Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Мобільно-орієнтована кіберфізична система з Bluetooth-підключенням для дистанційного виведення тексту на екран
Назва теми

КвРКІ 210244.21.02.79 ПЗ
Шифр

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія та програмування»*
Назва

Виконав: студент IV курсу, група КІ2-21-2 _____ Андрій МАКАРЧУК
Підпис — Ініціали, прізвище

Керівник _____ Олег ВОЙЧУР
Підпис, дата — Ініціали, прізвище

Нормоконтролер _____ Тетяна КИСІЛЬ
Підпис, дата — Ініціали, прізвище

До захисту допускаю:
зав. кафедри комп'ютерної інженерії та інформаційних систем _____ Ольга ПАВЛОВА
Підпис — Ініціали, прізвище

«18» червня 2025 р.

Хмельницький 2025

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Комп'ютерної інженерії та інформаційних систем

Освітній рівень Бакалавр

Галузь знань 12 Інформаційні технології

Спеціальність 123 Комп'ютерна інженерія

Освітня програма «Комп'ютерна інженерія та програмування»

ЗАТВЕРДЖУЮ

Зав. кафедри Ольга ПАВЛОВА

" 10 " 01 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Андрію МАКАРЧУКУ

Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Мобільно-орієнтована кіберфізична система з Bluetooth-підключенням для дистанційного виведення тексту на екран

Керівник проекту (роботи) Олег ВОЙЧУР, асистент

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 07.02.2025 р. № 23

2. Строк подання студентом проекту (роботи) на кафедру 10.06.2025 р.

3. Вихідні дані до проекту (роботи) Завдання на кваліфікаційну роботу

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Мобільно-орієнтована кіберфізична система з Bluetooth-підключенням для дистанційного виведення тексту на екран, аналіз існуючих аналогів

Проєктування мобільно-орієнтованої кіберфізичної системи з Bluetooth-підключенням для дистанційного виведення тексту на екран

Програмно-апаратна реалізація мобільно-орієнтованої кіберфізичної системи з Bluetooth-підключенням для дистанційного виведення тексту на екран

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

Блок-схема програми мікроконтролера та додатка

UseCase-діаграма користування додатком

Принципова схема та схема підключень

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Тетяна КИСІЛЬ, доцент кафедри КІІС		
Антиплагіат	Андрій НІЧЕПОРУК, доцент кафедри КІІС		

7. Дата видачі завдання «10» 01 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	10.01.2025	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2025	виконано
3	Робота над розділом 1 – дослідження предметної області та постановка задачі	01.03.2025	виконано
4	Робота над розділом 2 – вибір компонентів та попереднє проєктування системи з Bluetooth-підключенням для дистанційного виведення тексту на екран	01.04.2025	виконано
5	Робота над розділом 3 – реалізація проєкту системи з Bluetooth-підключенням для дистанційного виведення тексту на екран	29.04.2025	виконано
6	Оформлення пояснювальної записки згідно вимог	25.05.2025	виконано
7	Попередній захист ВКР	26.05.2025	виконано
8	Захист ВКР на засіданні ЕК	13.06.2025	

Студент _____ Андрій МАКАРЧУК
Підпис Ініціали, прізвище

Керівник роботи _____ Олег ВОЙЧУР
Підпис Ініціали, прізвище

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ 210244.21.02.79 ПЗ	Пояснювальна записка	61		
			Графічні матеріали			
2		КвРКІ 210244.21.02.79 Е8	Блок-схема програми мікроконтролера та додатка	1		
3		КвРКІ 210244.21.02.79 Е8	UseCase-діаграма користування додатком	1		
4		КвРКІ 210244.21.02.79 Е8	Принципова схема та схема підключень	1		

Зм	Арк	№ докум	Підпис	Дата	КвРКІ 210244.21.02.79 ВП		
Розробив		Макарчук			Відомість проекту	Літера	Аркуш / Аркушів
Перевір.		Войчур		12.06.25		У	1 / 1
Н. контр.		Кисіль		12.06.25		ХНУ, КІ2-21-2	
Затв.		Павлова		12.06.25			

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Мобільно-орієнтована кіберфізична система з Bluetooth-підключенням для дистанційного виведення тексту на екран».

Автор роботи: Андрій Макарчук Олександрович.

Керівник роботи: Войчур Олег Юрійович.

Пояснювальна записка: 61 с., 43 рис., 3 табл., 3 дод., 50 джерел.

Графічна частина: 3 креслення.

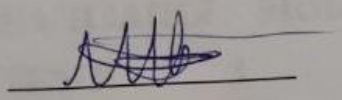
Bluetooth, Android, ESP32, КІБЕРФІЗИЧНА СИСТЕМА, Архітектура.

Метою дипломної роботи є проєктування та розробка мобільно-орієнтованої кіберфізичної системи з Bluetooth-підключенням для дистанційного виведення тексту на екран.

Об’єктом дослідження є системи з дистанційною взаємодією між програмною частиною кіберфізичної системи та їх візуалізаційною частино.

Предметом дослідження є оцінка наявних рішень на ринку та розробка власного варінту реалізації з розширенням можливостей системи.

Під час проведення даного дослідження був використаний метод систематичного огляду літератури для вивчення і аналізу предметної області даного дослідження з текстових джерел інформації.



Підпис студента

30.05.2025

Дата

ЗМІСТ

ВСТУП

Сучасне суспільство дедалі активніше впроваджує автоматизовані та інтелектуальні системи у повсякденне життя. Особливе місце серед них займають кіберфізичні системи (КФС), які забезпечують взаємодію фізичних об’єктів із цифровими платформами за допомогою апаратного і програмного забезпечення. Такі системи дозволяють створювати ефективні рішення для автоматизації, моніторингу та керування різноманітними процесами в реальному часі.

Із розвитком мобільних технологій зросла потреба у створенні мобільно-орієнтованих кіберфізичних систем, які можуть взаємодіяти з користувачем за допомогою смартфонів, планшетів та інших портативних пристроїв. Це забезпечує зручність управління, гнучкість і розширює функціональні можливості систем. Одним із найпоширеніших способів реалізації зв’язку між компонентами таких систем є використання технології Bluetooth, яка забезпечує стабільну бездротову передачу даних на короткі відстані з низьким енергоспоживанням.

У межах даної роботи розглядається розробка мобільно-орієнтованої кіберфізичної системи з Bluetooth-підключенням, що дозволяє дистанційно передавати текстову інформацію на зовнішній екран. Така система може знайти широке застосування в різних сферах: у навчальних закладах, на підприємствах, у засобах масової інформації, в системах для людей з обмеженими можливостями та у багатьох інших випадках, де необхідне оперативне відображення текстової інформації без використання дротового підключення.

Актуальність обраної теми обумовлена потребою в універсальних, простих у використанні та енергоефективних рішеннях для обміну даними між мобільними пристроями та відображувальними пристроями. З огляду на популярність мобільних пристроїв і широке розповсюдження стандарту Bluetooth, запропонована система має потенціал для впровадження в реальні умови.

1 МОБІЛЬНО-ОРІЄНТОВАНА КІБЕРФІЗИЧНА СИСТЕМА З BLUETOOTH-ПІДКЛЮЧЕННЯМ ДЛЯ ДИСТАНЦІЙНОГО ВИВЕДЕННЯ ТЕКСТУ НА ЕКРАН ТА ПОСТАНОВКА ЗАДАЧІ ЩОДО ЇЇ РОЗРОБКИ ТА ВДОСКОНАЛЕННЯ

1.1 Аналіз предметної області, історії розвитку систем дистанційної передачі інформації

Ідея передачі інформації на відстань має давню історію, яка бере початок задовго до появи цифрових технологій. Ще у XIX столітті були винайдені перші телеграфні системи, що дозволяли передавати текстові повідомлення за допомогою електричних сигналів по дротових лініях. Хоча ці рішення не мали компонентів виведення інформації на візуальні екрани, вони заклали основу для майбутніх систем комунікації.

На початку XX століття розвиток радіозв'язку дозволив передавати звукові повідомлення, але відображення текстової інформації залишалося завданням локальних пристроїв. У той самий період активно розвивались технології виведення інформації у вигляді механічних табло(рис. 1.1). Одним із прикладів стали інформаційні табло на залізничних вокзалах і в аеропортах, які використовували механічні елементи(рис. 1.2) для відображення змінних даних, таких яик розклад рейсів або номер платформи.

Рисунок 1.1 – Механічне табло на Смиховському вокзалі (Прага)

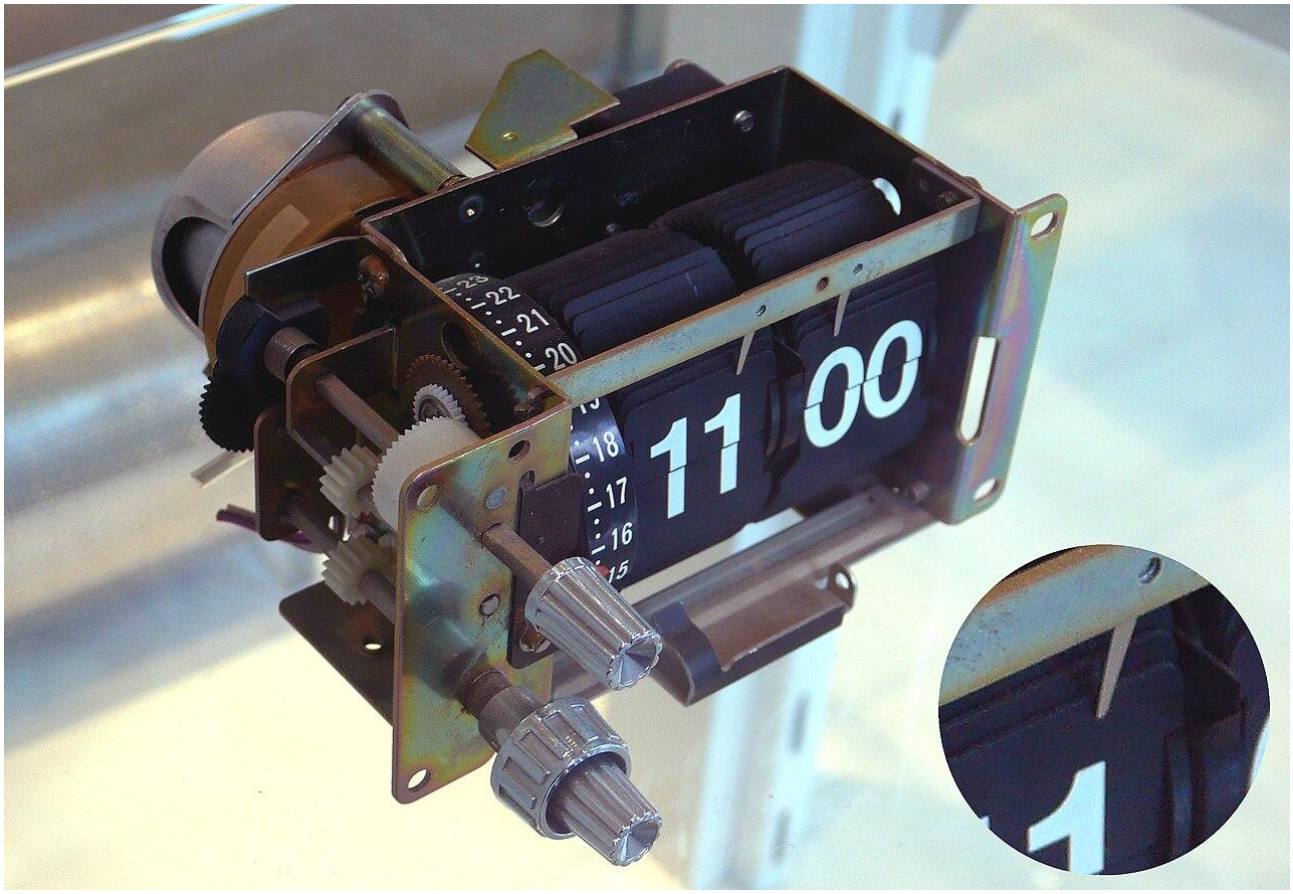


Рисунок 1.2 – Механічний чисельний елемент

У 1960-х роках з розвитком електроніки з’явилися перші електронні дисплеї. Вони використовували лампи або електромеханічні елементи для показу чисел і символів. Особливої популярності набули сегментні дисплеї, які стали стандартом для багатьох побутових пристроїв того часу.

Наступним етапом стала поява рекламних електронних білбордів і інформаційних панелей у громадських місцях. Перші такі системи базувались на світлодіодних (LED) технологіях і керувались централізованими проводовими контролерами. Завдяки цьому стало можливим створення великих екранів для відображення змінної інформації у режимі реального часу.

У 1970–1980-х роках сегментні дисплеї та прості матричні екрани отримали широке застосування в побутовій техніці, транспорті та промисловості. Для їхнього керування використовувались дротові з’єднання, які забезпечували надійний, хоча й обмежений за гнучкістю, канал зв’язку.

У цей період також почали з’являтися комп’ютеризовані системи виведення інформації, де дисплеї керувалися мікропроцесорами. Вони дали змогу не тільки виводити статичні повідомлення, а й змінювати вміст у залежності від зовнішніх умов або команд користувача.

З початку 1980-х років розвиток електроніки та комунікацій привів до перших спроб використовувати бездротову передачу даних для відображення інформації на віддалених екранах. Хоча на той час ці рішення були дорогими та обмеженими у застосуванні, вони заклали основу для майбутнього розвитку кіберфізичних систем.

Паралельно відбувався розвиток рекламних інформаційних систем, які використовували комп’ютери для централізованого оновлення контенту на великих електронних білбордах(рис. 1.3). Ці системи все ще використовували дротове з’єднання, але вже підтримували автоматизацію й програмне керування.

Рисунок 1.3 – Електронні білборди

У 1990-х роках розвиток персональних комп’ютерів і перших мобільних пристроїв розширив можливості виведення інформації. З’явились кишенькові органайзери (PDA), ранні мобільні телефони з невеликими рідкокристалічними дисплеями (LCD). Проте системи керування і передачі даних залишались здебільшого дротовими або здійснювались через фізичне підключення пристроїв.

З часом індустрія стикнулась із обмеженнями дротових рішень, особливо у випадках, коли було потрібно забезпечити гнучкість і мобільність пристроїв. Ці обмеження і стали поштовхом для розвитку перших бездротових технологій передачі даних, зокрема інфрачервоних (ІЧ) інтерфейсів і радіохвиль.

Однією з перших технологій став інфрачервоний (ІЧ) зв’язок, який набув популярності у побутових пристроях, таких як телевізори та аудіосистеми. ІЧ-передачу також використовували у ранніх комп’ютерних периферійних пристроях для обміну невеликими обсягами даних. Однак цей метод мав суттєве обмеження - необхідність прямої видимості між передавачем і приймачем, що не дозволяло створювати гнучкі рішення для керування інформаційними екранами в динамічних умовах.

У цей самий період радіочастотні (RF) технології стали альтернативою інфрачервоному зв’язку. Вони дозволяли передавати сигнали без необхідності прямої видимості, що значно розширювало можливості розміщення пристроїв. Радіокеровані табло та дисплеї почали використовуватись у сфері транспорту - наприклад, на великих вокзалах, де інформаційні панелі отримували оновлення без потреби в складних дротових з'єднаннях.

Проте і ці рішення залишались обмеженими. Вартість впровадження радіокерованих систем була високою, а стандартизація відсутня, що ускладнювало інтеграцію різних пристроїв у єдині системи.

Розвиток персональних комп’ютерів і мобільних пристроїв призвів до зростання потреби у більш ефективних і стандартизованих бездротових технологіях. Це стимулювало пошук нових рішень, які могли б забезпечити

стабільну передачу даних на короткі та середні дистанції з низьким енергоспоживанням.

Відповіддю на ці потреби стала поява нового стандарту - Bluetooth. Спочатку він розроблявся для заміни дротових з'єднань у мобільних пристроях. Його перевагами стали простота використання, універсальність та можливість інтеграції у широкий спектр обладнання.

Згодом технологія Bluetooth була адаптована і для кіберфізичних систем, особливо мобільно-орієнтованих, які вимагали гнучкого, надійного та енергоефективного способу комунікації між контролерами, датчиками та виконавчими пристроями.

У міру розвитку технологій Bluetooth зазнавав удосконалень, які значно розширили його можливості. Перша версія, Bluetooth 1.0, була представлена у 1999 році. Вона забезпечувала максимальну швидкість передачі даних до 721 кбіт/с і підтримувала з’єднання на відстані до 10 метрів. Хоча ця версія мала низку технічних обмежень, таких як складність встановлення з’єднання та недостатня безпека, вона стала першим кроком до широкого впровадження технології.

У подальших версіях було суттєво покращено продуктивність і надійність. Bluetooth 2.0, випущений у 2004 році, запровадив технологію EDR (Enhanced Data Rate), яка збільшила швидкість передачі даних до 3 Мбіт/с. Це дозволило передавати не лише текстову інформацію, а й мультимедійні файли та аудіопотоки з вищою якістю.

З випуском Bluetooth 3.0 у 2009 році було введено HS (High Speed) профіль, який використовував додатковий Wi-Fi-канал для передачі великих обсягів даних на високих швидкостях. Це стало важливим кроком для пристроїв, що потребували швидкого обміну великими файлами, хоча для простих завдань, таких як передача тексту, такі можливості залишались надмірними.

Bluetooth 4.0, випущений у 2010 році, став революційним завдяки введенню стандарту Bluetooth Low Energy (BLE). BLE забезпечив значне зниження енергоспоживання при збереженні достатньої швидкості для передачі невеликих

обсягів даних. Ця версія відкрила нові можливості для мобільно-орієнтованих кіберфізичних систем, особливо в умовах, де пристрої повинні працювати тривалий час на батарейному живленні.

Подальші версії, такі як Bluetooth 5.0 (2016 рік) і наступні, покращили дальність дії, швидкість і енергозбереження. Bluetooth 5.0 забезпечував швидкість до 2 Мбіт/с і дальність до 240 метрів (у режимі низької швидкості), що стало значним проривом для застосувань у великих просторах та промислових системах.

Таблиця 1.1 – Інформація щодо основних версій Bluetooth

Версія	Рік випуску	Макс. швидкість передачі даних	Максимальна дальність дії	Основні особливості
1.0	1999	732,2 кбіт/с	~10 м	Перше комерційне впровадження; проблеми з сумісністю між виробниками
2.0 + EDR	2004	3 Мбіт/с	~10 м	Введення Enhanced Data Rate (EDR) для підвищення швидкості передачі даних
3.0 + HS	2009	до 24 Мбіт/с	~10 м	Використання додаткового Wi-Fi-каналу для високошвидкісної передачі даних
4.0 (BLE)	2010	1 Мбіт/с	~50–100 м	Введення Bluetooth Low Energy (BLE) для зниження енергоспоживання
5.0	2016	2 Мбіт/с	до 240 м	Збільшена дальність дії та швидкість; покращена підтримка IoT-пристроїв

Кінець таблиці 1.1

5.1	2019	2 Мбіт/с	до 240 м	Введення функції визначення напрямку сигналу для покращення навігації
5.2	2020	2 Мбіт/с	до 240 м	Підтримка LE Audio та кодека LC3 для покращення якості звуку
5.3	2021	2 Мбіт/с	до 240 м	Покращення енергоефективності та управління з'єднаннями
5.4	2023	2 Мбіт/с	до 240 м	Оптимізації для пристроїв Інтернету речей (IoT)
6.0	2024	3 Мбіт/с	до 300 м	Введення точного визначення відстані між пристроями та покращення енергоефективності

Завдяки цим удосконаленням Bluetooth перетворився на універсальну платформу для обміну даними між різноманітними пристроями - від простих датчиків до складних дисплейних систем. Для кіберфізичних систем виведення тексту це забезпечило не лише зручність у використанні, а й стабільність, гнучкість та можливість інтеграції з мобільними платформами.

З розвитком технологій, особливо важливою стала поява багатофункціональних мікроконтролерів із вбудованою підтримкою бездротових технологій, таких як ESP32 від компанії Espressif Systems.

ESP32 став популярним рішенням завдяки поєднанню високої продуктивності, низького енергоспоживання та вбудованої підтримки як Wi-Fi, так і Bluetooth/Bluetooth Low Energy (BLE). Це дозволило розробникам створювати

доступні, компактні й гнучкі кіберфізичні системи без необхідності додаткових модулів або складних схем підключення.

Для мобільно-орієнтованих кіберфізичних систем, призначених для дистанційного виведення тексту, ESP32 відкрив нові можливості:

– Підтримка бездротового обміну даними з мобільними пристроями через BLE, що забезпечує стабільне з’єднання з низьким енергоспоживанням.

– Висока обчислювальна потужність для обробки вхідних даних і керування відображенням інформації.

– Простота програмування завдяки підтримці різних мов програмування та великій спільноті розробників.

Крім того, ESP32 широко використовується у прототипуванні й промислових розробках завдяки низькій вартості та наявності численних бібліотек для роботи з Bluetooth і дисплеями. Це зробило мікроконтролер одним із найпоширеніших рішень для створення недорогих і функціональних кіберфізичних систем у різних сферах - від "розумного дому" до промислової автоматизації.

Також завдяки широкому поширенню операційних систем Android та iOS, що підтримують Bluetooth і BLE на рівні системних бібліотек, стало можливим створення зручних і функціональних мобільних застосунків для керування кіберфізичними системами.

У випадку систем дистанційного виведення тексту мобільний пристрій виконує роль основного інтерфейсу користувача. Користувач може за допомогою спеціально розробленого застосунку або стандартного Bluetooth-підключення надсилати текстові повідомлення, які приймає ESP32. Надіслані дані обробляються мікроконтролером і виводяться на підключений дисплей у режимі реального часу.

Крім передачі тексту, мобільний пристрій може надавати додаткові функції, наприклад:

– Вибір шрифту та кольору тексту.

– Управління швидкістю прокручування або оновлення інформації.

– Збереження шаблонів повідомлень.

– Керування кількома дисплеями одночасно через мультипрофільні Bluetooth-з’єднання.

Програмна реалізація взаємодії зазвичай базується на використанні стандартних профілів Bluetooth, таких як Serial Port Profile (SPP) або Generic Attribute Profile (GATT) для BLE. Вони забезпечують простий і стабільний обмін даними між мобільним пристроєм і мікроконтролером.

Завдяки відкритим бібліотекам і численним програмним рішенням для Android та iOS, розробники мають змогу легко створювати інтерфейси користувача без необхідності розробки низькорівневого програмного забезпечення для роботи з Bluetooth. Це суттєво спрощує процес розробки і скорочує час впровадження готового рішення.

Таким чином, комбінація ESP32 та сучасних мобільних пристроїв дозволяє створювати доступні, ефективні та гнучкі мобільно-орієнтовані кіберфізичні системи для дистанційного виведення тексту на екрани в різноманітних сферах — від побутових до промислових застосувань.

1.2 Порівняльний аналіз переваг та недоліків існуючих рішень реалізації передачі візуальної інформації за допомогою Bluetooth

А тепер розглянемо наявні рішення, що забезпечують виведення інформації на екрани з можливістю дистанційного керування. Для кожного прикладу буде проаналізовано основні переваги та недоліки в контексті можливого застосування подібних підходів у розробці мобільно-орієнтованої кіберфізичної системи.

