTTP Introduction. URL: https://www3.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html (дата звернення: 08.09.2025).

12. What is WebSocket Protocol? URL: https://www.ramotion.com/blog/what-is-websocket/ (дата звернення: 15.09.2025).

13. Sulistyawan V. N., Salim N. A. Parking Tracking System Using Ultrasonic Sensor HC-SR04 and NODEMCU ESP8266 Based IoT. IOP Conference Series. 2023. Vol. 1203, No. 1. P. 22-30.

14. What is Flutter. URL: https://techvify.com/what-is-flutter/ (дата звернення: 27.09.2025).

15. Flutter docs. URL: https://docs.flutter.dev/resources/architectural-overview (дата звернення: 02.10.2025).

16. NodeMCU pinout. URL: https://www.etechnophiles.com/nodemcu-esp8266-pinout-specs-board-layout/ (дата звернення: 06.10.2025).

17. HC-SR04 pinout. URL: https://www.pinterest.com/pin/hcsr04-ultrasonic-distance-sensor-pinout-circuit--1039276051498959444/ (дата звернення: 09.11.2025).

18. Using the HC-SR04 Ultrasonic Distance Sensor with Arduino. URL: https://dronebotworkshop.com/hc-sr04-ultrasonic-distance-sensor-arduino/ (дата звернення: 10.11.2025).

19. The working principle, applications and limitations of ultrasonic sensors. URL: https://www.microcontrollertips.com/principle-applications-limitations-ultrasonic-sensors-faq/ (дата звернення: 10.11.2025).

20. HC-SR04 Value variations over time. URL: https://forum.arduino.cc/t/hc-sr04-value-variations-over-time/336543 (дата звернення: 10.11.2025).

21. Ultrasonic Sensor Review: Comparing DFRobot URM09, HC-SR04, Devantech SRF02 & Maxbotix MB1040. URL: https://www.dfrobot.com/blog-13482.html?srsltid=AfmBOork-AE4pxE0vwKiKWswpm942v3uMKxMup8VeAKDdvvPbDZkKf-L (дата звернення: 10.11.2025).