Першим є Bluetooth LED name badge - компактний пристрій у вигляді бейджа з вбудованим світлодіодним дисплеєм, який дозволяє відображати текстові повідомлення. Управління контентом здійснюється за допомогою мобільного додатку через Bluetooth-з'єднання.

Подібні бейджі широко застосовуються на виставках, конференціях, у сфері обслуговування, ресторанах, а також у розважальних закладах для персонального відображення повідомлень, імен або рекламних фраз.

Однією з популярних моделей є Coolbird Blue LED Name Tag(рис 1.4). Цей бейдж підтримує підключення до мобільних пристроїв на базі Android та iOS за допомогою додатку Bluetooth LED Name Badge (рис 1.5). Додаток дозволяє змінювати текст, вибирати стиль прокручування повідомлень, налаштовувати яскравість і шрифти.

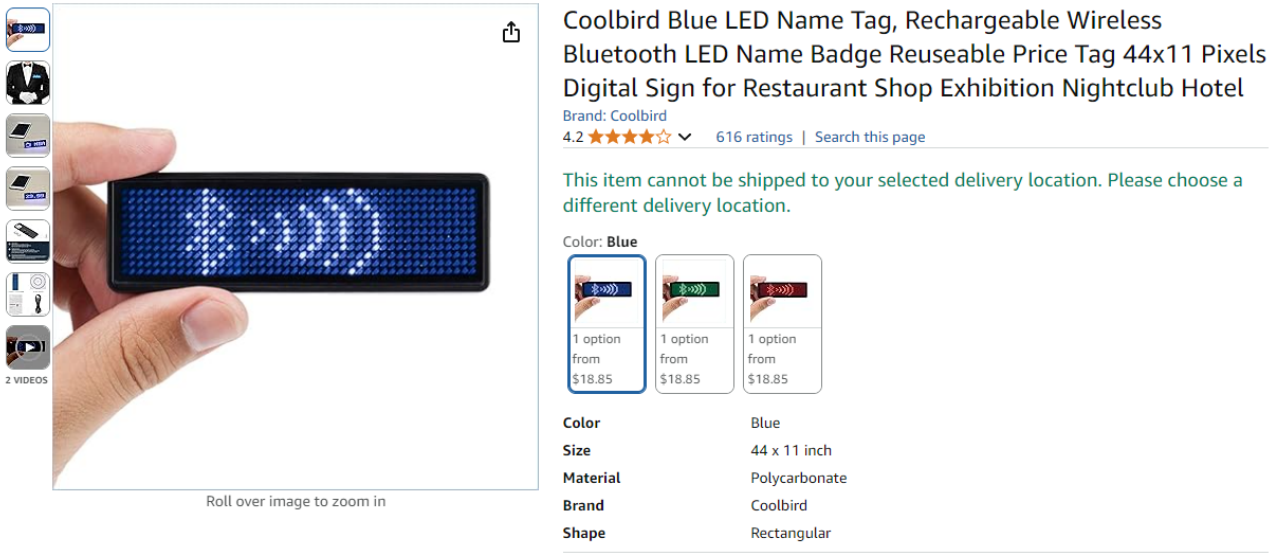


Рисунок 1.4 – Сторінка Coolbird Blue LED Name Tag на Amazon

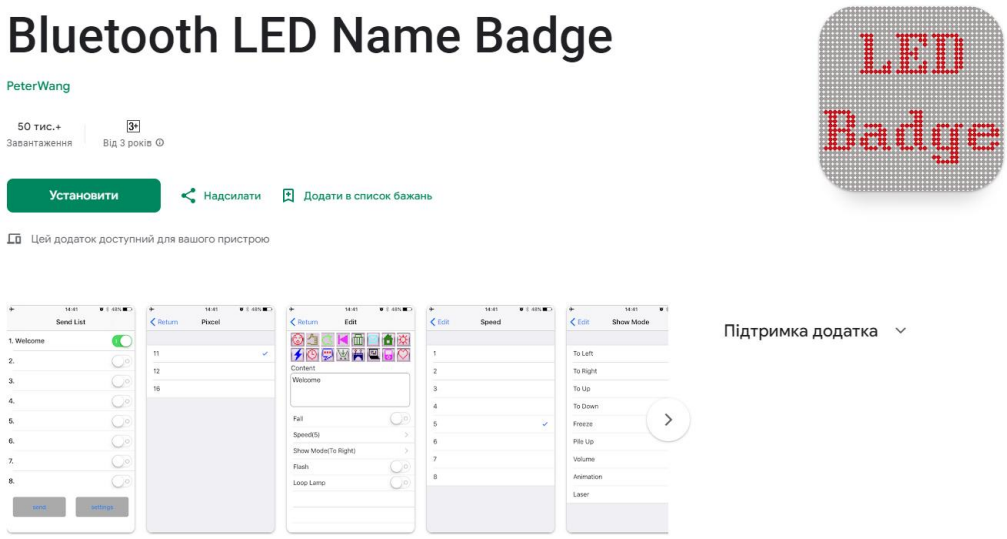


Рисунок 1.5 – Додаток Bluetooth LED Name Badge в Google Play

Серед функціональних можливостей є: відображення прокручуваного або статичного тексту, можливість змінювати текст у реальному часі через додаток, різноманітні режими анімації тексту (прокрутка, миготіння тощо), кілька рівнів яскравості для роботи в різних умовах освітлення.

Основними перевагами є простота використання, налаштування через Bluetooth, компактні розміри та портативність, доступна ціна у порівнянні з більшими інформаційними дисплеями, автономне живлення (вбудований акумулятор).

Слабкими сторонами рішення є : обмежений розмір дисплея та кількість символів, відсутність можливості виводити складну графіку чи великі обсяги інформації, моделі мають невелику дальність роботи Bluetooth (зазвичай до 10 метрів), для змінення інформації необхідно використовувати сумісний мобільний додаток.

Bluetooth LED name badge є прикладом простого й ефективного рішення для виведення коротких текстових повідомлень із можливістю дистанційного керування. Проте його функціонал обмежується лише текстом малої довжини.

Також одним із сучасних рішень для виведення інформації на екрани з дистанційним керуванням є настільні або настінні LED-дисплеї. Ці пристрої зазвичай мають вигляд матриці світлодіодів, які здатні відображати текст, просту графіку або анімовані зображення. Важливою особливістю таких систем є можливість зміни відображуваної інформації в режимі реального часу за допомогою мобільних додатків або інших бездротових засобів управління.

Прикладом такого пристрою є Divoom Pixoo 64(рис 1.6) - багатофункціональний LED-дисплей з роздільною здатністю 64×64 пікселі, призначений для домашнього або офісного використання. Пристрій підтримує як Bluetooth-, так і Wi-Fi-з'єднання, що дозволяє змінювати контент безпосередньо зі смартфона або комп'ютера.

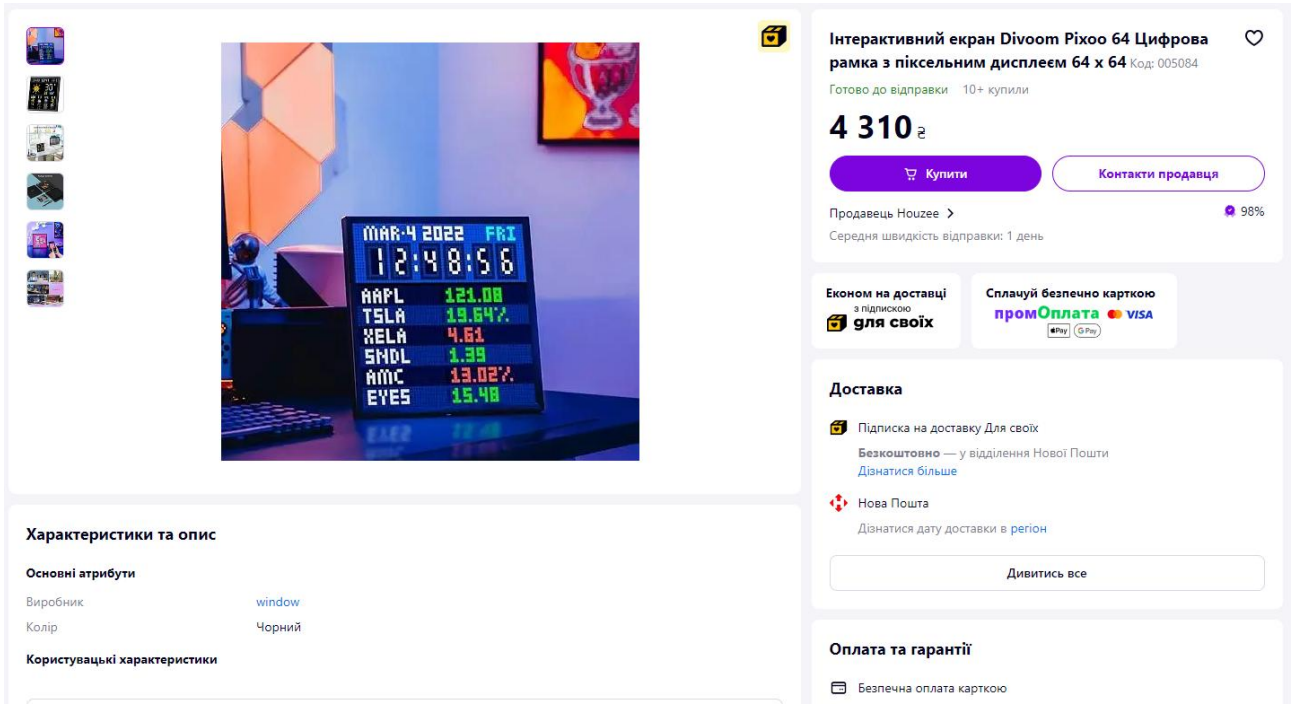


Рисунок 1.6 – Сторінка Divoom Pixoo 64 на Prom

Основні можливості:

– Виведення текстових повідомлень.

– Відображення простих зображень та анімацій.

– Показ сповіщень із соціальних мереж та додатків.

– Відображення часу, погоди, кількості підписників на платформах (наприклад, YouTube, Twitch).

– Інтеграція з платформами для розумного дому.

Для налаштування та керування Divoom Pixoo використовується мобільний додаток Divoom Smart(рис 2.7), що доступний для Android та iOS. Користувач може обирати готові шаблони відображення або створювати власні зображення й текстові повідомлення. Інтерфейс додатку інтуїтивно зрозумілий і дозволяє в реальному часі змінювати контент дисплея.

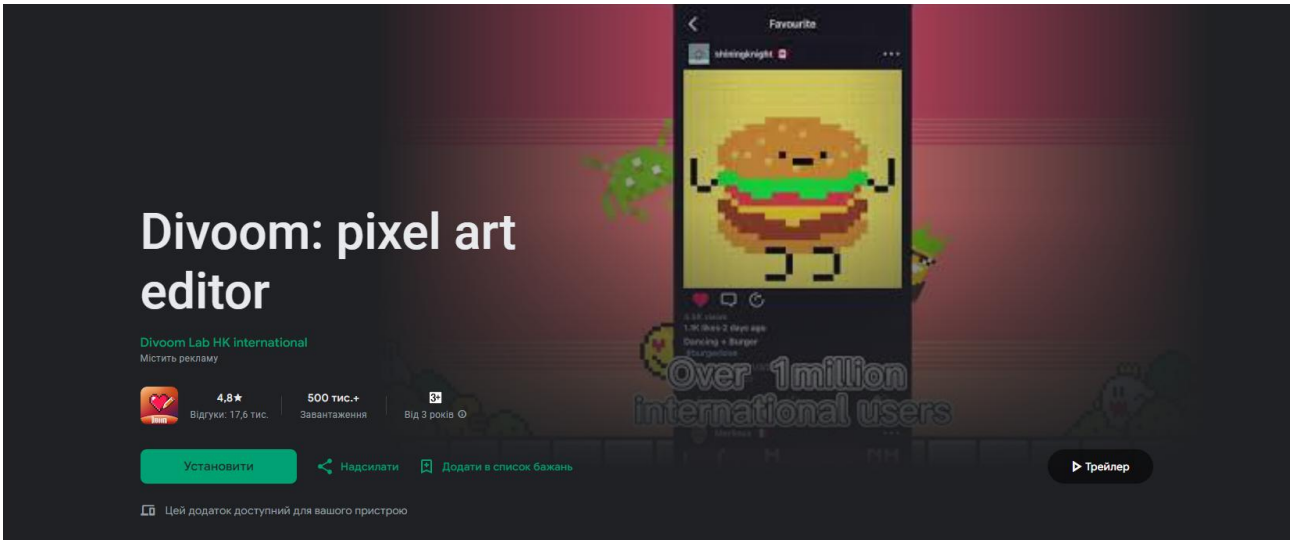


Рисунок 2.7 – Divoom Smart в Google Play

Недоліками цього пристрою є висока вартість у порівнянні з простішими пристроями (наприклад, Bluetooth-бейджами), відносно велике енергоспоживання, особливо при постійному виведенні анімацій, також не підходить для портативного використання через габарити.

Тим не менш Divoom Pixoo є хорошим прикладом багатофункціонального рішення для виведення інформації на екран із дистанційним керуванням. Незважаючи на недоліки, цей пристрій пропонує широкий функціонал і високі можливості для персоналізації та інтеграції з іншими сервісами. Проте для простих задач, таких як відображення коротких текстових повідомлень у компактному форматі, подібне рішення може бути надмірним.

Ще серед компактних пристроїв для відображення інформації окрему нішу займають LED-індикатори статусу, які дозволяють передавати прості кольорові сигнали (наприклад, «зайнято», «вільно», «не турбувати»). Одним із найпопулярніших продуктів цього типу є Luxafor Bluetooth.

Luxafor Bluetooth(рис 2.8) - невеликий LED-індикатор, розроблений для відображення статусів зайнятості або інших простих сигналів. Він використовується в офісах для позначення статусу співробітника або для візуального сповіщення про певні події.

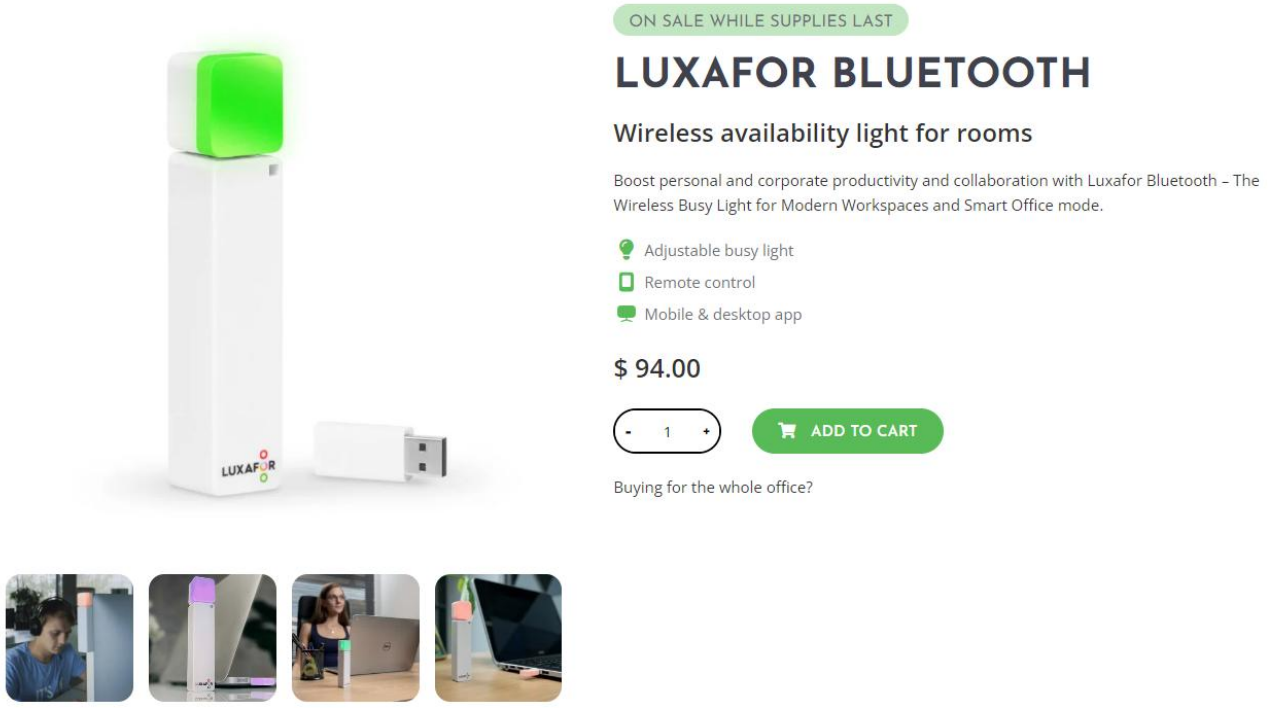


Рисунок 2.8 - Luxafor Bluetooth на офіційному сайті Luxafor

Основні можливості Luxafor: відображення кольорових сигналів, які можуть означати різні стани, можливість програмування кольорів та їхньої зміни через додаток(рис 2.9), керування через Bluetooth або Wi-Fi (у відповідних моделях), підтримка інтеграції з популярними офісними програмами (наприклад, Microsoft Teams, Skype, Google Calendar).

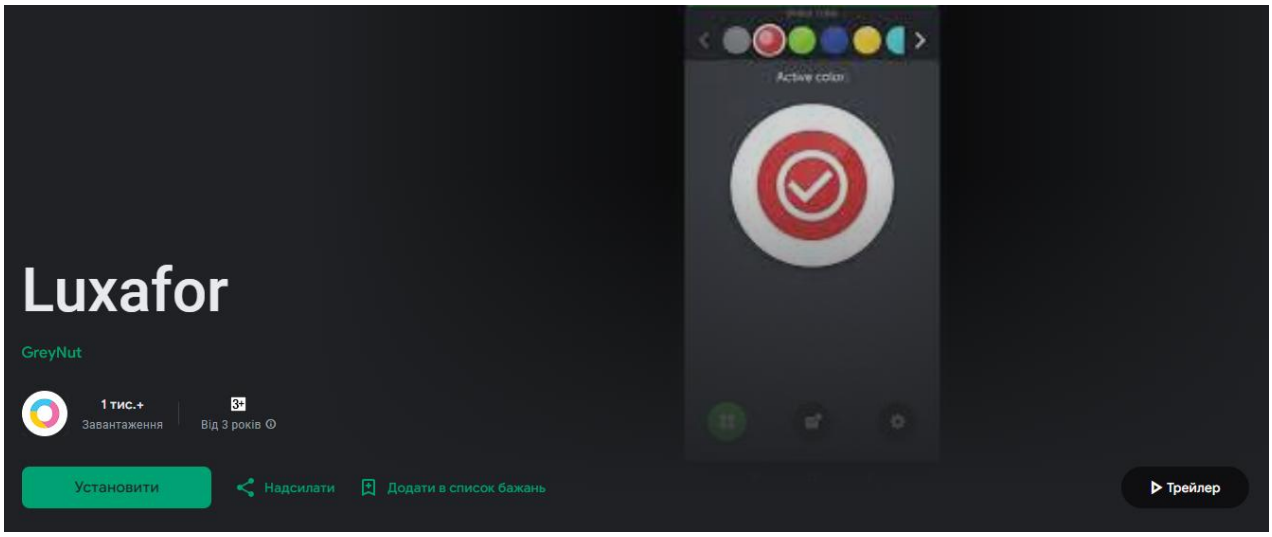


Рисунок 2.9 – Додаток Luxafor в Google Play

Перевагами такого рішення є простота налаштування і використання, дуже низьке споживання енергії, можливість автоматизації зміни статусу за подіями, компактні розміри, портативність. А от слабкими сторонами в порівнянні з іншими варіантами є, відсутність виведення тексту або складних повідомлень, обмежені можливості кастомізації порівняно з більш функціональними дисплеями.

Незважаючи на обмеження у функціональності, він ефективно справляється зі своїми задачами. Це рішення підходить для завдань, де необхідна індикація статусів без потреби у складній інформаційній взаємодії.

1.3 Висновки до першого розділу

У першому розділі було здійснено комплексне дослідження історичного розвитку та еволюції систем дистанційного виведення інформації, з акцентом на їх структурні та функціональні особливості. Особливу увагу приділено бездротовим технологіям передачі даних, зокрема Bluetooth, як одній з найперспективніших для створення компактних та енергоефективних кіберфізичних систем.

Проаналізовано основні версії Bluetooth та їхні можливості у контексті побудови систем із дистанційним управлінням. Проведено порівняльний аналіз існуючих рішень на ринку, виявлено їх переваги й обмеження, що дало змогу сформувати вимоги до власної розробки.

Отримані результати створили ґрунтовну теоретичну базу для подальшого проєктування мобільно-орієнтованої кіберфізичної системи з підтримкою Bluetooth, яка відповідатиме сучасним вимогам зручності, мобільності та інтеграції з персональними пристроями користувачів.

2 ПРОЄКТУВАННЯ МОБІЛЬНО-ОРІЄНТОВАНОЇ КІБЕРФІЗИЧНОЇ СИСТЕМИ З BLUETOOTH-ПІДКЛЮЧЕННЯМ ДЛЯ ДИСТАНЦІЙНОГО ВИВЕДЕННЯ ТЕКСТУ НА ЕКРАН

2.1 Постановка задачі та вимог до системи

Головною ціллю розділу проєктування є постановка задачі, визначення вимог до системи, вибір засобів реалізації проєкту а також побудова необхідних схем.

За темою цього дипломного проєкту, необхідно зпроєктувати мобільно-орієнтовану кіберфізичну систему для дистанційного виведення інформації на екран. Отже, система повинна складатися з екрану для виводу інформації, програмованого мікроконтроллера, який матиме підтримку дистанційного з’єднання типу Bluetooth, а також мобільного додатку, за допомогою якого можна буде підключитися до мікроконтроллера по Bluetooth, а також передати текстову чи візуальну інформацію. На рисунку 2.1 зображено схематичний вигляд цієї кіберфізичної системи.

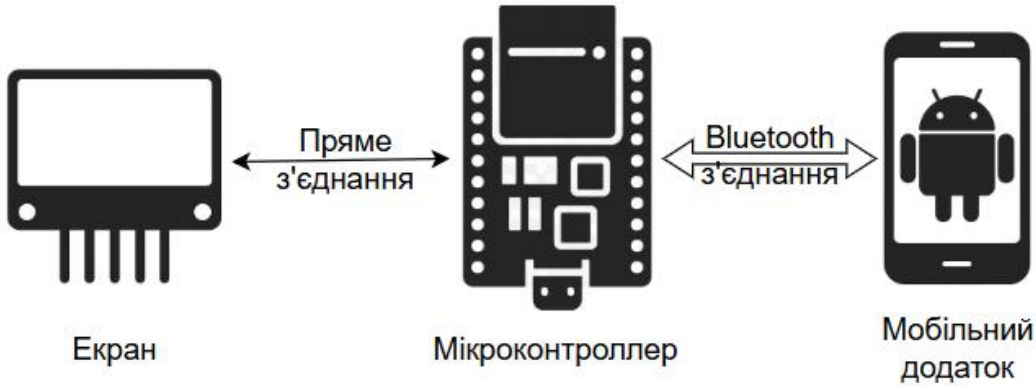


Рисунок 2.1 – Схема кіберфізичної системи

В кінечному результі очікуються такі можливості системи :

– Справне знаходження і підключення до мікроконтроллера через мобільний додаток.

– Передача та вивід текстової та візуальної інформації на екран.

2.2 Вибір мікроконтролера. ESP32

Після визначення цілей проєкту та складання схеми кіберфізичної системи, можна перейти до аналізу та вибору компонентів системи.

Розпочнемо з серця системи - програмованого мікроконтролера, він же й буде посередником між екраном та додатком. Отже, на ринку є чимало моделей Raspberry Pi, Arduino, ESP та інших, і виходячи з вимог системи оберемо найбільш підходящий та доступний варіант. Оскільки для зв'язку з мобільним додатком використовуватиметься Bluetooh, логічно буде розглянути варіанти мікроконтролерів з вбудованими Bluetooth-модулями. Також для роботи з передачею зображень необхідна достатня кількість оперативної та флеш-пам'яті.

Далі розглянемо наявні варіанти мікроконтролерів, проаналізуємо які підходять і виберемо найоптимальніший.

Першими кандидатами є Arduino Uno та Arduino Nano. Це хороші універсальні мікроконтролери зі справді обширним комюніті. Однак, їхніми недоліками є доволі слабкі характеристики оперативної та флеш пам'яті, 32кб флеп-пам'яті та лише 2кб оперативної пам'яті. З таким об'ємом буде складно працювати навіть зі статичними бітмап-зображеннями, а про анімації взагалі можна не думати. Єдиний варіант це розширення пам'яті. Але це не головний недолік, бо в них також відсутній Bluetooth, і в цьому випадку доведеться окремо докуповувати й встановлювати Bluetooth-модуль типу HC-05, HC-06 або HM-10. Використання цього варіанту можливе, але це ускладнить систему, збільшить час діставання комплектуючих, збірки, і сумарно може вийти дорожче аніж інші варіанти.

Вирішенням цих проблем можуть стати плати Arduino MKR-серії та Nano 33. Вони мають вбудований модуль Bluetooth, а також суттєво більший об'єм пам'яті. Однак ціна їхня трохи «кусається», в районі 1000-1500 грн за плату.

Також серед кандидатів можна розглянути ESP8266. Це доволі дешевий варіант з нормальним обсягом пам'яті. Але має таку ж проблему як деякі з попередніх кандидатів – не мають вбудованого Bluetooth-модуля.

Ще одними серед кандидатів є моделі Raspberry Pi. Це вже в більшості не просто плата, а SBC - Single Board Computer, тобто цілий одноплатний комп'ютер. Відверто кажучи, це занадто для нашої задачі, адже ціна немаленька, функціональність значно більше ніж нам необхідно.

Як варіант можна було б ще розглянути Raspberry Pi Pico. Ця мікроконтролерна плата має достатню потужність, невисоку ціну, однак знову таки не має вбудованого Bluetooth-модуля.

Останньою розглянемо плату ESP32 з модулем ESP32-WROOM-32. Цей варіант має 520кб оперативної пам'яті та 4мб флеш-пам'яті, що є комфортним об'ємом для нашого проєкту. Також ця плата має вбудований модуль Bluetooth, працює відразу в двох режимах : Bluetooth Classic та Bluetooth Low Energy. Ну і на останок можна виділити, що ця модель має оптимальну ціну - біля 300 гривень за одну плату.

Тепер можна скласти порівняльну таблицю (табл 2.1) розглянутих варіантів і обрати найоптимальніший.

Таблиця 2.1 – Порівняння характеристик плат з мікроконтролерами

Модель	Оперативна пам'ять	Flash-пам'ять	Bluetooth	Орієнтовна ціна, грн
Arduino Uno / Nano	2 КБ	32 КБ	-	250
Arduino MKR / Nano 33	32–64 КБ	до 256 КБ	+	1000-1500
ESP8266	до 80 КБ	1–4 МБ	-	150-250
Raspberry Pi (3/4/Zero W)	512 МБ і більше	до 32 ГБ	+	1000-3000
Raspberry Pi Pico	264 КБ	2 МБ	-	250
ESP32 (WROOM-32)	520 КБ	4 МБ	+	300

Розглянувши усі варіанти, можна прийти до висновку що найоптимальнішим вибором буде плата ESP32 (рис 2.2). Вона при невисокій вартості закриває усі потреби нашого проєкту.

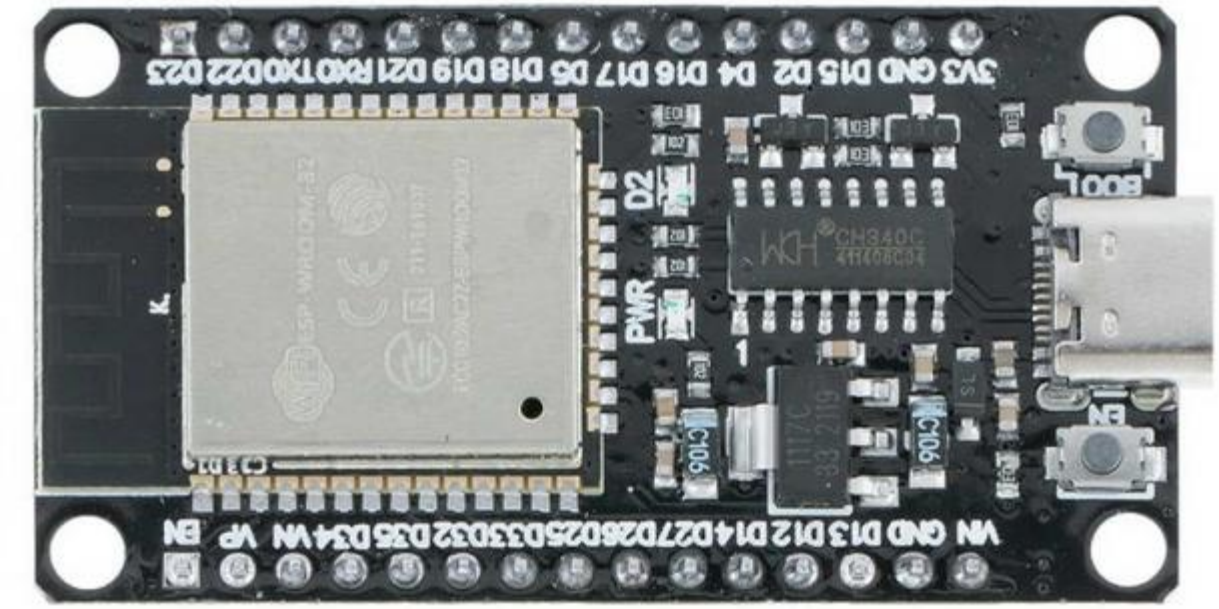


Рисунок 2.2 – ESP32

2.3 Вибір екрану. 0.96’’ OLED (SSD1306)

Наступним компонентом системи є дисплей. Основною задачею для нього буде вивід тексту, отриманого від мікроконтролера. Також він повинен бути придатним для виводу зображень, але оскільки не було поставлено серйозних вимог щодо їх формату, можна використовувати Bitmap-формат. Це дозволить зекономити на вартості дисплея, адже нас задовільнить простий екран з монохромною гаммою.

Варіантів вибору серед дисплеїв справді є багато, на будь-який смак і потреби. Але з огляду на невеликі вимоги проєкту, можна звернути увагу на дисплеї, які можуть іти в комплекті з мікроконтролерною платою та іншими компонентами.

Одним з таких є OLED-дисплей на базі мікроконтроллера SSD1306 (рис. 2.3). Він йде в комплекті з ESP32 в наборі ESP32 Basic Starter Kit. Його роздільна здатність 128 на 64 пікселі, монохромними кольорами (чорний/білий), інтерфейсом підключення SPI, що чудово підходить для ESP32.

Рисунок 2.3 – 0.96’’ OLED-дисплей

Отже компактність і універсальність даного дисплея дозволить виконати всі вимоги системи без жодних проблем, а ще, на відміну від LCD-дисплеїв, цей екран не потребуватиме додаткового освітлення Також, через особливість будови, чорний колір справді буде виглядати чорним, оскільки підсвічуються лише білі пікселі.

2.4 Проєктування системи ESP32-Екран

На даному етапі проєктування ми вже обрали використовуваний дисплей та плату з мікроконтролером, а отже у нас наявні усі компоненти для побудови фізичної частини кіберфізичної системи. В першу чергу необхідно побудувати принципову електричну схему, на якій основною задачею буде розмістити усі елементи системи та побудувати коло руху струму.

Для побудови буде використаний графічний редактор draw.io. Він є достатньо простим, але водночас досить функціональним щоб покрити усі наші потреби відносно побудови схем та діаграм.

Розпочати слід з мікроконтролера ESP32. Для його зображення буде використаний звичайний прямокутний елемент з підписом по-центру. Далі розмістимо на схемі дисплей, який теж буде зображений як прямокутник з підписом, але вже інших розмірів, таким чином щоб візуально виникала асоціація з прототипом.

Після розміщення основних елементів, потрібно позначити джерела струму. Оскільки вся система живиться від плати, яка в свою чергу живиться через вхід micro-USB, вважатимемо ESP32 за джерело струму. Позначимо вихід 3.3V, що є «+» кола, а також GND, що є «-», і з'єднаємо плату з дисплеєм.

Також для повноти системи було вирішено додати до кола світлодіод, який використовуватиметься для сигналізації встановленого Bluetooth-з'єднання між мобільним додатком та платою. Для цього введемо ще один «+», який підпишемо як GPIO. Далі розташуємо паралельно до дисплею світлодіод, він буде позначений символом лампи. І для безпеки розташуємо резистор R 220Ом, який буде стандартно зображений в вигляді прямокутника. Виконавши усі підключення, отримаємо схему, що зображена на рисунку 2.4.

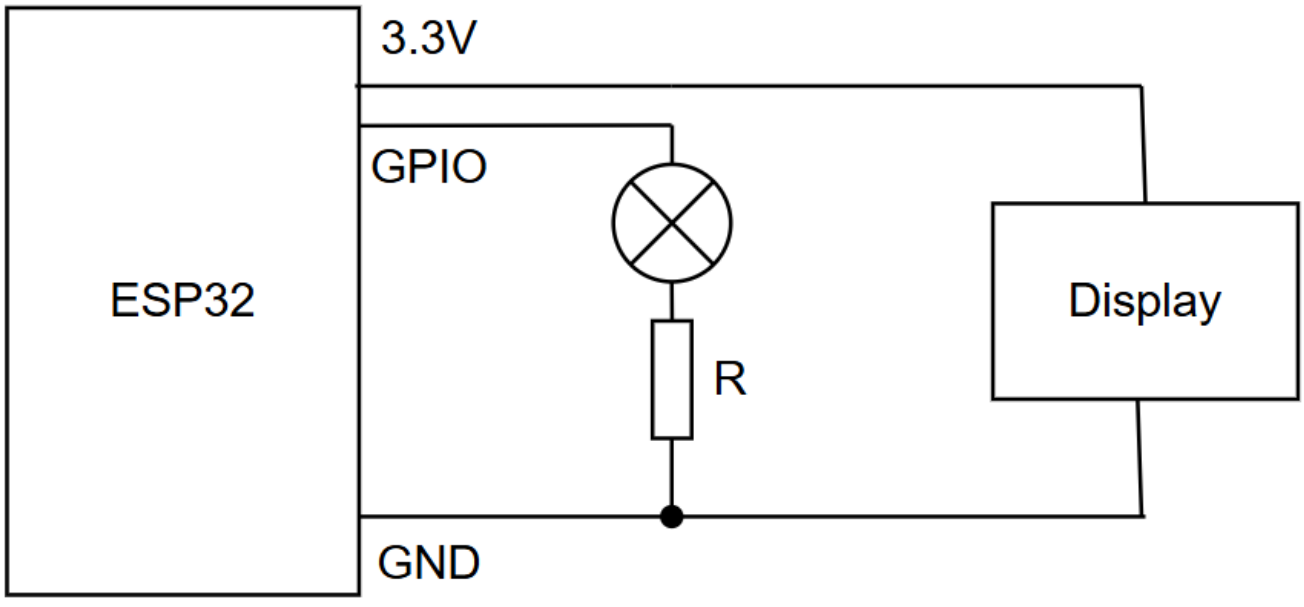


Рисунок 2.4 – Принципова схема електричного кола системи

Тепер, після проєктування принципової схеми, необхідно побудувати схему підключень, яку ще називають монтажною схемою. На ній ми вже точно вкажемо усі підключення, і кінцевий вигляд проєктованої системи.

Як середовище проєктування використаємо wokwi.com, адже цей ресур надає усі необхідні інструменти для створення якісних і точних схем підключень.

Зайшовши на сайт, обираємо лінійку ESP32, і серед моделей виберемо класичний варіант. На робочому просторі вже з'явиться готова плата (рис 2.5) зі всіма необхідними пінами, тому можна приступити до розташування решти елементів.

Рисунок 2.5 – ESP32 на сайті Wokwi

В першу чергу використаємо макетну плату, аби уникнути плутанини і створити акуратну схему підключень. Під наші потреби підійде і повнорозмірна, і зменшена. Міні-варіант теж можна використати, але це може вплинути на зручність і обмежити можливості, тому для наочності використаємо повнорозмірну монтажну плату (рис 2.6).

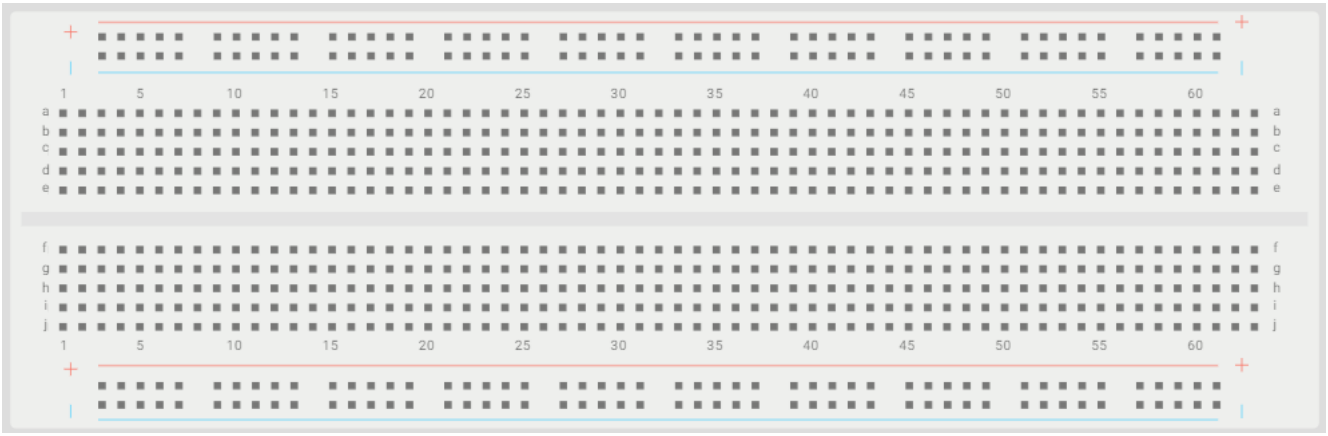


Рисунок 2.6 – Повнорозмірна монтажна плата на сайті Wokwi

Оскільки в проєкті ми використовуємо стандартний дисплей який часто іде в наборі з ESP32, на Wokwi ми можемо знайти точно такий же екран (рис 2.7) серед стандартних компонентів, що спрощує процес проєктування. Розташуємо дисплей на платі, встановивши стовпці E, GND на 29 рядок, VCC на 30 рядок, SCL на 31 рядок і SDA на 32 рядок.

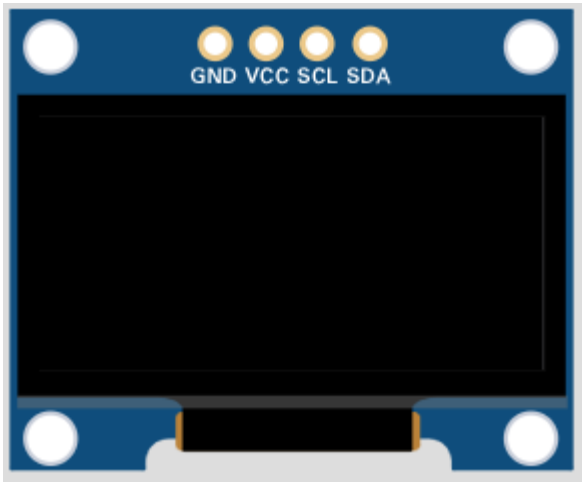


Рисунок 2.7 – OLED дисплей на сайті Wokwi

Наступним кроком під’єднаємо плату та екран. Для цього під’єднаємо пін 3.3V ESP32 на лінію позитивного напряму струму, і від неї підключимось до VCC дисплею. Далі GND плати пустимо на лінію негативного напрямку струму, та під’єднаємо до GND дисплею.

Тепер потрібно під’єднати інтерфейси передачі даних та такту, тобто SDL та SCL відповідно. На ESP32 SDA прив’язаний до GPIO21, а SCL до GPIO22. А значить під’єднуємо GPIO21 та GPIO22 до SDA та SCL дисплею відповідно.

І на завершення потрібно під’єднати до схеми світлодіод (рис 2.8) та убезпечити його резистором (рис 2.9). Для цього виберемо зі списку стандартних елементів «LED» та розташуємо його на монтажній платі. На стовпці J поставимо анод на 10 стрічку, а катод на 11 стрічку.

Рисунок 2.8 – LED на сайті Wokwi

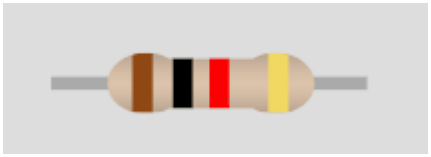


Рисунок 2.9 – Резистор на сайті Wokwi

Тепер під’єднаємо резистор. Оскільки він неполярний елемент, просто розташуємо його на платі, встановивши на стовпці F одною ніжко на 4 стрічку, а іншою на 10 стрічку. Тепер з’єднаємо діод та резистор з мікроконтролерною платою. Для цього підключимо вихід GPIO5 плати до стрічки 4 на стовпці J, а канод діода проведемо до GND ESP32.

Отже тепер схема з’єднань готова, ми підключили мікроконтролер, екран а також діод, і тепер правильно запрограмувавши ESP32 ми можемо добитися необхідного результату. При збірці фізичного пристрою не обов’язково виконувати підключення на макетній платі саме так, як ми описали, головне сама логіка підключень була сбережена. На рисунку 2.10 ми можемо побачити повну схему.

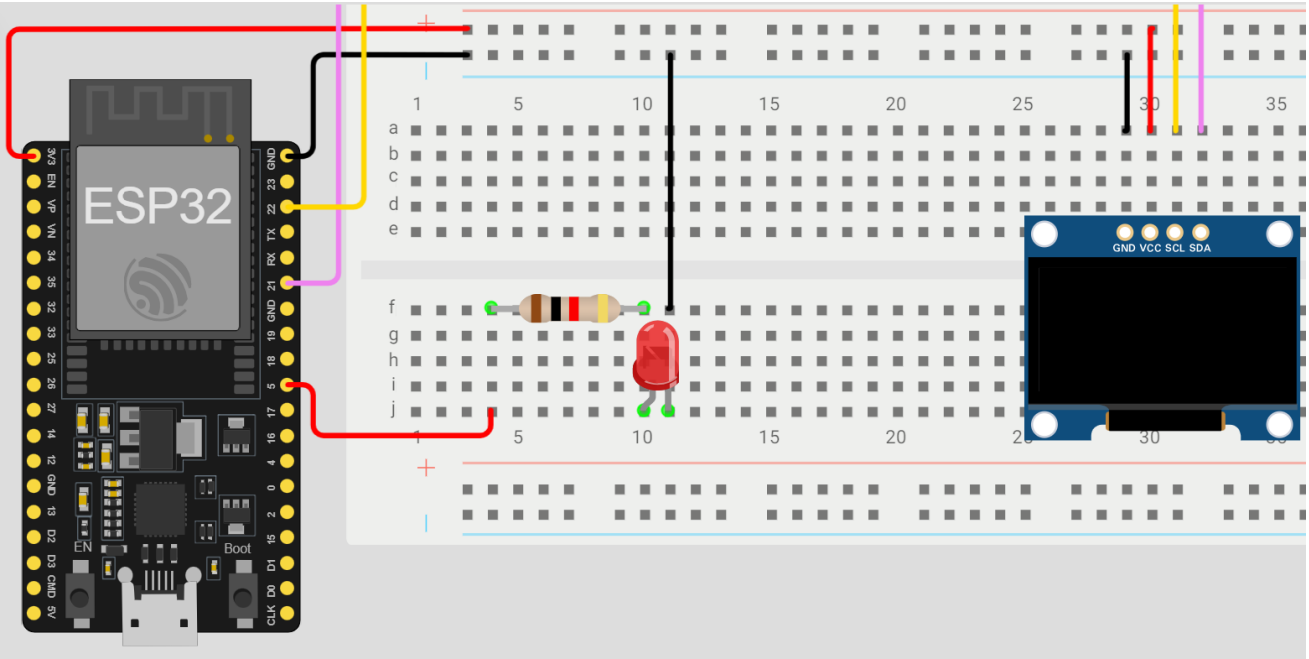


Рисунок 2.10 – Схема підключень системи мікроконтролер-дисплей

2.5 Проєктування мобільного додатку

Після того, як ми успішно зпроєктували фізичну частину системи, необхідно приступити до мобільного додатка. В цьому розділі ми лише загально оглянемо який додаток очікується, які засоби будемо використовувати і як частини додатку взаємодіють одна з одною.

Є два основних підходи до створення мобільних додатків, а саме нативна розробка та кросплатформенна. Розглянемо кожен із них.

Кросплатформенний підхід дозволяє розробити один додаток, який працюватиме на двох платформах, на IOS та Android. Таке рішення зручне для додатків в яких немає необхідності в складній взаємодії з системними API. Таке рішення дозволяє зекономити кошти на розробку, адже не потрібно займати двох нативних програмістів або створювати дві команди. Серед основних кросплатформенних фреймворків (рис 2.11) є React Native на мові JavaScript, Flutter на мові Dart, .NET MAUI на C# а також Kotlin Multiplatform Mobile на Kotlin.

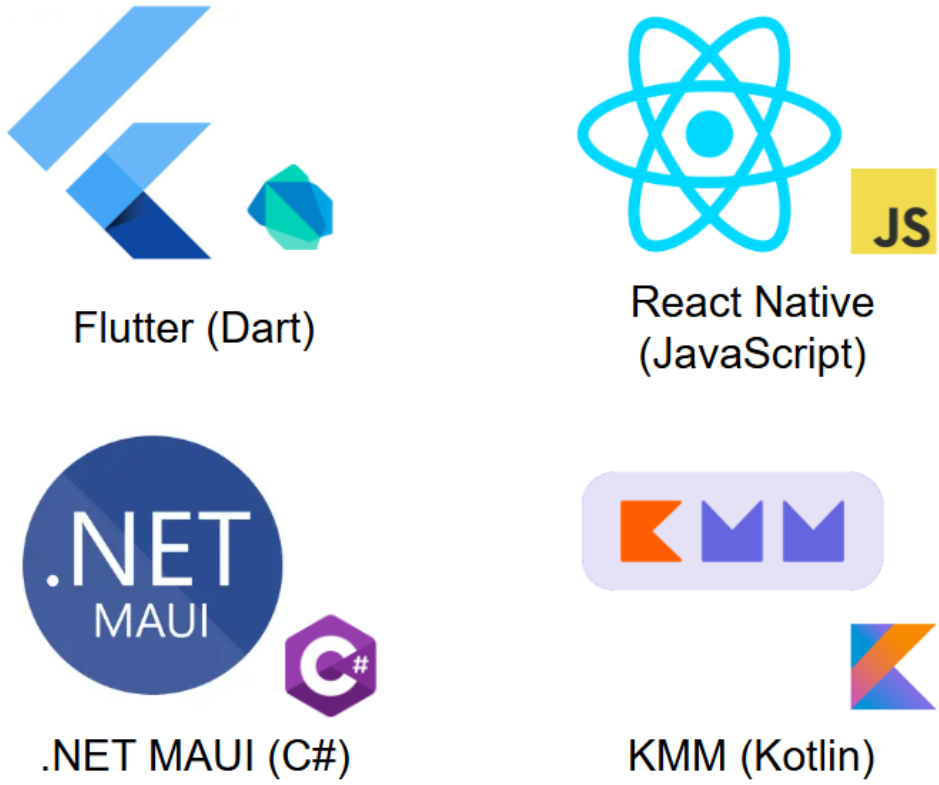


Рисунок 2.11 – Кросплатформені фреймворки

Нативна ж розробка полягає в створенні окремих додатків для обох платформ. Це може вийти дорожче, а також вимагає або знання нативної розробки під обидві платформи та необхідну техніку (створювати та тестувати додатки на IOS можна тільки на Mac та пристроях з ОС IOS), або найму двох розробників чи створення двох команд окремо для кожної платформи. Також ускладнюється процес синхронізації платформ.

Але в цього підходу також є плюси, адже оскільки розробке ведеться під конкретну платформу, можна врахувати всі особливості системи та на повну використати її можливості продуктивності та оптимізації, а також більш вільне користування системними API. Також вартує зазначення те, що при паралельному програмуванні додатків на двох платформах це займає менше часу аніж при кросплатформенній розробці.

Якщо розглядати конкретні засоби(рис 2.12), то для розробки під Android використовують дві мови : Java та Kotlin. Java є класичною мовою, майже всі нативні додатки яким більше 5 років написані на ній. А Kotlin це вже новинка, яку просуває Google, в основному на ній пишуться всі нові додатки, і, якщо це має сенс, переходять старі. А для розробки під IOS використовується Swift, та колись використовувався ObjectiveC, який зараз є застарілим.

Рисунок 2.11 – Нативні мови програмування для операційних систем

Враховуючи, що автор дипломної роботи має бодай якесь уявлення лише про розробку під Android на Kotlin, а також те що необхідно буде працювати з системними API Bluetooth Low Energy, в проєкті буде використовуватись саме цей підхід.

Тепер, коли була наведена ясність щодо використовуваних технологій, можна розглянути архітектуру майбутнього додатка. Для цього потрібно

визначитися з необхідною функціональністю додатку. Спочатку розглянемо усі сценарії, а в кінці все складемо в UseCase-діаграму.

Отже, оскільки за завданням проєкту нам потрібно взаємодіяти з мікроконтролером за допомогою Bluetooth, буде логічно визначити першим необхідним кроком підключення до пристрою. Тоді спочатку користувач повинен натиснути на кнопку пошуку, а наступним кроком обрати необхідний пристрій зі списку доступних.

Після підключення, користувач отримає доступ до двох опцій, а саме введення і відправка тексту на екран, та перехід в редактор зображень.

Оскільки в нашому проєкті ми працюємо з бітмап-зображеннями, важливо перед конвертацією та відправкою обробити їх, аби досягти коректного розподілення монохромних кольорів.

Загалом цієї логіки достатньо для виконання вимог проєкту, тому можна впевнено перейти до структурування сценаріїв в UseCase-діаграму. Ми маємо два основних шляхи, кінцеві цілі користувача.

Перша - вивести на екран текст, і для цього користувачу необхідно виконати таку послідовність дій : запустити додаток (а також при необхідності надати потрібні дозволи), розпочати пошук доступних пристроїв, обрати потрібний пристрій зі списку, далі ввести текст та відправити його на мікроконтролер, який в свою чергу виведе текст на екран.

А друга ціль – відправити зображення на екран. В такому випадку користувачу необхідно виконати таку послідовність дій : відкрити додаток, ввімкнути пошук пристроїв, обрати в списку потрібний пристрій, перейти на екран створення зображення для відправки, обрати з галереї бажане зображення, відредагувати його (контраст, яскравість, форматування по розміру екрану), і нарешті відправити готове зображення на автоматичну конвертацію в формат Bitmap та відправку на пристрій.

Кінцева UseCase-діаграма зображена на рисунку 2.13.

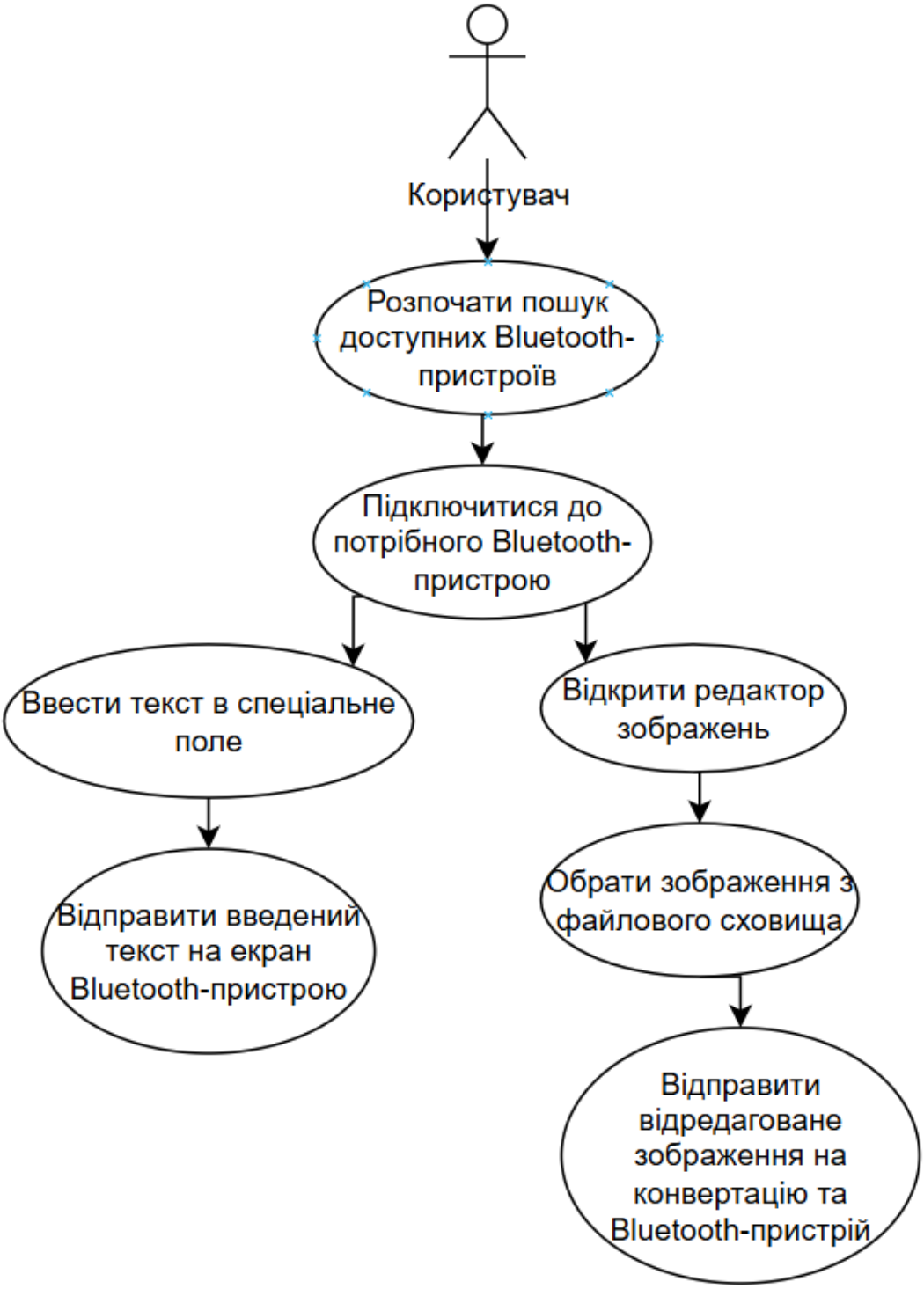


Рисунок 2.13 – Діаграма UseCase мобільного додатку

Тепер можна детальніше поговорити про очікування від додатка. Проаналізуємо потреби користувача, складемо наповнення екранів, їхню кількість та функціональність.

Отже, при відкритті додатку в користувача можуть бути запитані дозволи на управління Bluetooth. Після надання дозволів, користувач потрапить на головний екран додатку.

Оскільки, за сценарієм UseCase, користувач обов'язково захоче розпочати пошук пристроїв, необхідно розмістити на екрані кнопку, що запустить функцію пошуку Bluetooth-пристроїв в радіусі дії. Також знайдені пристрої потрібно буде кудись вивести, тому додамо на активіті контейнер-список, в який будуть виводитися адреси пристроїв та їх імена.

Наступним кроком користувача має бути підключення до потрібного пристрою, тому логічно буде зробити підключення через натиснення на бажаний пристрій в раніше доданому списку.

Після цієї дії, користувач по сценарію матиме два варіанти, або бажання відправити текст, або бажання відправити картинку.

Для відправки тексту, користувачу достатньо ввести його в відповідне поле та натиснути на кнопку відправлення. Отже розмістимо на екрані також відповідні елементи, які з'являтимуться після підключення до правильного пристрою.

А от реалізація другого сценарію є складнішою. Якщо користувач захоче відправити зображення, додаток має дати вибрати його в файловому менеджері або галереї. Потім зображення необхідно буде відредагувати, щоб добитися бажаного відображення на монохромному екрані. З огляду на це все, логічно буде винести всі операції над зображеннями на окремий екран. А отже, на головному екрані, при підключенні до потрібного пристрою, з'являтиметься кнопка переходу на екран редагування зображень.

Далі на новому екрані потрібно розмістити кнопку, що дозволятиме відкрити галерею та завантажити бажане зображення в додаток. Логічно також буде розташувати на екрані контейнер, в якому буде відображатись обране зображення, а також воно буде синхронно змінюватись зі значеннями регуляторів яскравості та контрасту, які ми також додамо на екран.

Також необхідно врахувати, що розміри екрану, на який буде відправлено зображення складає 128 на 64 пікселі, тобто співвідношення сторін 2 : 1, а отже для коректного відображення необхідно додати можливість заповнення простору, який не вписався в очікувані розміри. Ну і для зручності зробимо можливість обрати колір заповнення.

На останок додамо кнопку для підтвердження результатів обробки зображення, його конвертацію в Bitmap-формат та відправку на мікроконтролер.

Отже макети головного екрану зображено на рисунку 2.14, а макет екрану редагування зображення на рисунку 2.15.

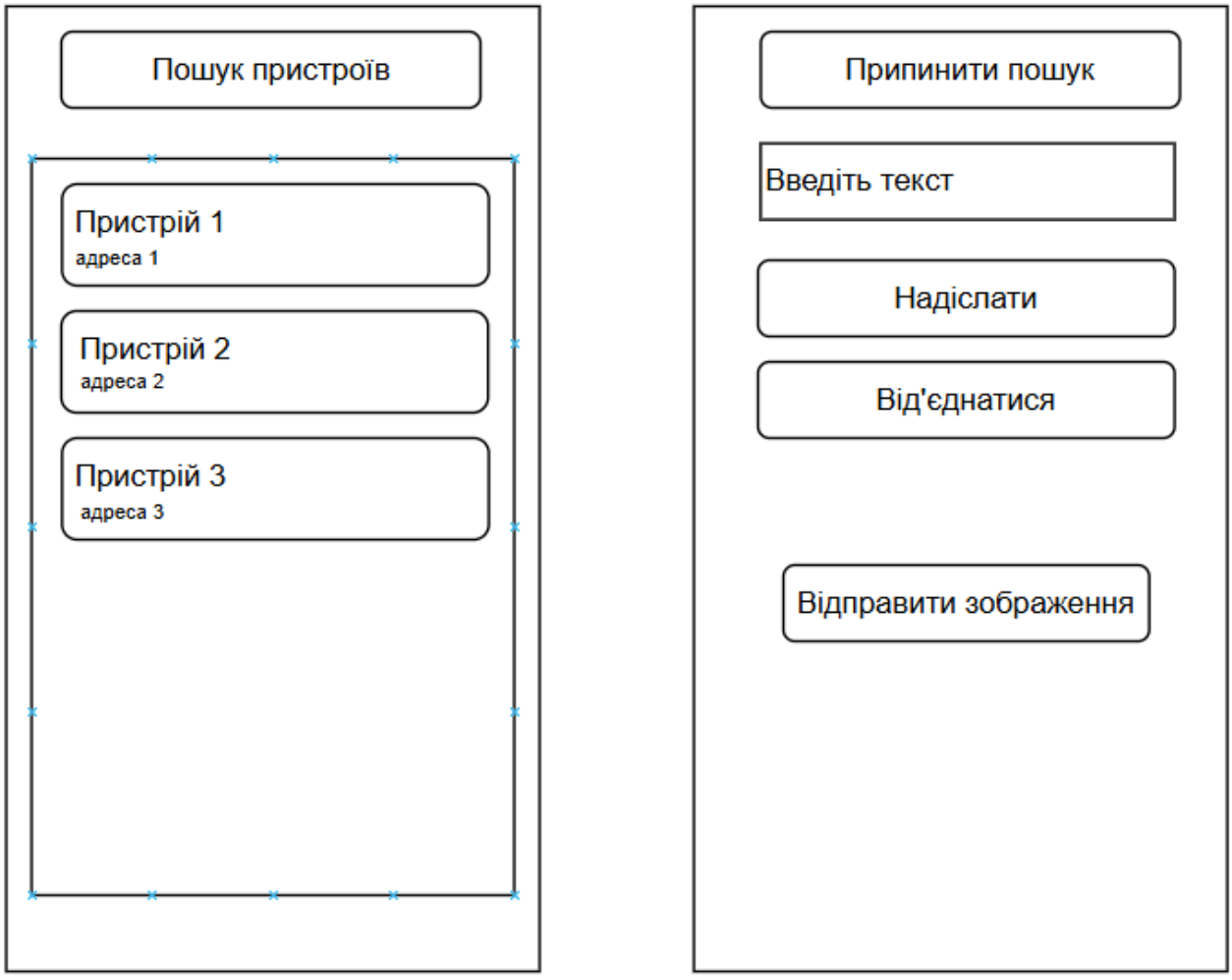


Рисунок 2.14 – Макети головного екрану

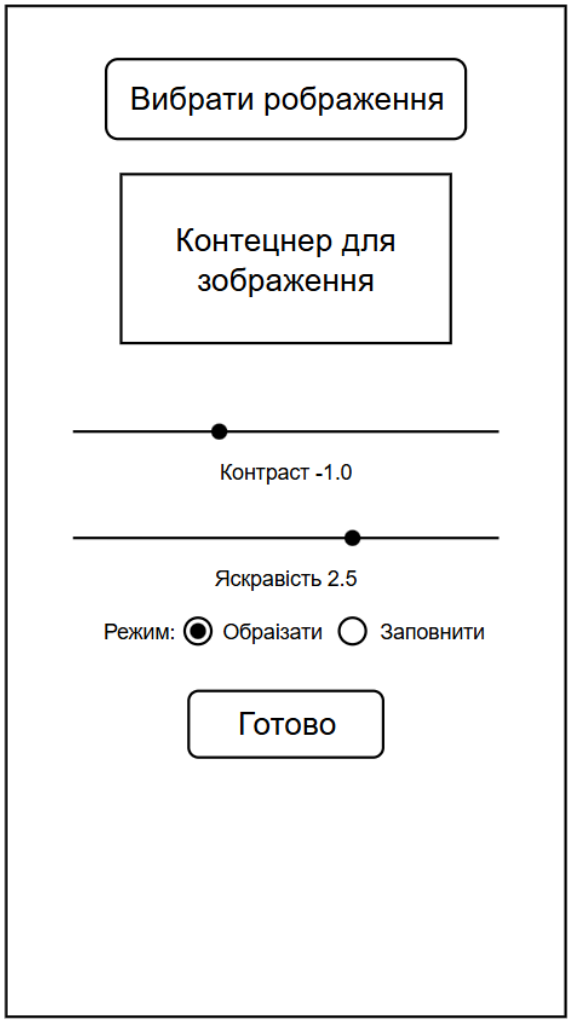


Рисунок 2.15 – Макет екрану редагування

2.6 Обґрунтування вибору технології дистанційної передачі даних

У розділі 1 ми вже коротко зачепили питання Bluetooth-технологій, а тепер час обрати та обґрунтувати технологію яка використовуватиметься в нашому проєкті.

Оскільки ми вже обрати мікроконтролер, логічно буде проводити аналіз технологій, відштовхуючись від його можливостей.

Мікроконтролер ESP32 має в своєму арсеналі два види підключень : Bluetooth та WI-FI. Наявність обох цих варіантів відкриває дуже великі можливості для експериментів та реалізації цікавих проєктів. Однак, оскільки за завданням дипломного проєкту від нас вимагається реалізація лише Bluetooth-підключення, а

також оскільки функціональність проєкту не вимагає надто складної дистанційної взаємодії та доступу в інтернет, було прийнято рішення використовувати лише Bluetooth-підключення. В майбутньому при бажанні можна модернізувати систему та використати можливості мікроконтролера на повну, а зараз зосередимось на виконанні задачі проєкту.

Варте уваги те, що лише обрати між Bluetooth та WI-FI не достатньо, адже ESP32 має цілих два режими роботи Bluetooth-підключення. А саме : Bluetooth Classic та Bluetooth Low Energy.

Щоб обрати який саме режим роботи ми використовуватимемо, потрібно проаналізувати кожен з них, виділити їх сильні та слабкі сторони, порівняти та співставити з вимогами проєкту.

Bluetooth Classic є старішою версією Bluetooth-з'єднань, її суть полягає в стабільній потоковій передачі даних. До того ж вона є досить швидкісною та дозволяє передавати до 3 Мбіт/с, що робить її підходящою для передачі та трансляції великої кількості даних, як от прослуховування музики через Bluetooth-навушники та колонки, передачу великих файлів. Одним з мінусів такого з'єднання є велике енергоспоживання через постійне з'єднання.

Bluetooth Low Energy (BLE) є більш новим рішенням. Воно має дещо меншу швидкість передачі даних, до 1 Мбіт/с, а також не підтримує потокову передачу інформації. Однак головною перевагою цього режиму є низьке енергоспоживання, яке досягається за рахунок періодичної передачі окремих невеликих пакетів інформації. В ті моменти коли немає передачі інформації, пристрій може залишатись в режимі сну. Таких підхід популярний і найбільш ефективний для фітнес-браслетів, IoT-пристроїв, та і в цілому там, де не вимагається постійна потокова передача даних.

З огляду на перелічені факти, можна скласти порівняльну таблицю (табл. 2.2), яка дозволить все структурувати, та остаточно вирішити яку технологію нам слід використовувати в проєкті.

Таблиця 2.2 – Порівняння характеристик Bluetooth-режимів

Характеристика	Bluetooth Classic	Bluetooth Low Energy (BLE)
Рік розробки	1999	2010
Максимальна швидкість	До 3 Мбіт/с (з EDR)	До 1 Мбіт/с
Дальність дії	До 100 м	До 100 м
Споживання енергії	Високе	Низьке
Час встановлення з’єднання	100–500 мс	~3 мс
Передача аудіо	Так	Ні
Профілі	SPP, A2DP, HFP тощо	GATT (Client/Server модель)
Топологія мережі	Piconet (1 master – до 7 slave)	Point-to-point, mesh
Призначення	Потокові дані, аудіо	Сенсори, IoT, періодична передача

З огляду на отримані висновки, кращим рішенням буде використання в проєкті технології Bluetooth Low Energy. Причиною цьому є те, що за завданням проєкту не вимагається передача великої кількості даних, а також не передбачається потокова передача. Нам необхідно передавати лише текст, та Bitmap-зображення, які достатньо розбити на невеликі пакети для успішної передачі.

2.7 Огляд програмної складової проєкту

Програмна частина прошивки для ESP32 передбачатиме використання стандартного ядра Arduino, що забезпечуватиме шаблон setup-loop і водночас відкриватиме доступ до планувальника FreeRTOS. Для обміну даними між

мікроконтролером і мобільним застосунком проєкт надалі спиратиметься на стек бібліотек BLEDevice, BLEUtils та BLEServer. Ці компоненти створять GATT-сервер із власним сервісом і однією характеристикою, завдяки чому передавання текстових рядків або пакетів растрового зображення відбуватиметься у межах типової процедури Write та не вимагатиме розробки спеціального транспортного протоколу.

Відображення отриманої інформації на OLED-екрані плануватиметься через комбінацію бібліотек Adafruit_GFX та Adafruit_SSD1306. Перша надаватиме універсальний набір графічних примітивів і базових шрифтів, тоді як друга виконуватиме зв’язок із конкретним контролером дисплея та інкапсулює послідовність команд, необхідних для оновлення буфера відеопам’яті.

У мобільному додатку роль «середовища виконання» виконуватиме стандартний Android-SDK, тож низькорівневий доступ до бездротового стека забезпечуватиметься напряму через пакети android.bluetooth та android.bluetooth.le.

Графічний інтерфейс буде реалізовано через Jetpack Compose — декларативний UI-фреймворк, який на рівні androidx.compose.runtime зберігає стан екрану у об’єктах MutableState. Це дозволить миттєво відображати зміну параметрів (наприклад, контрасту чи яскравості) без виклику імперативних методів onDraw.

Компоненти оформлення поставлені з бібліотеки Material 3: Card, Button, Slider та RadioButton сформують типовий «austere-UI», гарантуючи відповідність гайдлайнам Google щодо кольорів, типографіки й відступів.

Для взаємодії з системними діалогами буде використано контрактну модель ActivityResultContracts. Launcher, прив’язаний до RequestMultiplePermissions, єдиним викликом перевірятиме і запитуватиме BLE-дозволи, а окремий контракт StartActivityForResult запускатиме BitmapCreator та повертатиме оброблені зображення. Подібна схема прибирає потребу у застарілому методі onActivityResult і спростить життєвий цикл Activity.

Обробка зображень відбуватиметься всередині android.graphics.Bitmap: власна функція to1BitArray сконвертує пікселі у 8-розрядні блоки, а допоміжні процедури adjustImage, cropToAspect та addPadding — це чисті функції Kotlin, які користуються Color-API з пакета androidx.compose.ui.graphics. На час виконання всі перетворення відбуватимуться у робочому потоці; короткі паузи між записами на Gatt накладатимуться через клас Handler із головним Looper, завдяки чому стек BLE не переповниться навіть під час серії швидких write-операцій.

Таким чином, мобільний застосунок спиратиметься виключно на штатні Android-бібліотеки: BLE-стек забезпечить зв’язок, Jetpack Compose і Material 3 — реактивне UI-шарування, ActivityResultContracts — безпечну роботу з дозволами, а графічні пакети — локальну обробку бітмапів. Жодних сторонніх фреймворків чи нативних розширень NDK не передбачено, що спростить подальший супровід і унеможливить конфлікти ліцензій.

2.8 Висновки до розділу проєктування

Отже, в цьому розділі була попередньо зпроєктована кіберфізична система передачі даних на дисплей через Bluetooth-з’єднання. Ми проаналізували вимоги завдання проєкту, визначили необхідні компоненти системи, врахувавши її потреби. Також ми склали принципову схему роботи системи та схему підключень.

Для мобільного додатку була складена діаграма USeCase для визначення необхідного функціоналу, а також на її основі були складені макети екранів додатку.

Спираючись на результати цього розділу, далі ми займемось реалізацією цього проєкту за допомогою реальних засобів та фізичної плати, створимо працюючу систему яка буде вже не на теорії, а на практиці виконувати поставлені завдання.

3 РЕАЛІЗАЦІЯ МОБІЛЬНО-ОРІЄНТОВАНОЇ КІБЕРФІЗИЧНОЇ СИСТЕМИ З BLUETOOTH-ПІДКЛЮЧЕННЯМ ДЛЯ ДИСТАНЦІЙНОГО ВИВЕДЕННЯ ТЕКСТУ НА ЕКРАН

3.1 Постановка цілей на реалізацію проєкту

В цьому розділі ми приступимо до реалізації системи дистанційної передачі інформації, яку ми успішно зпроєктували в попередньому розділі. Тепер у нас є готові матеріали, на основі яких ми на практиці оберемо складові нашої системи.

Також ми покроково зберемо працюючу систему, виконавши всі необхідні з’єднання, а також виконаємо програмування мікроконтролера згідно вимог та кінцевих цілей нашого проєкту.

І звісно ми розробимо мобільний додаток, опираючись на попередні розмірковування щодо коректної його реалізації. Маючи за основу макети екранів, ми напишемо зручний повний додаток, роботу котрого ми перевіримо на практиці

3.2 Купівля необхідних компонентів

Зважаючи на попередні розрахунки та міркування, для реалізації системи потрібно придбати усі необхідні компоненти, а саме : мікроконтролерну плату ESP32, OLED-дисплей на базі мікроконтролера SSD1306, а також світлодіод та резистор.

Якщо замовляти усі компоненти через інтернет окремо, вартість доставки кожного компонента окремо може зробити систему невигідною з фінансових причин, тому варто розглянути можливість придбання усіх чи більшості компонентів в одному місці або купівлю готового набору з необхідними частинами.

Найкращим вибором може бути набір ESP Basic Starter Kit. Такий набір містить в собі саму плату ESP32, OLED-дисплей, набори світлодіодів, резисторів, конекторів типу male-male, male-female, female-female, декілька датчиків та кнопок, а також full-size макетну плату.

На маркетплейсі Rozetka (рис. 3.1) є такий набір, та його вартість складає близько 850 гривень. Це достатньо обґрунтована ціна за повноцінний набір, однак автор дипломного проєкту знайшов такий-же б/в варіант дещо дешевше на сайті оголошень OLX.

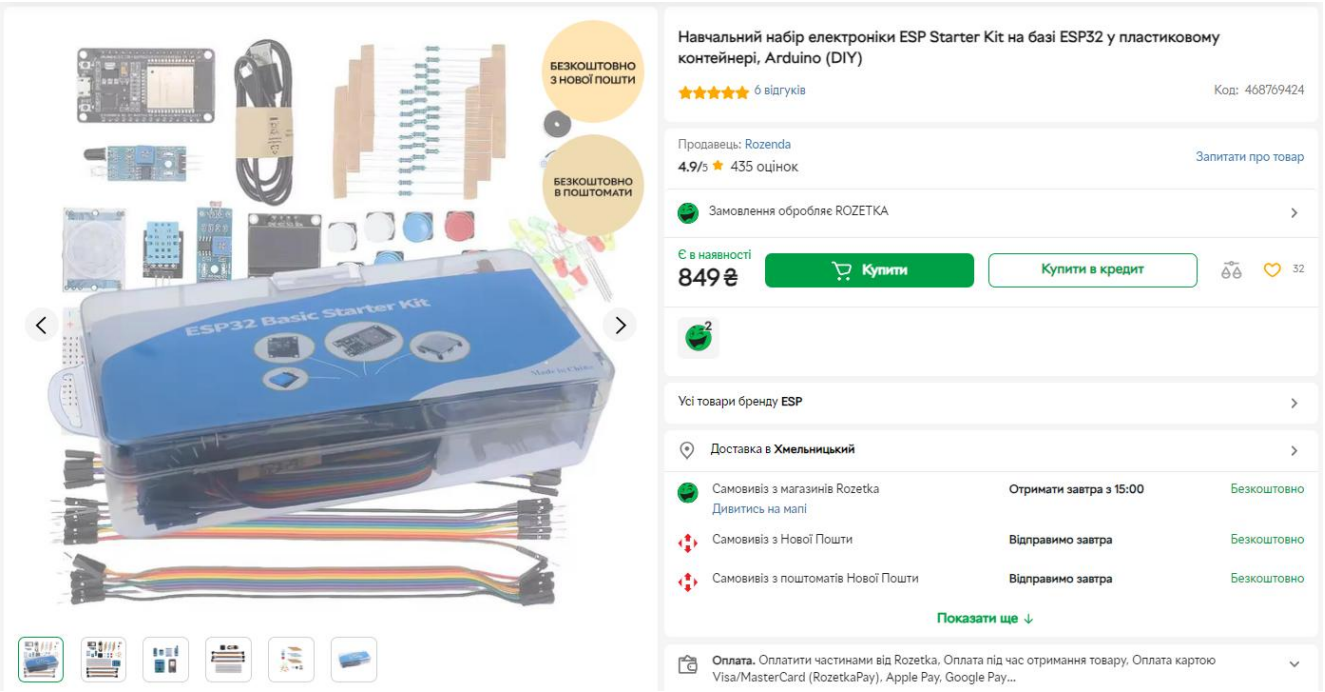


Рисунок 3.1 - ESP Basic Starter Kit на маркетплейсі Rozetka

3.3 Збірка фізичної частини системи

Отже, коли в нас є всі необхідні компоненти, можна розпочати збірку. В ході збірки ми оптимально розташуємо компоненти на платі, виконаємо всі з’єднання, а також виконаємо прошивання контролера. Після збірки ми виконаємо тестування за допомогою додатка nRF Connect, до дозволить впевнитися в справному функціонуванні системи.

Для початку потрібно приготувати макетну плату. Придбаний набір передбачає в собі full-size плату, її ми дістанемо та розташуємо на робочому місці (рис. 3.2).

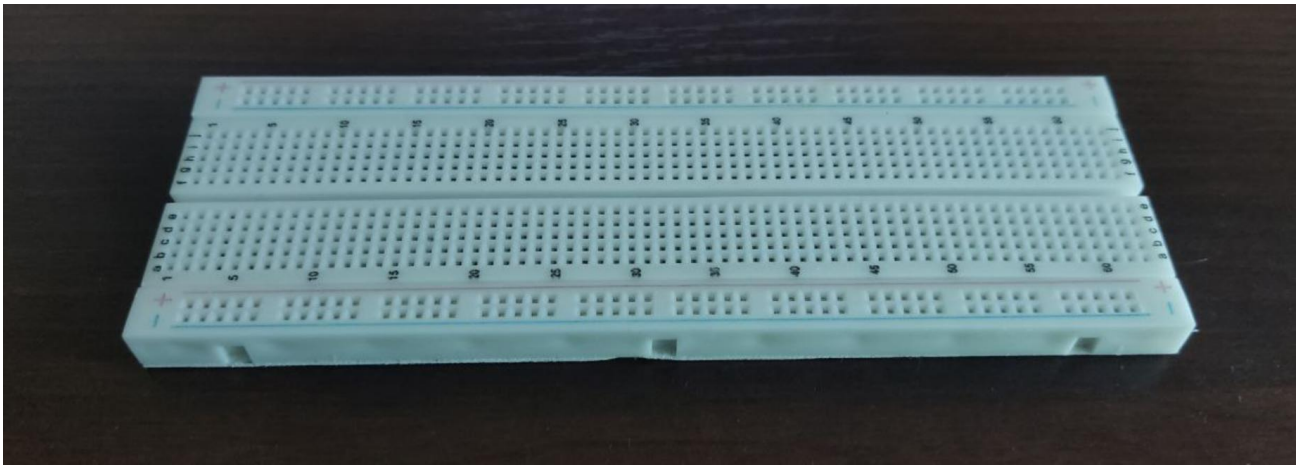


Рисунок 3.2 – Підготовлена макетна плата

Тепер потрібно визначитися з розташуванням плати ESP32. Варто зазначити, що компонент майже неможливо комфортно розташувати на макетній платі, оскільки ширина її стовпців є дещо меншою, аніж потрібно щоб були доступні усі піни мікроконтролера. Тобто, ESP32 можна розташувати так, що буде доступна для підключення лише одна її сторона.

Однак, якщо подивитись на розпінування плати, можна звернути увагу, що потрібні для нас місця підключень розташовані лише з однієї сторони плати. Це в свою чергу означає, що нас задовільнить таке розташування (рис 3.3). мікроконтролера на макетній платі.

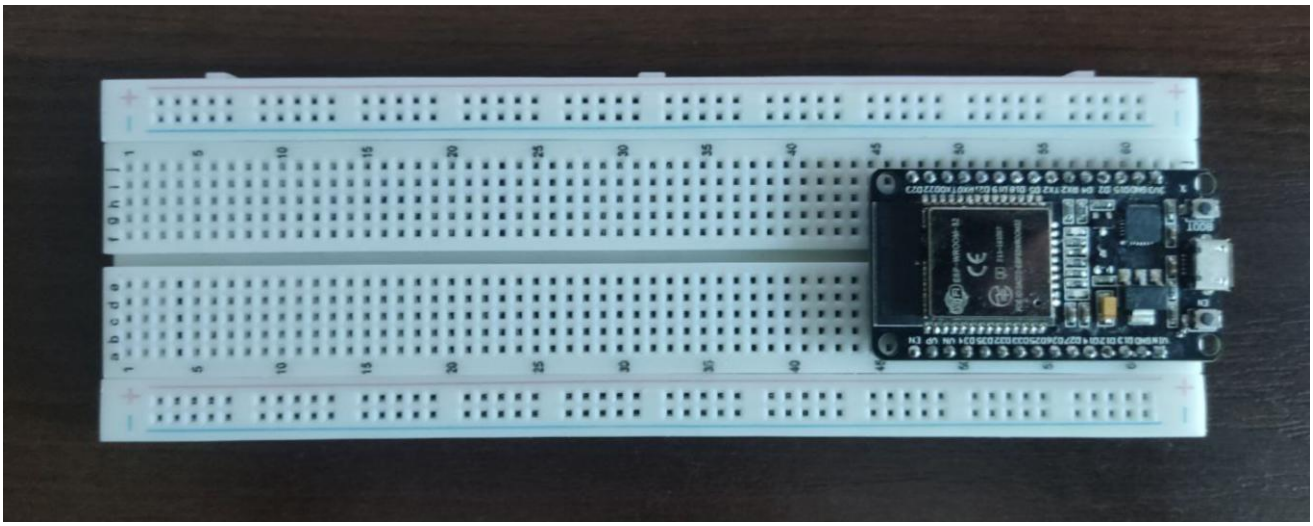


Рисунок 3.3 – Розташування мікроконтролера ESP32 на макетній платі

Наступним кроком буде встановлення OLED-дисплея на зону макетування. Потрібно розташувати екран так, щоб він не закривав потрібні доріжки, в також щоб компоненти не заважали одне одному на платі. Булу вирішено поставити дисплей таким чином, як на рисунку 3.4.

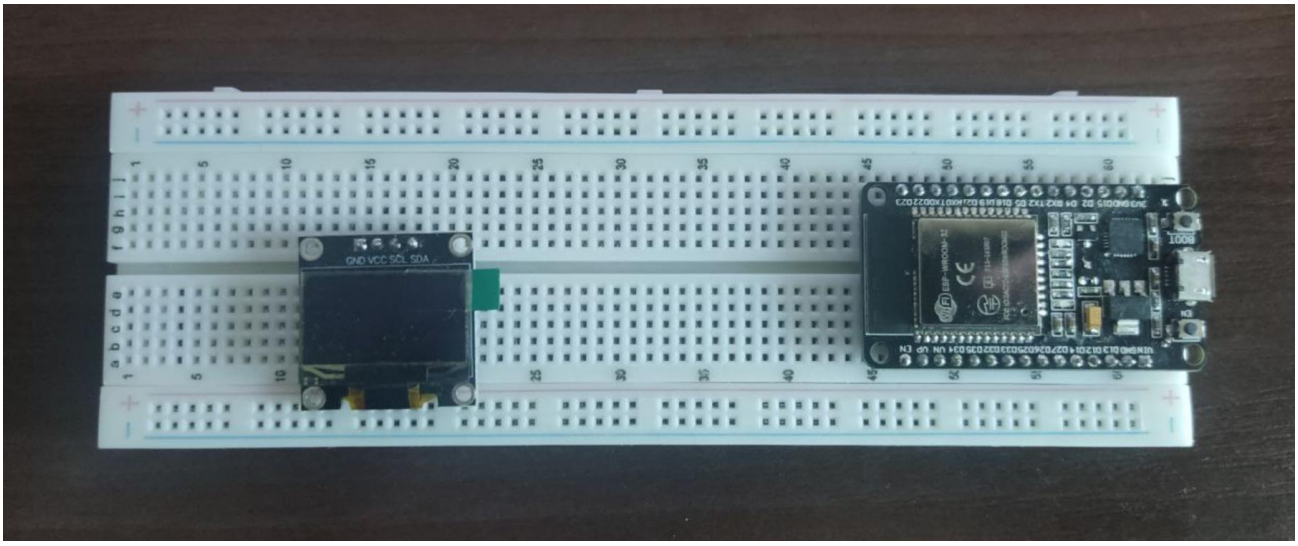


Рисунок 3.4 – Розташування OLED-дисплея на макетній платі

Тепер потрібно виконати підключення дисплею до мікроконтролера, для цього використовуватимемо конектори типу male-male. Для початку підключимо 3V3(60 доріжка, стовпець j) та GND(61 доріжка, стовпець j) мікроконтролера до живильних шин, аби забезпечити зручне паралельне з’єднання з іншими елементами. Тоді вже підключимо напругу та землю до дисплею, під’єднавши відповідно до VCC(16 доріжка, стовпець f) та GND(15 доріжка, стовпець f) дисплея.

Після цього підключимо SCL(17 доріжка, стовпець f) тактових імпульсів дисплею та SDA(18 доріжка, стовпець f) лінії передачі даних до відповідних пінів на ESP32. SCL до GPIO22(48 доріжка, стовпець j) та SDA до GPIO21(51 доріжка, стовпець j). Після виконаних операцій, підключення дисплею до мікроконтролера завершено (рис 3.5). Вже на цьому етапі можна прошивати ESP32 та перевіряти працездатність системи, але автор проєкту вирішив проводити тестування після остаточної збірки фізичної частини.

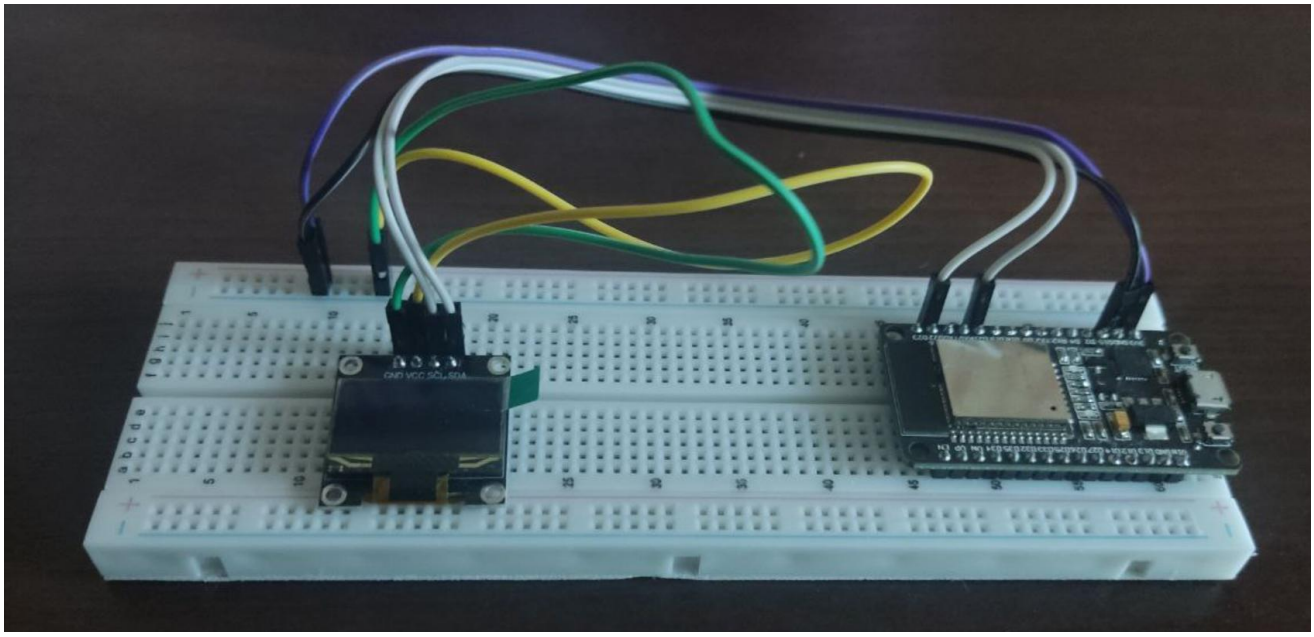


Рисунок 3.5 – Система після виконання підключень екрана до ESP32

Щоб довершити реалізацію фізичної частини проєкту, необхідно додати до системи світлодіод, який використовуватиметься в ролі індикатора Bluetooth-підключення. Для цього потрібно розташувати LED на макетній платі. Для цього встановимо анод на 40 доріжці стопця j, а катод на на мінусі живильної шини.

Також встановимо резистор на 220 Ом, аби врегулювати напругу яка йтиме на діод. Встановимо резистор на стопці f, на 40 та 42 доріжці.

Оскільки ми хочемо керувати діодом програмно, буде краще під’єднати його не до живильної шини, а до окремого GPIO на платі. Використаємо GPIO5 ESP32, підключивши клітинки на стовці j, в доріжках 42 та 51.

На рисунку 3.6 показаний кінцевий вигляд цієї частини проєкту. Після виконаної збірки та підключень маємо готову систему, яка готова до прошивання та подальшого тестування.

Також для більш естетичного та акуратного вигляду конструкції мохна зафіксувати конектори за допомогою стяжок, резинки, або будь-якого іншого фіксуючого засобу.

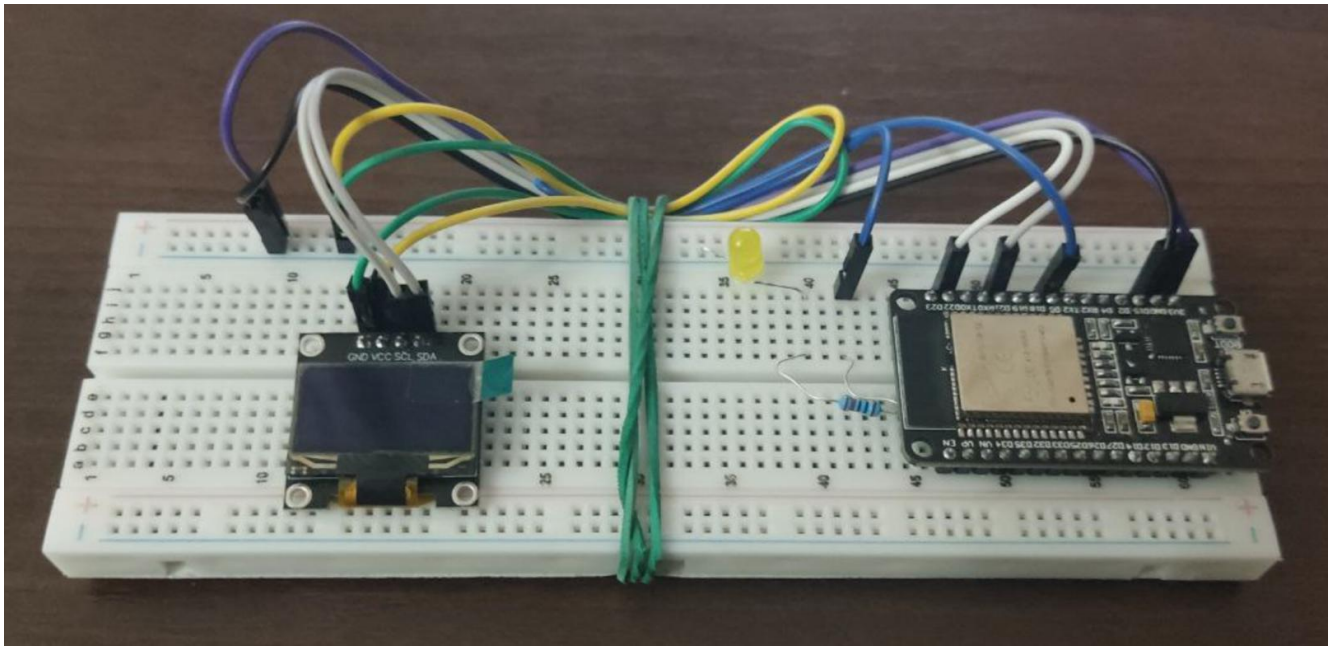


Рисунок 3.6 – Кінцевий вигляд фізичної частини проєкту

3.4 Прошивання мікроконтролера ESP32

Тепер можна перейти до програмування системи. Для цього використовуватимемо середовище розробки Arduino IDE. Але перед написанням коду необхідно підготувати середовище розробки, виконавши декілька нескладних кроків.

По-перше, потрібно додати посилання на плату щоб IDE могло її бачити. Для цього в Arduino IDE потрібно натиснути в верхньому меню File > Preferences, а потім в полі «Additional board manager URLs» потрібно ввести посилання: https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json (рис. 3.7)

Рисунок 3.7 – Правильний вигляд поля менеджера плат

Далі необхідно перейти в менеджер плат, та виконати встановлення ESP32. Це робиться або через бокове меню, або через Tools > Board > Board Manager.

В цьому меню потрібно встановити «esp32 by Espressif Systems» (рис 3.8).

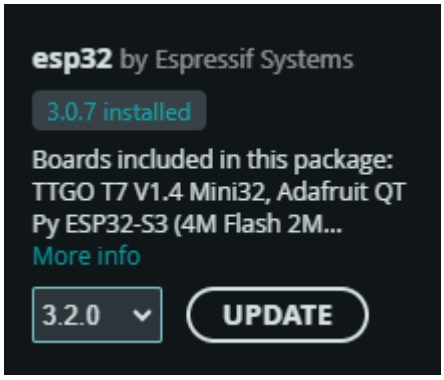


Рисунок 3.8 – esp32 by Espressif Systems в менеджері плат

Тепер можна обрати нашу плату та порт. В даному проєкті використовуватиметься плата «ESP32 Dev Module» та порт COM5.

Отже перейдемо до програмування. Вся програма буде написана в «.ino», тобто на мові C++ але адаптованій під мікроконтролери. На початку підключимо необхідні бібліотеки для роботи з Bluetooth Low Energy (BLE), OLED-дисплеєм SSD1306 і шиною I2C: це BLEDevice.h, BLEUtils.h, BLEServer.h, Adafruit_GFX.h, Adafruit_SSD1306.h та Wire.h.

Далі в коді визначаємо розміри дисплея (128x64 пікселів) та константу скидання (OLED_RESET), яка встановлена в -1, оскільки в даній моделі дисплея фізичного піну скидання не передбачено. Створимо об'єкт display, через який відбуватиметься вся взаємодія з дисплеєм. Також оголосимо змінні для збереження буферу зображення (масив із 1024 байтів), лічильника отриманих байтів, а також прапорець, який вказує, чи йде зараз прийом зображення.

Наступним задаємо пін «ledPin», до якого підключено світлодіод для індикації підключення до Bluetooth, тобто пін GPIO5. Також ініціалізуємо змінні, які будуть зберігати об’єкт Bluetooth-сервера та характеристику для прийому даних, а також логічну змінну «deviceConnected», що сигналізуватиме, чи є підключення до пристрою.

Після цього визначимо UUID для Bluetooth-сервісу та характеристики, які мають бути унікальними, щоб клієнт (мобільний додаток) міг знайти та взаємодіяти з пристроєм. Для сервісу ID «4fafc201-1fb5-459e-8fcc-c5c9c331914b», а для характеристики ID «beb5483e-36e1-4688-b7f5-ea07361b26a8»

Тепер реалізуємо два класи з callback-функціями. Перший - MyServerCallbacks, який реагуватиме на підключення та відключення пристрою. При підключенні встановлюватиметься відповідний прапорець deviceConnected = true, а також вмикнеться світлодіод. При відключенні — прапорець скидається, світлодіод вимикається, а також знову запускається реклама getAdvertising() Bluetooth-сервісу, щоб пристрій був знову видимим.

Другий клас - MyCharacteristicCallbacks, у якому реалізуємо метод onWrite. У цьому методі оброблятимуться отримані дані. Спочатку зчитуватиметься вміст і довжина пакету. Якщо отримані дані починаються з префіксу "TXT:", це означає, що користувач надіслав текстове повідомлення. У цьому випадку текст очищається від префікса та виводиться на дисплей.

У випадку, якщо префіксу "TXT:" немає, це означає, що надіслано фрагмент зображення. Якщо це перший фрагмент, то активується режим прийому зображення й скидається лічильник. Дані послідовно записуватимуться в буфер, доки не буде зібрано всі 1024 байти. Коли буфер заповниться, зображення виведеться на дисплей за допомогою методу drawBitmap, після чого прапорець прийому й лічильник скинуться.

У функції setup() відбудеться ініціалізація послідовного порту для діагностики, піна світлодіода, а також самого дисплея. У разі якщо не вдасться ініціалізувати дисплей, програма зупиниться. Далі на дисплей виведеться повідомлення про очікування з'єднання по Bluetooth.

Ініціалізація Bluetooth починається з BLEDevice::init(...), де задається ім’я пристрою. Створюється сервер, до якого прив’язуються колбеки підключення. Далі створюється сервіс з UUID і характеристика для прийому даних. До характеристики додається можливість запису (PROPERTY_WRITE) і запису без

відповіді (PROPERTY_WRITE_NR), а також прив'язується клас обробки вхідних даних. Сервіс запускається, після чого конфігурується реклама: пристрій повідомляє про наявність сервісу та налаштовує параметри реклами.

На завершення, в setup() стартує реклама, що дозволить підключатися до пристрою з мобільного застосунку, а також виводиться повідомлення в консоль про те, що пристрій очікує з’єднання.

У циклі loop() перевіряється, чи є підключення. Якщо з’єднання втрачено, реклама Bluetooth-сервісу перезапускається з інтервалом у 5 секунд. Це дозволяє знову знайти пристрій, якщо користувач випадково відключився або вийшов за межі досяжності.

Тепер програма готова до завантаження на мікроконтролер ESP32. Для цього під’єднаємо micro-USB до контролера, а USB до обраного в Arduino IDE роз’єму для завантаження програми. Після цього натискаємо Upload для Запуску процесу компіляції коду та прошивання плати (рис. 3.9).

```
Sketch uses 1169265 bytes (89%) of program storage space. Maximum is 1310720 bytes.
Global variables use 40344 bytes (12%) of dynamic memory, leaving 287336 bytes for local variables. Maximum is 327680 bytes.
esptool.py v4.6
Serial port COM5
Connecting.....
Chip is ESP32-D0WD-V3 (revision v3.1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: cc:db:a7:3d:f7:b0
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 921600
Changed.
Configuring flash size...
Flash will be erased from 0x00001000 to 0x00007fff...
Compressed 24896 bytes to 16262...
Writing at 0x00001000... (100 %)
Wrote 24896 bytes (16262 compressed) at 0x00001000 in 0.6 seconds (effective 339.2 kbit/s)...
Hash of data verified.
Compressed 3072 bytes to 146...
Writing at 0x00008000... (100 %)
Wrote 3072 bytes (146 compressed) at 0x00008000 in 0.0 seconds (effective 505.0 kbit/s)...
Hash of data verified.
Compressed 8192 bytes to 47...
Writing at 0x0000e000... (100 %)
Wrote 8192 bytes (47 compressed) at 0x0000e000 in 0.1 seconds (effective 650.7 kbit/s)...
Hash of data verified.
Compressed 1175840 bytes to 745133...
Writing at 0x00010000... (2 %)
Writing at 0x0012c687... (100 %)
Wrote 1175840 bytes (745133 compressed) at 0x00010000 in 11.9 seconds (effective 787.3 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

Рисунок 3.9 – Інформація з консолі про успішне виконання прошивання плати

На цьому етапі роботи вже можна перевірити успішність збірки та програмування системи. Використовуючи раніше згадану програму nRF Connect, ми виконаємо підключення мобільного пристрою до плати та перевіримо можливість передачі тексту.

Для початку зайдемо в додаток, та натиснемо «Scan» в правому верхньому куті. Далі, коли в списку з’явиться пристрій «ESP32_Display» натискаємо «Connect». В результаті (рис 3.10) світлодіод засвітився, що свідчить про те, що нам вдалося підключитися до плати.

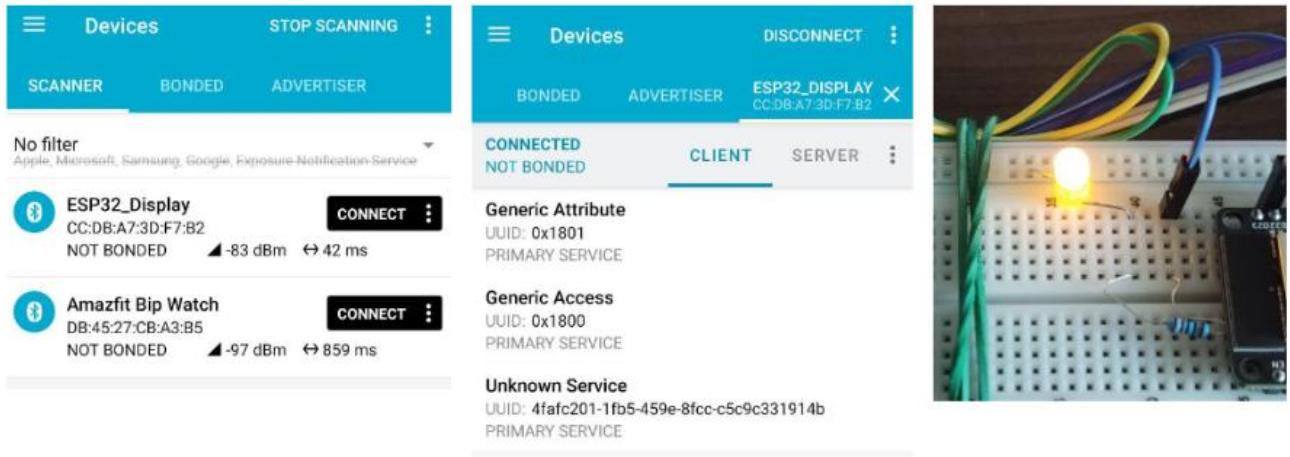


Рисунок 3.10 – Результат підключення через nRF Connect

Тепер потрібно спробувати передати інформацію на дисплей. nRF Connect має передбачені можливості для тестування характеристик в BLE, і це значно спрощує нам життя. Спочатку натискаємо на «Unknown Service», після цього на екрані з’явиться «Unknown Characteristic»(рис 3.11).

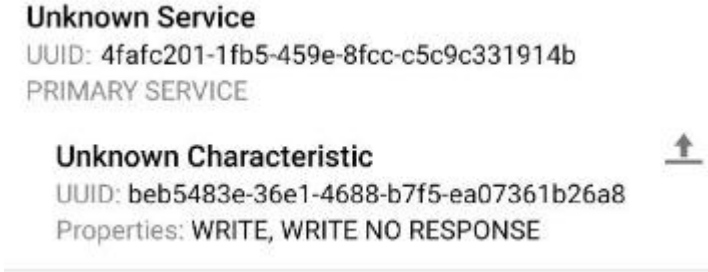


Рисунок 3.11 – Характеристика пристрою в nRF Connect

Натиснувши на іконку Upload справа зверху характеристики, в нас з’явиться меню її редагування. В першу чергу, для коректної передачі тексту виберемо текстовий тип даних (TEXT UTF-8). Після цього в поле для значення характеристики впишемо текст для тесту. Для коректної роботи необхідно на початку приписати «TXT:», щоб мікроконтролер зміг правильно обробити дані. Тепер напишемо «Test1, hi there» та натиснемо «SEND»(рис. 3.12).

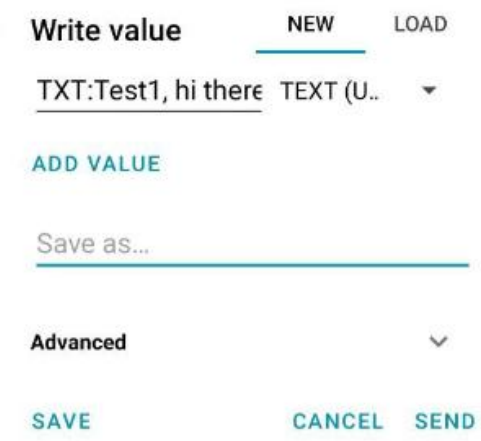


Рисунок 3.12 – Налаштування характеристики перед відправкою

Результати успішного тестування зображено на рисунку 3.12.

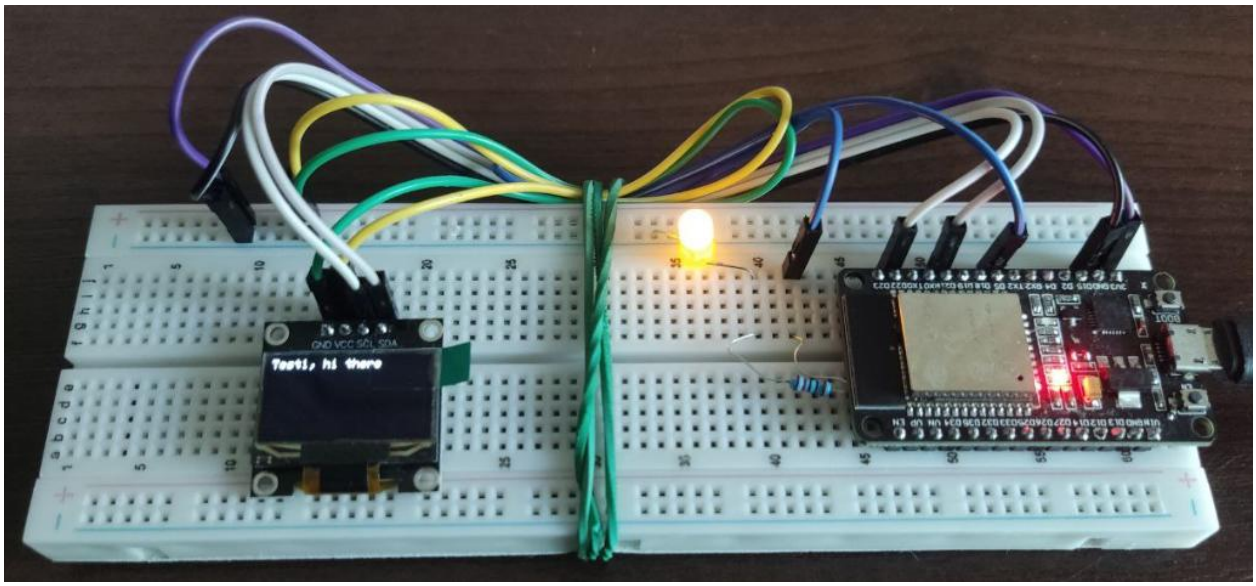


Рисунок 3.12 – Результат тестування через програму nRF Connect

3.5 Розробка мобільного додатку

Хоча nRF Connect добре показав себе при попередньому тестуванні системи, очевидно що його можливостей недостатньо для реалізації запланованих завдань проєкту. Ми можемо відносно добре передавати текстову інформацію, але якщо справа стосується зображень, то це рішення не є підходящим. Звісно, якщо прикласти певних зусиль, ми дійсно могли б здійснити відправку зображення через попередню окрему генерацію бітмап-масиву та відправку його у відповідному форматі, але це рішення є занадто багатошаровим та непрактичним. До того ж при роботі з текстом необхідно буде щоразу ставити префікс «TXT:», що також не є практичним.

З огляду на це, логічним рішенням буде розробка власного додатку, що задовільнить потреби конкретно нашого проєкту.

В попередньому розділі ми вже провели проєктування нашого додатка, і тепер нам необхідно зайнятися його реалізацією. Для цього використовуватимемо середовище розробки Android Studio, створене саме для написання мобільних додатків під операційну систему Android.

На початку потрібно зробити базові речі, а саме створити та налаштувати проєкт. Оскільки ми використовуватимемо декларативний підхід в розробці інтерфейсу, при створенні проєкту відразу виберемо шаблон для додатку з Jetpack Compose (рис 3.13). Це дозволить скоротити час налаштування.

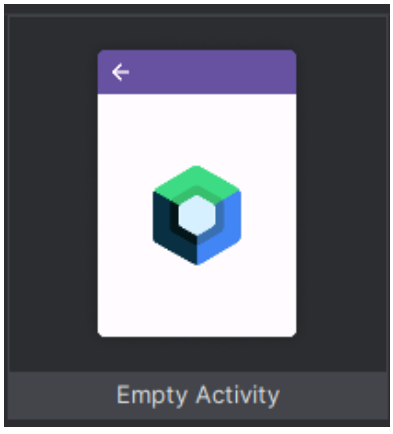


Рисунок 3.13 – Шаблон для додатку на Jetpack Compose

Після вибору шаблону потрібно обрати деякі властивості проєкту, такі як назва, пакет, місце збереження та мінімальна підтримувана версія Android.

Назвемо проєкт «ESP32 Text sender via Bluetooth», а решту інформації залишимо за замовчуванням, включно з мінімальною версією, при необхідності її можна змінити в налаштуваннях Gradle. На рисунку 3.14 зображені налаштування проєкту.

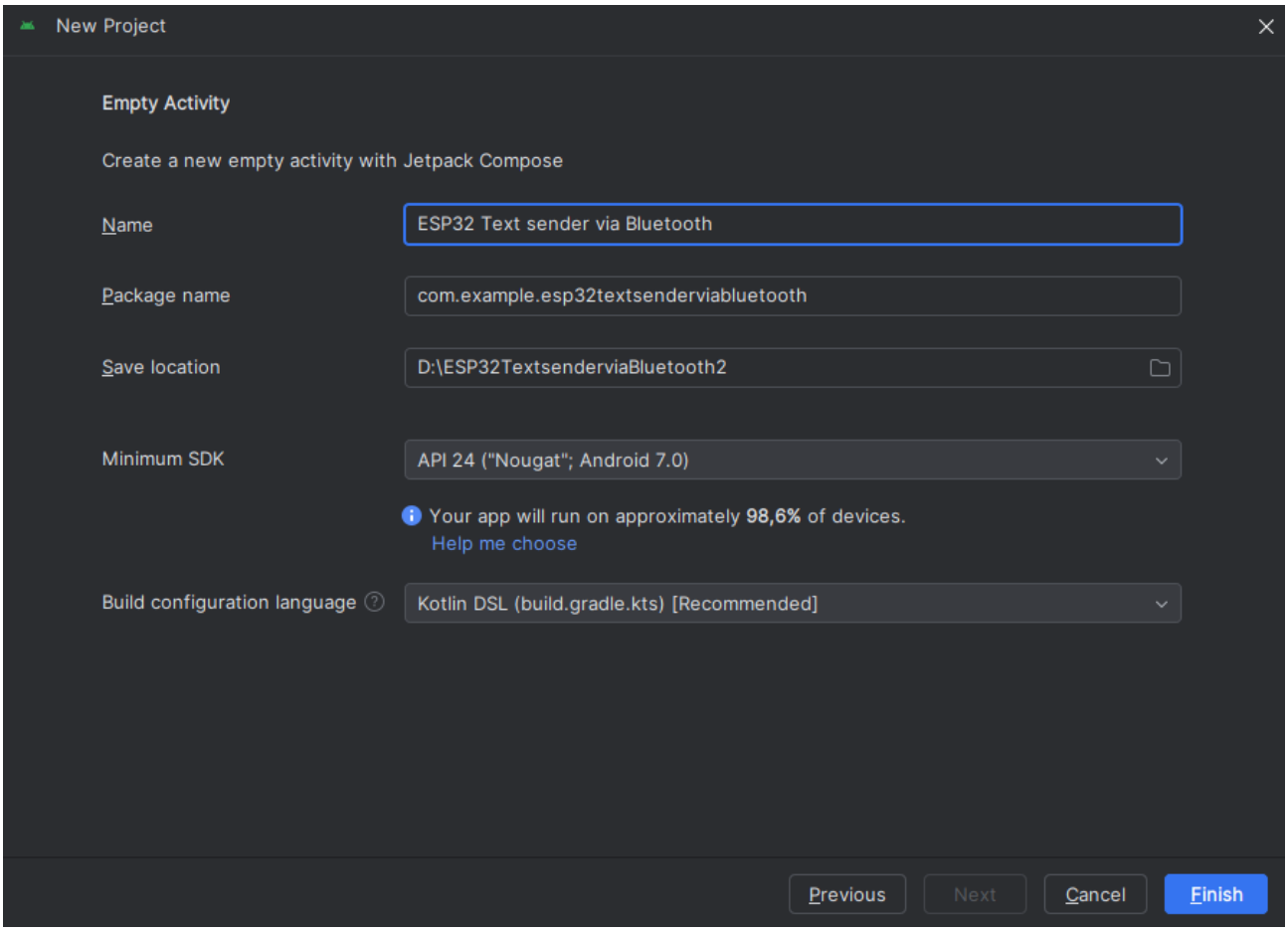


Рисунок 3.14 – Налаштування при створенні проєкту

Тепер можна приступити до написання коду. Першим кроком у реалізації є створення окремого класу BleScanner, який відповідає за всю логіку Bluetooth Low Energy-з'єднання між Android-додатком і мікроконтролером ESP32. У конструкторі цього класу виконується ініціалізація Bluetooth-менеджера та адаптера через системний сервіс Context.BLUETOOTH_SERVICE. Додатково ініціалізується BLE-сканер, GATT-контролер та задаються значення UUID сервісу

й характеристики, які мають відповідати конфігурації на стороні ESP32. Ці UUID задаються у вигляді констант SERVICE_UUID та CHARACTERISTIC_UUID.

Наступним етапом є реалізація механізму пошуку доступних BLE-пристроїв. Для цього в класі створюється об'єкт scanCallback на основі анонімного класу ScanCallback, де перевизначається метод onScanResult. У цьому методі кожен знайдений пристрій перевіряється за MAC-адресою на унікальність і, у разі її відсутності в поточному списку, додається до змінної devices, яка є обгорнутою у mutableStateOf, що дозволяє автоматично оновлювати інтерфейс при зміні. Запуск сканування виконується методом startScan, який ініціює bleScanner.startScan(scanCallback) та передає статус у функцію зворотного зв’язку onScanStateChanged. Зупинка сканування реалізується методом stopScan.

Після вибору користувачем пристрою реалізовується підключення до нього за допомогою методу connectToDevice. Перед початком нового з’єднання виконується перевірка поточного стану, а також викликається метод disconnect для гарантованого завершення попередніх сесій. Щоб уникнути помилки GATT 133, перед викликом connectGatt реалізовується затримка в 300 мілісекунд за допомогою handler.postDelayed. Залежно від версії Android, викликається відповідний варіант методу connectGatt, у якому вказується використання Low Energy-підключення через BluetoothDevice.TRANSPORT_LE.

Подальша робота з GATT-з’єднанням виконується за допомогою обробника gattCallback. У методі onConnectionStateChange перевіряється результат підключення. У разі успіху змінюються значення станів isConnected та isConnecting, після чого викликається requestConnectionPriority для встановлення високого пріоритету з’єднання. Після цього, із затримкою у 600 мс, викликається метод discoverServices, який починає пошук доступних GATT-сервісів на пристрої.

У методі onServicesDiscovered здійснюється пошук GATT-сервісу за UUID. Якщо сервіс знайдено, з нього отримується характеристика за UUID CHARACTERISTIC_UUID, яка зберігається у змінній txCharacteristic. Якщо ж сервіс не буде виявлено, викликається метод disconnect, що скидає стан з’єднання.

Для передачі текстових повідомлень передбачимо метод sendText, який формує байтовий масив, додаючи до введеного тексту префікс «TXT:». Отримані дані присвоюємо характеристиці, після чого викликається метод writeCharacteristic, який передає повідомлення на пристрій. Використання префіксу дозволяє ESP32 розпізнати тип вхідної інформації.

Для надсилання зображень реалізовується метод sendBitmap, який приймає бітмап у вигляді масиву байтів. Зображення розбивається на блоки по 20 байт - максимальний розмір, який допускається при стандартному MTU без додаткових налаштувань. Передача здійснюється з параметром WRITE_TYPE_NO_RESPONSE, що дозволяє уникати зворотного підтвердження для кожного пакета і підвищити швидкість. Щоб уникнути перевантаження BLE-стека, між записами вводиться затримка 20 мс.

Для завершення з’єднання створюється метод disconnect, який відповідає за коректне закриття GATT-сеансу. У цьому методі викликаються disconnect() і close() на об'єкті bluetoothGatt, після чого скидаються всі пов’язані змінні стану. Це дозволяє уникнути ресурсних витоків і забезпечує стабільну роботу додатка при багаторазових підключеннях.

Після створення класу для Bluetooth-з’єднання ми переходимо до написання логіки, що забезпечує коректну взаємодію з Bluetooth-модулем з боку користувача.

Насамперед у MainActivity реалізовується перевірка та запит дозволів, необхідних для роботи з BLE-пристроями. Оскільки набір необхідних permission'ів змінюється залежно від версії Android, ми формуємо динамічний список requiredPermissions, до якого додаємо відповідні значення залежно від API. Наприклад, для старіших версій Android додається дозвіл на визначення місцеположення, тоді як для новіших використовуються BLUETOOTH_SCAN та BLUETOOTH_CONNECT. Цей список перетворюється у масив і передається в систему через ActivityResultLauncher, який ми створюємо за допомогою контракту RequestMultiplePermissions.

Після запуску цього лаунчера перевіряється, чи всі дозволи були надані, і якщо так — викликається метод checkBluetooth, який відповідає за подальшу ініціалізацію модуля. Якщо ж користувач відмовляється надати необхідні дозволи, виводиться відповідне повідомлення через Toast, і сканування пристроїв не запускається.

Наступним кроком ми реалізовуємо перевірку активності самого Bluetooth-модуля. У методі checkBluetooth послідовно перевіряється, чи підтримується Bluetooth на пристрої, а також чи він увімкнений. Якщо модуль відсутній або вимкнений, виводиться повідомлення про помилку. У разі, коли модуль є, але знаходиться в неактивному стані, ініціюється стандартний системний запит на його увімкнення. Для цього ми створюємо ще один ActivityResultLauncher, цього разу з контрактом StartActivityForResult, і викликаємо його з інтентом BluetoothAdapter.ACTION_REQUEST_ENABLE. Після завершення цього запиту, у випадку позитивної відповіді, ми негайно запускаємо процес сканування. Якщо ж користувач відмовляється вмикати Bluetooth, робота додатку в цій частині припиняється.

Коли всі дозволи надані, а Bluetooth-модуль активний, ми переходимо до реалізації механізму сканування BLE-пристроїв. Цей процес запускається викликом методу startScanning, в якому ми спочатку очищаємо поточний список пристроїв devices, щоб уникнути дублювання, а потім ініціюємо виклик методу startScan у BleScanner.

Паралельно зі скануванням, у користувацькому інтерфейсі активується індикатор прогресу, а також виводиться лічильник кількості знайдених пристроїв. Користувач має можливість у будь-який момент зупинити сканування за допомогою відповідної кнопки, що викликає метод stopScanning.

Під час сканування всі знайдені пристрої, які ще не присутні у списку, додаються до колекції devices у BleScanner, після чого вони автоматично відображаються у вигляді списку в UI. За відображення списку відповідає компонент LazyColumn, а кожен пристрій оформлюється як картка з назвою, MAC-

адресою та, за необхідності, позначкою активного підключення. Таким чином реалізовується повноцінний процес виявлення доступних BLE-пристроїв, який враховує усі необхідні перевірки та забезпечує зручний візуальний інтерфейс для користувача.

Переходячи до побудови інтерфейсу користувача, будемо використовувати сучасний підхід Jetpack Compose. Це дозволяє створювати інтерфейс декларативно, без використання XML-файлів. Такий підхід значно спрощує структуру коду і покращує читабельність, оскільки вся логіка відображення елементів зосереджена безпосередньо в Kotlin-функціях.

У головній активності MainActivity ми ініціалізуємо інтерфейс через виклик setContent, у якому вказуємо функцію AppUI. Вона містить всю візуальну логіку, яку бачить користувач під час роботи з додатком. Одразу ж на початку ми визначаємо, в якому саме стані знаходиться програма: чи встановлено з’єднання з ESP32, чи ще ні. Від цього залежить, що саме буде виведено на екран.

Якщо підключення ще не встановлено, ми показуємо список пристроїв (рис. 3.15), знайдених під час сканування. Для цього ми створюємо функцію DeviceList, у якій використовується LazyColumn - стандартний компонент для побудови прокручуваних списків. Кожен знайдений пристрій ми передаємо до функції DeviceItem, яка відображає назву пристрою, MAC-адресу, а також — якщо пристрій уже підключено — позначку про активне з’єднання. При натисканні на пристрій викликається функція connectToDevice, яка ініціює підключення до вибраного елемента.

Окремо ми реалізовуємо ще одну функцію ScanControls, у якій розміщується кнопка запуску або зупинки процесу сканування. Вона реагує на стан змінної isScanning і змінює свій вигляд відповідно до того, що відбувається: якщо сканування триває, кнопка показує напис «Зупинити», і навпроти неї з’являється прогрес-індикатор. Додатково виводиться кількість знайдених пристроїв.

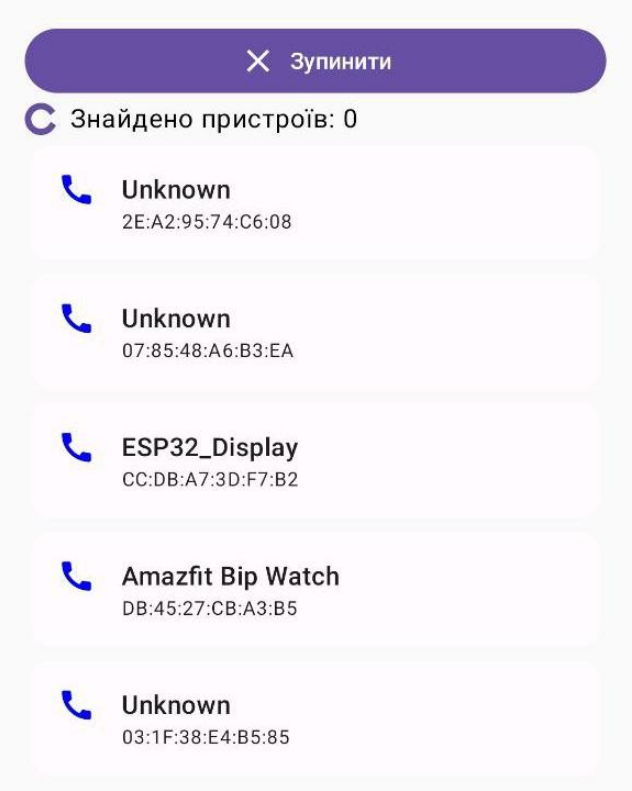


Рисунок 3.15 – Список пристроїв, знайдених під час сканування

Як тільки встановлено з’єднання з ESP32, ми замінюємо вміст екрана на панель передачі даних (рис 3.16). Цю частину ми реалізовуємо у функції TransferPanel. Тут розміщується текстове поле для вводу повідомлення, кнопка для його надсилання, а також кнопка для завершення з’єднання. При натисканні на «Send» викликається метод sendText, який передає текст через Bluetooth у вигляді байтового масиву з префіксом. Водночас, натискання кнопки «Disconnect» замикає цикл сесії через виклик функції disconnect.

Окрім передачі тексту, ми додаємо в інтерфейс кнопку «Редагувати зображення», яка відкриває окрему активність для взаємодії з графікою. Ми створюємо Intent, передаємо його через ActivityResultLauncher і чекаємо результат. Після завершення редагування ми отримуємо зображення у вигляді об’єкта Bitmap, який далі буде конвертовано у 1-бітовий масив і передано на ESP32. Для цього використовується функція to1BitArray, а передача здійснюється через sendBitmap. У разі успіху або помилки користувач отримує відповідне повідомлення, що дозволяє краще орієнтуватися у процесі.

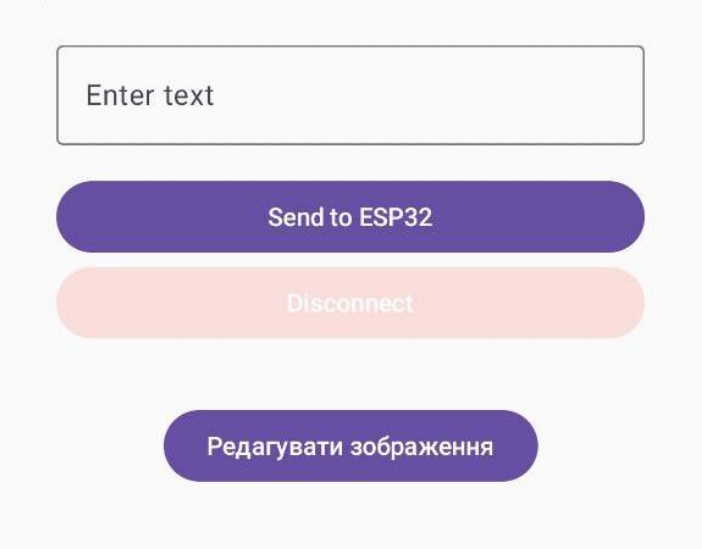


Рисунок 3.16 – Панель передачі даних

Наступним етапом розробки є створення інтерфейсу редагування зображення. Для цього ми реалізовуємо окрему активність під назвою BitmapCreator, яка відповідає за повний цикл обробки зображення перед його передачею на пристрій.

Усередині активності ми викликаємо метод setContent, в якому задаємо основний composable-функціонал - ImageEditScreen. У цьому екрані реалізовано всі елементи, необхідні для взаємодії з користувачем: вибір зображення з галереї, налаштування параметрів (яскравість, контрастність, колір фону, режим обрізки), попередній перегляд та застосування змін.

Одразу після відкриття BitmapCreator користувачеві пропонується вибрати зображення. Для цього ми створюємо ActivityResultLauncher із контрактом ActivityResultContracts.GetContent(), який дозволяє обрати файл з галереї. Лаунчер зберігається в змінній galleryLauncher, а обробка результату відбувається у вигляді URI, який ми одразу конвертуємо в об’єкт Bitmap через MediaStore.Images.Media.getBitmap. Отриманий Bitmap зберігається у стані originalBitmap, з яким ми далі й працюємо.

Одразу після завантаження ми створюємо ще одну змінну стану - processedBitmap, яка зберігає поточну оброблену версію зображення. Цей об'єкт

оновлюється щоразу при зміні будь-якого параметра - таким чином, у користувача є можливість бачити результат редагування в реальному часі.

Щоб керувати параметрами редагування, ми визначаємо кілька змінних стану: contrast, brightness, cropMode та bgColor. Змінні contrast і brightness представляють значення яскравості та контрастності й контролюються через повзунки (слайдери). cropMode - це булевий прапорець, який вказує, чи потрібно обрізати зображення до співвідношення 2:1, чи додавати відступи. Змінна bgColor містить колір фону у вигляді Color, який додається по краях зображення у випадку, якщо обрізка вимкнена.

Для обробки зображення ми реалізовуємо функцію processBitmap, яка послідовно викликає низку допоміжних функцій. Передусім, ми передаємо originalBitmap, contrast та brightness у функцію adjustImage. Ця функція проходить по всіх пікселях зображення та змінює значення RGB на основі формули, яка враховує контраст і яскравість. Після цього оновлений Bitmap передається в одну з двох наступних функцій - cropToAspect або addPadding, залежно від значення cropMode.

У режимі обрізки (cropMode == true) ми викликаємо функцію cropToAspect, яка розраховує центральну область зображення з фіксованим співвідношенням 2:1, після чого масштабує її до розміру 128x64 пікселів. Якщо ж користувач обрав режим збереження повного зображення, ми використовуємо функцію addPadding, яка спершу масштабує зображення по меншій стороні, а потім розміщує його на новому полотні (canvas) розміром 128x64 пікселів, додаючи фон із кольору bgColor. При цьому ми центровано розміщуємо зображення по обох осях.

У результаті кожного такого оновлення processedBitmap зберігає повністю оброблену версію зображення, яку ми показуємо користувачеві у вигляді прев'ю на екрані. Користувач може повторно змінювати будь-який параметр, і кожного разу ми викликаємо processBitmap, оновлюючи попередній перегляд.

На рисунку 3.17 зображено екран до вибору зображення, після, а також вигляд зображення після масштабування.

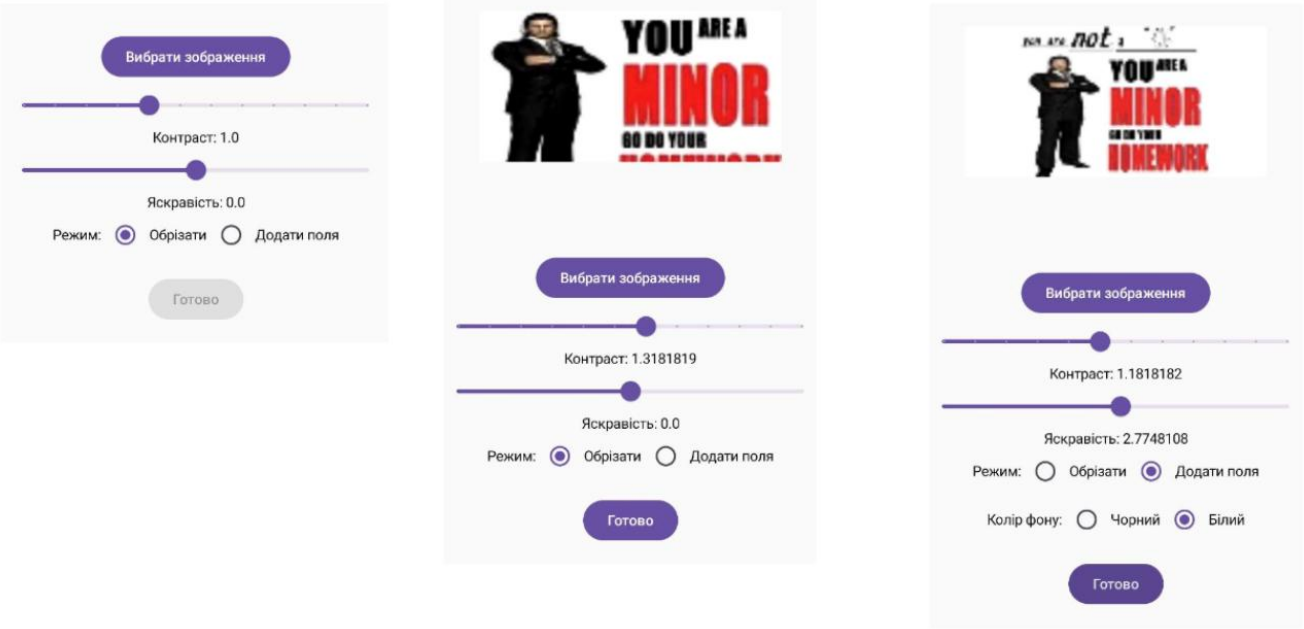


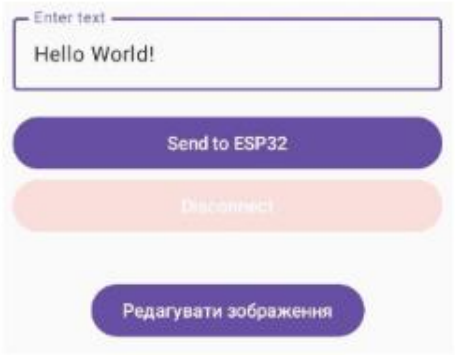
Рисунок 3.17 – Екран редагування зображення

Після завершення редагування користувач натискає кнопку «Готово». При цьому викликається метод onBitmapProcessed, який передає processedBitmap назад у головну активність. Для передачі ми використовуємо стандартний механізм setResult, у який вкладаємо зображення у вигляді байтового масиву, отриманого через функцію to1BitArray. Ця функція розширення виконує перетворення кожного пікселя у біт (чорний або білий), і групує по 8 пікселів у байт, формуючи остаточний масив із 1024 байтів, який відповідає роздільності екрану ESP32.

3.6. Тестування системи

Тепер наша система повністю готова, оскільки ми завершили розробку фізичної та програмної частини додатку. А це означає, що тепер можна виконати коротке тестування на її працездатність.

Спершу виконаємо підключення та відправимо текстове повідомлення на екран. Для цього виконаємо такі кроки : підключимо ESP32 до джерела живлення > в мобільному додатку знайдемо пристрій та виконаємо підключення > введемо текст в відповідне поле та відправимо. Результати зображено на рисунку 3.18



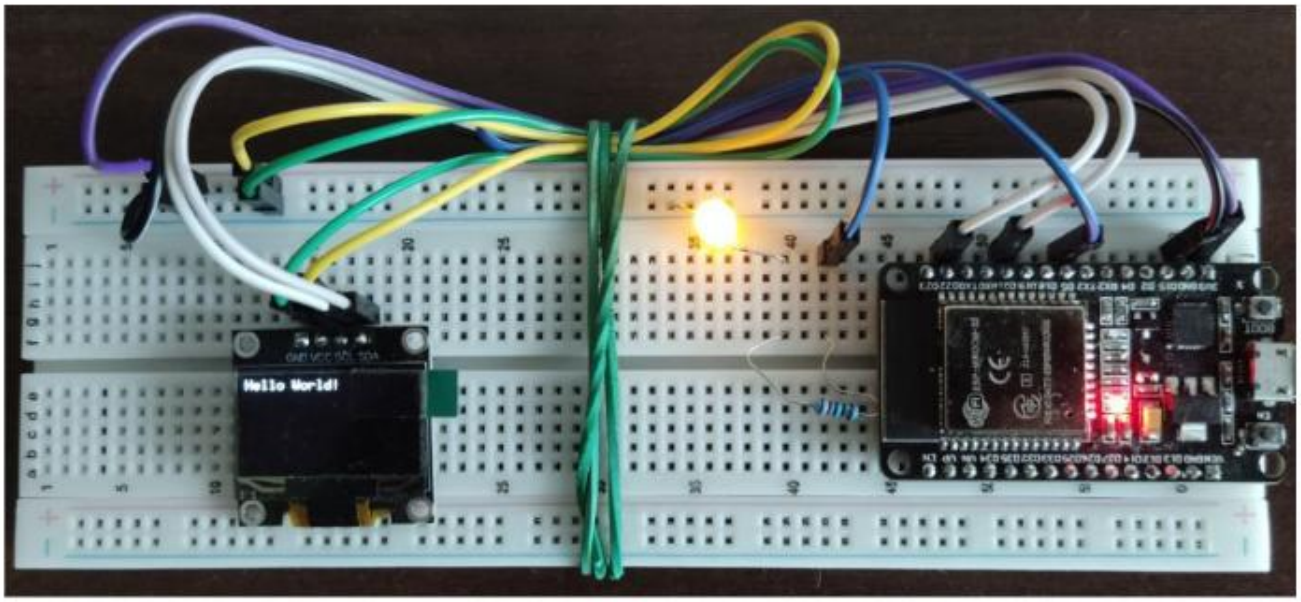


Рисунок 3.18 – Результат тестування системи на відправку повідомлень

Також перевіримо справність надсилання зображень. Для цього робимо такі кроки : виконуємо всі підключення > переходимо на екран редагування зображення > обираємо потрібне зображення, редагуємо, додаємо відступи > відправляємо на дисплей. Після виконання усіх кроків відправлене зображення повинне бути отримане мікроконтролером та виведене на екран.

На рисунку 3.19 зображено результат роботи системи, з якого можна дійти висновку, що проєктування виконано успішно та система працює справно на усіх рівнях реалізації.

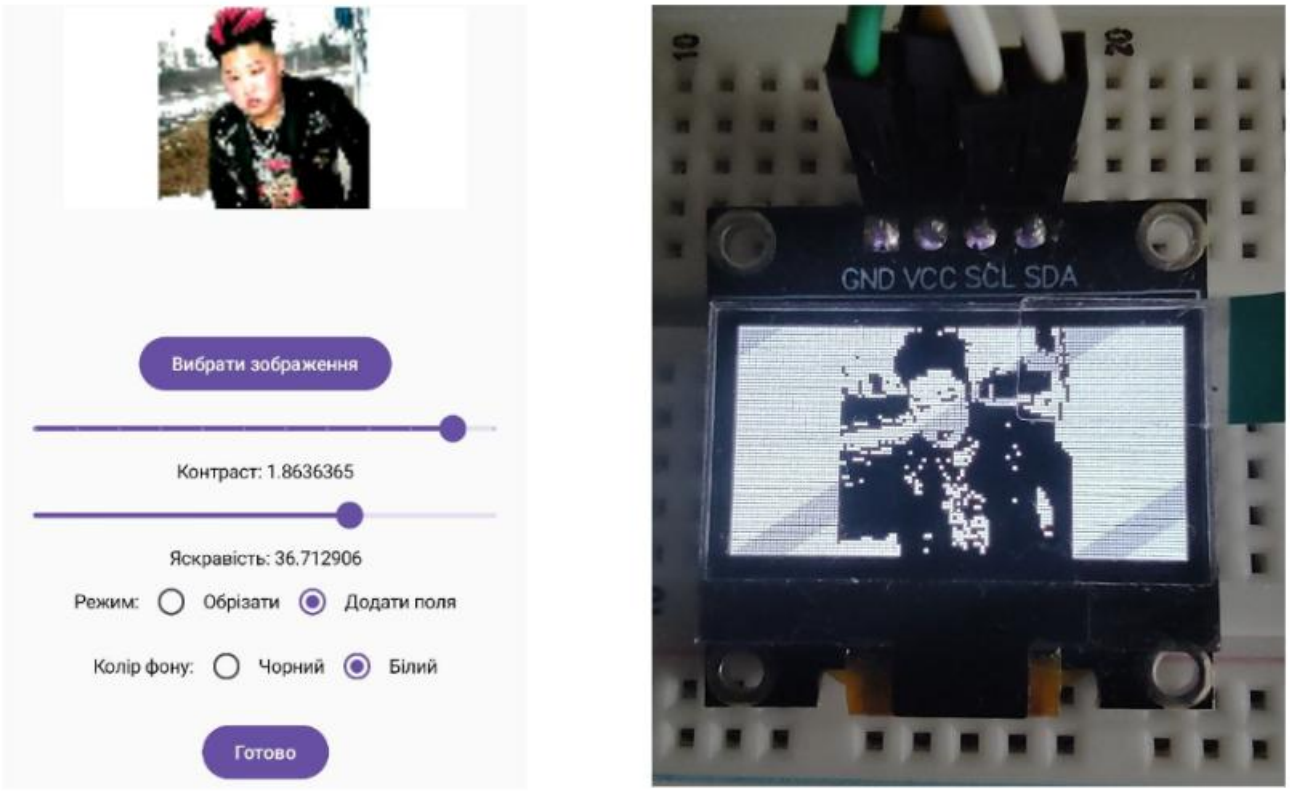


Рисунок 3.19 – Результати роботи системи з зображенням

3.7 Висновки до розділу реалізації

Отже, в розділі реалізації системи ми виконали збирання фізичної частини, зробивши необхідні з’єднання згідно попереднього проєктування з розділу 2.

Також ми справно прошили мікроконтролер ESP32 за допомогою середовища розробки Arduino IDE, що дозволило нам перейти до реалізації мобільного додатка.

Мобільний додаток був можливо найоб’ємнішою частиною роботи. Ми прописали логіку роботи програми, намалювали усі потрібні екрани, та впевнилися в тому, що програма працюватиме на всіх підтримуваних версіях ОС Android.

В кінці ми провели тестування виконаної роботи в результаті якого впевнилися, що у всіх заданих сценаріях система відпрацьовує справно та виконує завдання проєкту.

ВИСНОВКИ

У цьому диплому проєкті було розроблено кіберфізичну систему, що включала в себе фізичний пристрій з мікроконтролером ESP32 та OLED-екарном, а також мобільний додаток під операційну систему Android. З результатами розробки, мобільний додаток може виконувати підключення до пристрою з дисплеєм, та передавати текстові повідомлення та зображення.

Розглядаючи проект поетапно, можемо сказати що в першому розділі було проведене теоретичне дослідження робочої області, була розглянута історія її розвитку, а також було проаналізовано вже наявні доступні рішення. Було проаналізованої їх переваги та недоліки, що допомогло клаще зрозуміти бажаний фінальний результат дипломного проєкту.

В другому розділі ми зпроєктували систему, окремо її компоненти а також логіку взаємодії цих компонентів. Була виконана принципова схема, схема підключень, а також UseCase-діаграма. Також було проаналізовано склад програмного забезпечення та інших засобів проєктування. За результатами розділу ми чітко зрозуміли і на теоретичному рівно зпроєктували реалізацію системи.

В третьому розділі ми виконали практичну реалізацію зпроєктованої кіберфізичної системи. Проведений раніше аналіз допоміг чітко зрозуміти які компоненти слід використовувати при реалізації проєкту, а завдяки розробленим схемам вдалося правильно виконати усі підключення. Після завершення збирання фізичної частини системи, був також розроблений мобільний додаток, а процес його створення і вся логіка були описані в 3 розділі. В кінці розділу ми провели тестування системи та впевнилися у її працездатності, а також в тому що вона задовольняє початкові вимоги згідно теми проєкту.

Загалом цей дипломний проєкт допоміг краще розібратися в предметній області та покращив навички, що використовувалися в роботі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Коваль А. О. Bluetooth-технології в системах IoT. *Вісник КНУ. Серія: Фізико-математичні науки*. 2021. № 3.С. 50–55.
2. Kim, S. H. Bluetooth Low Energy: The Developer's Handbook. Boston: Newnes, 2015. 258 p.
3. Karvonen, J. Bluetooth Low Energy: Understanding BLE Protocols for Internet of Things. Helsinki: Metropolia Publishing, 2018. 180 p.
4. Плотніков І. С. Мікроконтролер ESP32: архітектура, програмування та застосування. Харків: ФОП Ранок, 2022. 212 с.
5. Adafruit. Adafruit SSD1306 OLED Display Library. Documentation. URL: https://github.com/adafruit/Adafruit_SSD1306 (дата звернення 10.06.25)
6. Adafruit. Adafruit GFX Graphics Library. Documentation. URL: https://github.com/adafruit/Adafruit-GFX-Library (дата звернення 10.06.25)
7. Valvano, J. Introduction to Embedded Systems: Interfacing to the Real World with the MSP432. 2nd ed. CreateSpace, 2017. 606 p.
8. Задорожний О. В. Основи Android-розробки мовою Kotlin: навч. посіб. Київ: Видавничий дім «Професіонал», 2021. 192 с.
9. Nagy, B. Mastering Android Development with Kotlin. Packt Publishing, 2019. 368 p.
10. JetBrains. Kotlin Language Documentation. URL: https://kotlinlang.org/docs/home.html (дата звернення 10.06.25)
11. Google. Android Jetpack. Android Developers. URL: https://developer.android.com/jetpack (дата звернення 10.06.25)
12. Колісник С. М. Обробка зображень в Android: бібліотеки та підходи. *Вісник КНУ. Серія: Прикладна математика.* 2021. №17.С. 89–95.
13. González, R. Digital Image Processing. 4th ed. Pearson, 2018. 1062 p.
14. Білоус О. А. Засоби бездротової передачі даних: Bluetooth, Wi-Fi, ZigBee. Львів: ЛНУ, 2020. 146 с.

15. Сидоренко Д. І. Протокол BLE для IoT-пристроїв. *Сучасні інформаційні технології.* 2022. №2.С. 44–49.

16. Назаренко І. П. Побудова IoT-пристроїв на базі ESP32. *Вісник НТУУ "КПІ".* 2021. №3.С. 112–117.

17. Ruiz, M. Internet of Things Projects with ESP32. Packt Publishing, 2019. 402 p.

18. Miro Samek. Practical UML Statecharts in C/C++: Event-Driven Programming for Embedded Systems. 2nd ed. Newnes, 2008. 660 p.

19. Голубєв А. І. Основи мікроелектроніки: навчальний посібник. Одеса: ОНПУ, 2020. 320 с.

20. Мельниченко О. В. Основи побудови вбудованих систем на базі мікроконтролерів. Черкаси: ЧДТУ, 2019. 210 с.

21. Singh, A. Hands-On Image Processing with Python. Packt Publishing, 2018. 454 p.

22. OpenCV. Open Source Computer Vision Library Documentation. URL: https://docs.opencv.org/ (дата звернення 10.06.25)

23. Hecht, J. Understanding OLED displays. IEEE Spectrum. 2020. №2.С. 22–26.

24. Janin, M. OLED Technology and Applications. Springer, 2017. 264 p.

25. Ковальчук С. П. Вбудовані системи реального часу на ESP32. *Системи управління та навігації.* 2022. №1.С. 88–93.

26. Бойко І. В. Методи підвищення контрастності зображень. *Радіоелектроніка і телекомунікації.* 2020. №2.С. 55–59.

27. Sharan, D. Android UI Development with Jetpack Compose. O'Reilly Media, 2022. 312 p.

28. Google Developers. Jetpack Compose Overview. URL: https://developer.android.com/jetpack/compose (дата звернення 10.06.25)

29. Chet Haase. Android Graphics: A Journey Through the Android Graphics Pipeline. URL: https://graphics.studio.android.com/ (дата звернення 10.06.25)

30. Ульянов А. С. Основи розробки UI в Android Studio. Дніпро: ДНУ, 2021. 138 с.

31. Bluetooth SIG. Bluetooth Core Specification v5.4. URL: https://www.bluetooth.com/specifications/bluetooth-core-specification (дата звернення 10.06.25)

32. Faragher, R. Understanding BLE Channel Selection Algorithms. *IEEE Communications Magazine.* 2021. Vol. 59, № 9.P. 56–62.

33. Potnis, K. Bluetooth Low Energy Security Essentials. Birmingham: Packt Publishing, 2020. 242 p.

34. Espressif Systems. ESP-IDF Programming Guide (v5.1) URL: https://docs.espressif.com/projects/esp-idf (дата звернення 10.06.25)

35. Salmoria, P. Getting Started with ESP-IDF. Cambridge: Elektor, 2022. 176 p.

36. Davidson, S. OLED Display Fundamentals and Applications. Boca Raton: CRC Press, 2019. 308 p.

37. Кравчук М. І. Розроблення графічних інтерфейсів Jetpack Compose. *Наукові записки ХНУ*. 2023. № 4.С. 101–107.

38. Google. Compose Runtime and Coroutines. URL: https://developer.android.com/jetpack/compose/coroutines (дата звернення 10.06.25)

39. Nazar, P. Kotlin Coroutines: Deep Dive. O'Reilly Media, 2021. 294 p.

40. Guennemann, S. Practical BLE Application Development. *ACM Transactions on Embedded Computing Systems.* 2022. Vol. 21, № 2.P. 1–26.

41. Khanna, P. Power Consumption Analysis of BLE Devices. *IEEE Sensors Journal.* 2020. Vol. 20, № 15.P. 8872–8878.

42. Курилко О. В. Wokwi як онлайн-середовище для проєктування IoT-пристроїв. *Вісник ЧНТУ*. 2022. № 1.С. 66–70.

43. draw.io. User Manual (v23.1). URL: https://www.drawio.com/manual (дата звернення: 10.06.25)

44. Hennig, M. Clean Architecture for Android Developers. Leanpub, 2020. 185 p.

45. Dietrich, C. BLE Mesh Networking: An Overview. *IEEE Internet of Things Magazine.* 2022. Vol. 5, № 3.P. 42–47.

46. Brownlee, J. Deep Learning for Image Processing Projects. Machine Learning Mastery, 2019. 304 p.

47. ISO/IEC 30141:2022. Internet of Things (IoT). Reference Architecture. Geneva: ISO, 2022. 64 p.

48. Kivy, D. Practical Android Bluetooth Classic and BLE. Apress, 2023. 146 p.

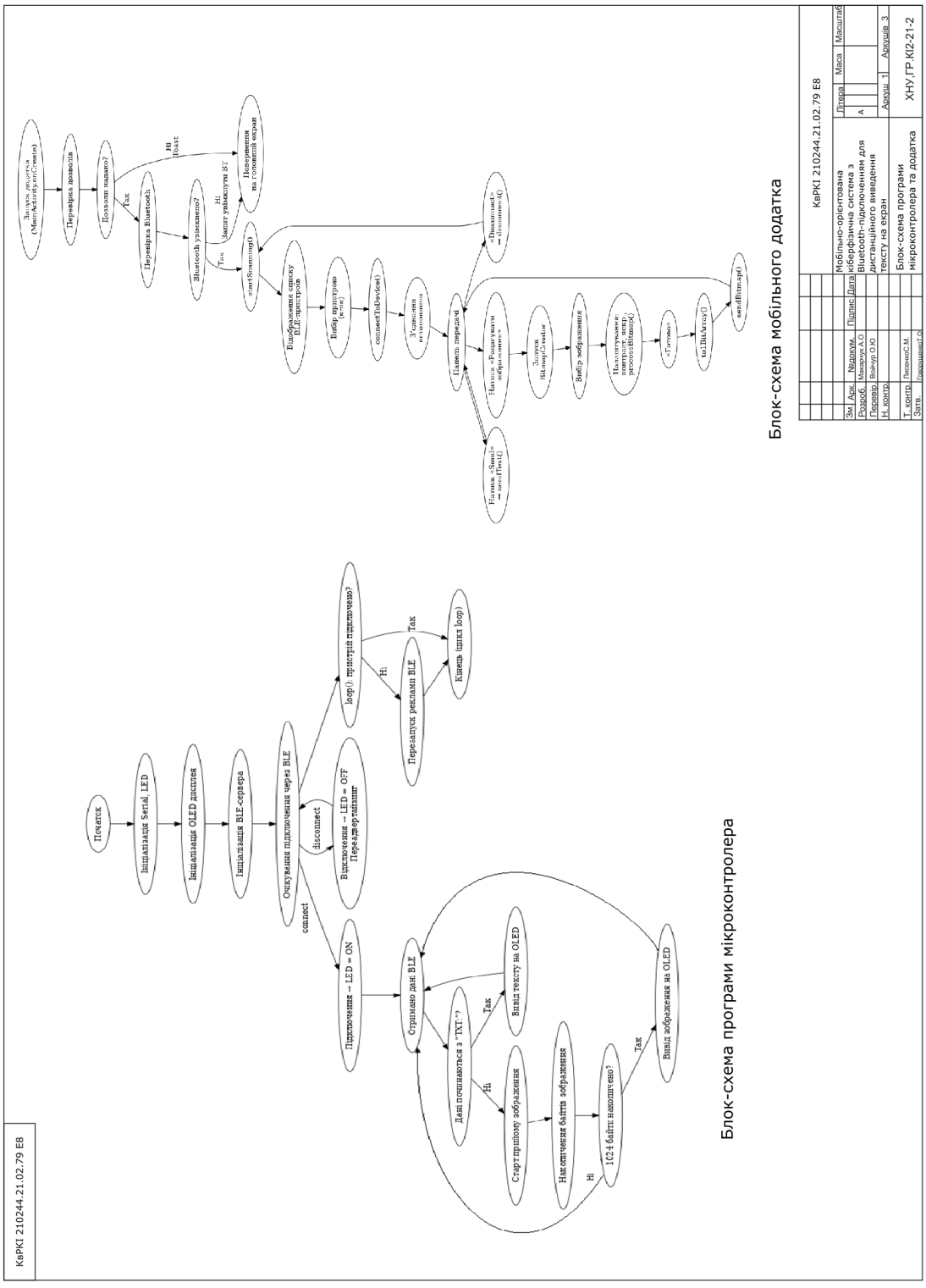
49. Пастух І. Л. Мікроконтролери в інженерії: сучасні підходи до проєктування IoT-пристроїв. Тернопіль: ТНТУ, 2022. 188 с.

50. IEEE Std 11073-20601-2022. Health informatics - Personal health device communication - Application profile optimized for fitness sensors. New York: IEEE, 2022. 192 p.

Додаток А

(обов’язковий)

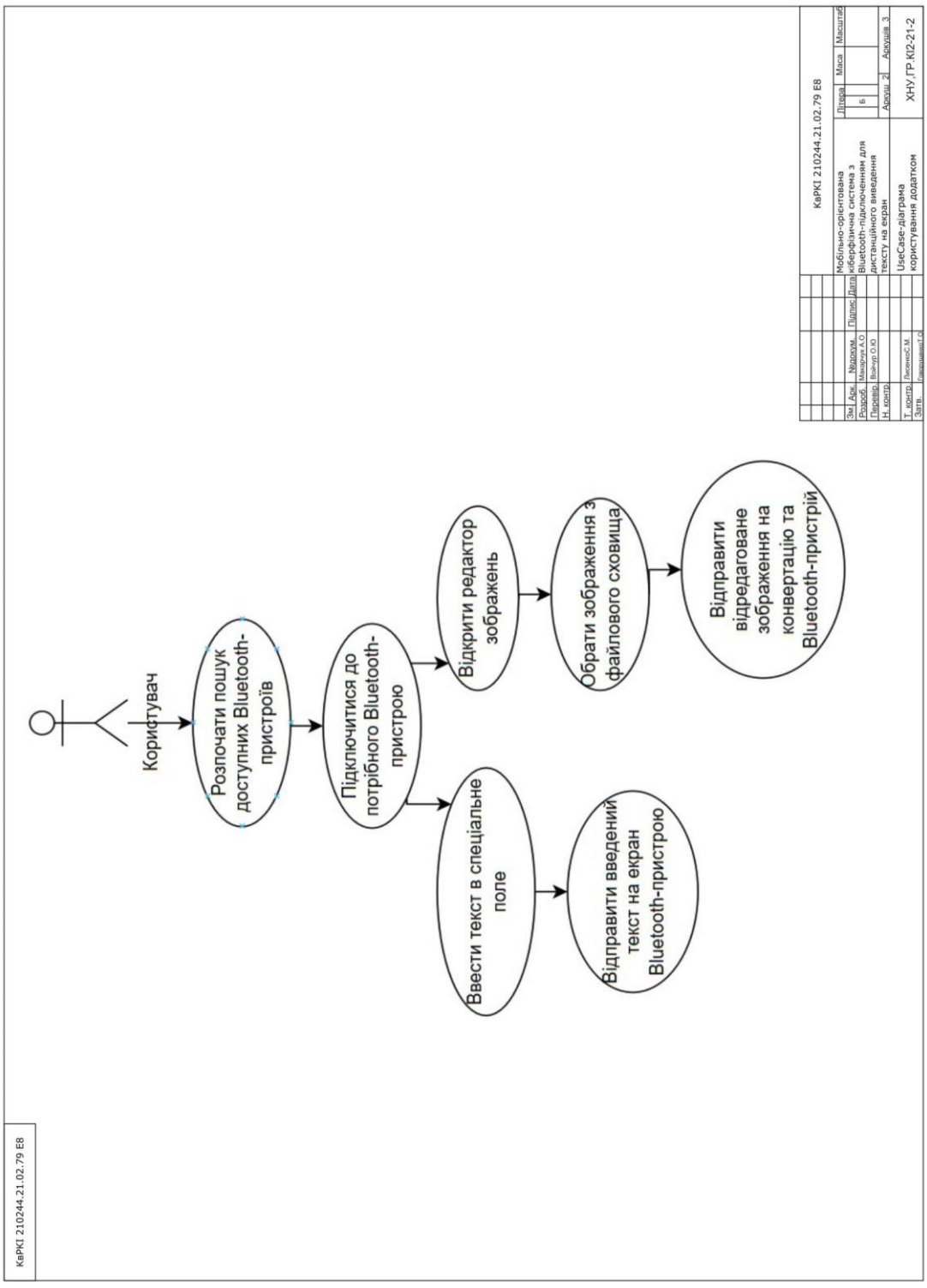
КОПІЯ КРЕСЛЕННЯ «ЛОГІЧНА СХЕМА ПРОГРАМИ МІКРОКОНТРОЛЕРА ТА ЗАСТОСУНКА»



Додаток Б

(обов’язковий)

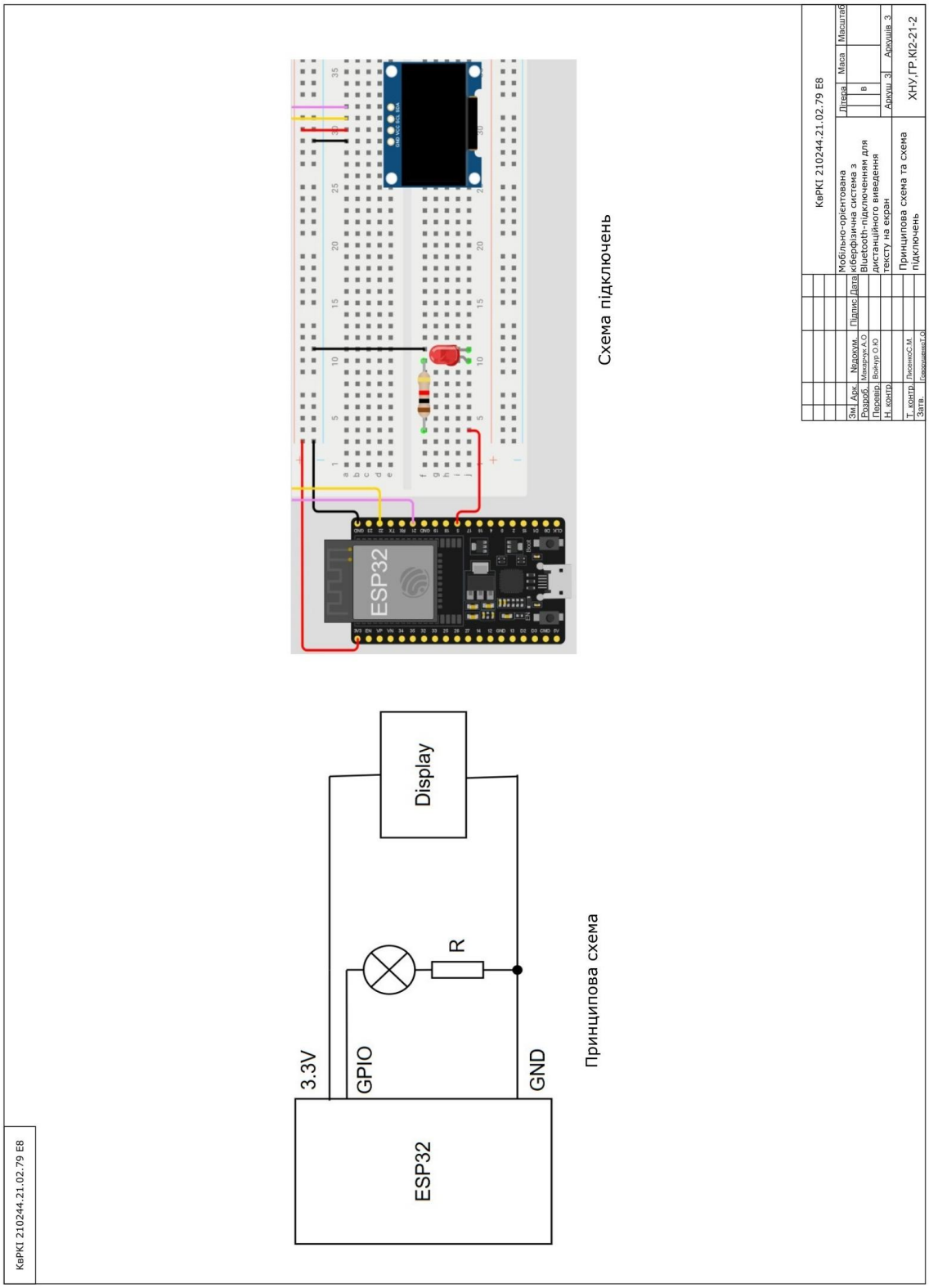
КОПІЯ КРЕСЛЕННЯ «USECASE-ДІАГРАМА КОРИСТУВАННЯ ЗАСТОСУНКОМ»



Додаток В

(обов’язковий)

КОПІЯ КРЕСЛЕННЯ «АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ ПРОЄКТУ»



Протокол аналізу звіту подібності експертом

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Андрій МАКАРЧУК

Співавтор:

Назва: Макарчук_Мобільно-орієнтована кіберфізична система з Bluetooth-підключенням для дистанційного виведення тексту на екран

Експерт:

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1:1.3%

Коефіцієнт подібності 2:0.5%

Мікропробіли: 7

Заміна букв: 1

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2025-06-10 23:28:50.0

Після аналізу Звіту подібності констатую наступне:

☒ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

2025-06-11 Доцент Андрій Нічепорук

Дата експерт

Anti-Plagiarism (UA) v-15.281 Educational

The maximum coincidence with one document 23.0%

Dictionaries check: en_US, ru_RU, ua_UA. **Errors in the documents: 14%**

	Document		Sum coincidence on the DB	
ID: 244754 Title: БКР Мобільно-орієнтована кіберфізична система з Bluetooth-підключенням для дистанційного виведення тексту на екран Added in a DB: 2025-06-10 Authors: Андрій МАКАРЧУК Heads: Олег ВОЙЧУР Consultants: Opponents:	Symbols	Lexemes	Symbols	Lexemes
	76911	669	18410 (24%)	144 (22%)

Plagiarism sources

ID	Description	Plagiarism presence in the document	
		Symbols	Lexemes
240792	Title: Звіт з ПДП Мобільно-орієнтована кіберфізична система з Bluetooth-підключенням для дистанційного виведення тексту на екран Added in a DB: 2025-05-03 Authors: Макарчук А. О. Heads: Войчур О.Ю. Consultants: Opponents:	18024 (23.0%)	138 (21.0%)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Макарчук Андрій Олександрович

Тема: Мобільно-орієнтована кіберфізична система з Bluetooth-підключенням для дистанційного виведення тексту на екран

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень ___3___ Кількість сторінок записки ___61___

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є проєктування та реалізація мобільно-орієнтованої кіберсистеми з Bluetooth-підключенням для дистанційного виводу тексту на екран.

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: В першому розділі кваліфікаційної роботи проведено дослідження предметної області (проаналізовано предметну область дослідження, виконано пошук готових рішень що відповідали б вимогам завдання, а також огляд їх переваг та недоліків) та виконано постановку задачі. В другому розділі кваліфікаційної роботи були обрані компоненти системи та інші засоби реалізації проєкту. Також було виконано повне проєктування, була розроблена принципова схема та схема підключень; були обрані засоби для програмування мікроконтролера та мобільного додатку, включаючи мову програмування, підхід до розробки та основні бібліотеки також на основі UseCase-діаграми були поставлені основні вимоги до функціоналу мобільного додатка. В третьому розділі кваліфікаційної роботи виконано апаратну та програмну реалізацію кіберфізичної системи; зібрано фізичний пристрій на базі мікроконтролера ESP32 та OLED-екрана, програмування пристрою; налаштування Bluetooth-системи; розроблено мобільний додаток для взаємодії з фізичною частиною системи..

4. Позитивні сторони роботи: висока практична цінність роботи.

5. Негативні сторони роботи: поверхова теоретична частина..

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на належному науково-технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: відмінно

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) _____

доцент кафедри ЗІІЗ Яшина О.М.

"12" червня 2025 р. _____ (підпис)

Завідувачу кафедри КІІС
д-р. філософії, доц. Ользі ПАВЛОВІЙ

Андрія МАКАРЧУКА
ПІБ здобувача вищої освіти

ФІТ, 4 курсу, групи КІ2-21-2

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Strike-Plagiarism та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

09.06 2025 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ
КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Мобільно-орієнтована кіберфізична система з Bluetooth-підключенням для дистанційного виведення тексту на екран

Автор: Андрій МАКАРЧУК

Спеціальність: 123– Комп'ютерна інженерія

Освітня програма: освітньо-професійна

Науковий керівник: Олег ВОЙЧУР, асистент

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;
2) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
3) окремі виявлені збіги є загальновживаними фразами або виразами, і не є запозиченими з конкретної авторської роботи

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості StrikePlagiarism, складає 1.32% і адресується до 16 першоджерел; та системою Anti-Plagiarism складає 23%, що, з урахуванням наведених обгрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ Олег ВОЙЧУР

Гарант ОП _____ Андрій НІЧЕПОРУК

Завідувач кафедри КІІС _____ Ольга ПАВЛОВА