Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерної інженерії та управління_____
(повна назва)
Кафедра _____Автоматизації проектування обчислювальної техніки_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти ____перший (бакалаврський)
(рівень вищої освіти)

Розумний годинник на базі Arduino

(тема)

Виконав:

здобувач IV року навчання,

групи КІУКІ-21-7

_____Чайка В.Є._____
(прізвище, ініціали)

Спеціальність *123 Комп'ютерна інженерія*

_____
(код і повна назва спеціальності)

Тип програми ___освітньо-професійна____

Освітня програма _____
_____*Комп'ютерна інженерія*_____
(повна назва освітньої програми)

Керівник __проф. Хаханова Г.В._____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ _Чумаченко С.В
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерної інженерії та управління

Кафедра Автоматизації проєктування обчислювальної техніки

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 Комп'ютерна інженерія
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Комп'ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри
(підпис)

«06»_____ 2025 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Чайці Владиславі Євгеніївні
(прізвище, ім'я, по батькові)

1. Тема роботи Розумний годинник на базі Arduino

затверджена наказом по університету від 21 травня 2025 р. № 403 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 10.06.2025

3. Вихідні дані до роботи (проекту)

Плата Arduino Uno

Середовище розробки Arduino IDE

Середовище розробки Android Studio

RTC модуль DS3231SN

Bluethooth модуль HC-06

4. Перелік питань, що потрібно опрацювати в роботі

Аналіз предметної галузі та постановка задачі проектування.

Розробка апаратної частини системи.

Розробка програмної частини системи.

Розробка мобільного застосунку

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) 15 слайдів

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім’я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Видача теми проекту, узгодження і затвердження теми	06.05.2025 – 09.05.2025	
2	Аналіз проблемної галузі, постановка задачі, вибір інструментальних засобів	09.05.2025 – 14.05.2025	
3	Вибір апаратної платформи	14.05.2025 – 16.05.2025	
4	Розробка функціональної схеми	16.05.2025 – 17.05.2025	
5	Розробка програмного модуля	17.05.2025 – 23.05.2025	
6	Оформлення пояснювальної записки	23.05.2025 – 25.05.2025	
7	Перевірка виконаного проекту керівником, допуск до захисту	25.05.2025 – 10.06.2025	
8	Захист проекту	12.06.2025 – 24.06.2025	

Дата видачі завдання 06 травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Хаханова Г.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи бакалавра: 53 с., 12 рис., 1 табл., 15 лістингів, 9 джерел, 2 додатка.

РОЗУМНИЙ ГОДИННИК, ARDUINO UNO, RTC, БЕЗПРОВІДНЕ З’ЄДНАННЯ, FLUTTER, BLUETOOTH, C++, DART.

Об'єктом розробки є розумний годинник, призначений для відображення часу та виконання додаткових функцій з використанням мікроконтролера.

Предметом розробки є апаратно-програмний комплекс на базі Arduino, який забезпечує функціональність розумного годинника, включаючи збір, обробку та виведення даних.

Метою розробки є створення розумного годинника на базі Arduino, який забезпечує точне відображення часу, функції будильника та таймера, можливість вимірювання навколишніх параметрів, таких як температура, а також передавання даних для подальшого використання.

ABSTRACT

Explanatory note of the bachelor's qualification work: 53 pages, 12 figures, 1 table, 15 listings, 9 sources.

SMART CLOCK, ARDUINO UNO, RTC, WIRELESS CONNECTION, FLUTTER, BLUETOOTH, C++, DART.

The object of the development is a smart clock designed to display time and perform additional functions using a microcontroller.

The subject of the development is a hardware and software system based on Arduino that provides the functionality of a smart clock, including data collection, processing, and output.

The purpose of the development is to create a smart clock based on Arduino that ensures accurate time display, includes alarm and timer functions, the ability to measure environmental parameters such as temperature, and data transmission for further use.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 7
ВСТУП 8
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ 10
1.1 Аналіз предметної області 10
1.2 Огляд вже існуючих рішень 13
1.3 Порівняльна характеристика 17
1.4 Постановка задачі 18
2 РОЗРОБКА АПАРАТНОЇ ЧАСТИНИ ПРОЄКТУ 19
2.1 Arduino 19
2.2 Arduino Uno 20
2.3 Модуль годинника реального часу DS3231SN 22
2.4 Модуль Bluetooth HC-06 23
2.5 Цифровий датчик температури DS18B20 25
2.6 LCD 1602 I2C символьний дисплей 26
2.7 Безпровідна зарядка 27
2.8 Buzzer 28
2.9 Розробка апаратної частини системи 29
3 РОЗРОБКА ПРОГРАМНОЇ ЧАСТИНИ ПРОЕКТУ 31
3.1 Arduino IDE 31
3.2 Бібліотека microDS18B20 32
3.3 Бібліотека microDS3231 33
3.4 Бібліотека LiquidCrystal_I2C 34
3.5 Розробка програмної частини системи 35
4 РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ 41
4.1 Android Studio 41
4.2 Flutter 42
4.3 Бібліотеки 44
4.4 Розробка мобільного застосунку 47
ВИСНОВКИ 52
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 53
ДОДАТОК А 54
ДОДАТОК Б 62

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

RISC (англ. Reduced Instruction Set Computer) – комп’ютер із скороченим набором інструкцій.

ICSP (англ. In-Circuit Serial Programming) – послідовне програмування в колі.

PWM (англ. Pulse-Width Modulation) – широтно-імпульсна модуляція.

UART (англ. Universal Asynchronous Receiver-Transmitter) – універсальний асинхронний приймач-передавач.

SPI (англ. Serial Peripheral Interface) – послідовний периферійний інтерфейс.

I2C (англ. Inter-Integrated Circuit) – інтерфейс для зв’язку між мікросхемами на платі.

Діапазон ISM (англ. Industrial, Scientific and Medical) – набір частот, призначених для промислового, наукового та медичного використання. Також застосовується в бездротових технологіях, таких як Wi-Fi та Bluetooth.

FHSS (англ. Frequency-Hopping Spread Spectrum) – частотний розширений спектр.

RTC (англ. Real-Time Clock) – годинник реального часу. Незалежна мікросхема, яка відстежує поточну дату й час, навіть коли основне живлення вимкнене.

SRAM (англ. Static Random-Access Memory) – статична оперативна пам’ять.

ВСТУП

В епоху швидкого технологічного розвитку вбудовані системи та рішення на основі мікроконтролерів стали важливими для розробки інтелектуальних електронних пристроїв. Зростаючий попит на автоматизацію та точність призвів до широкої інтеграції мікроконтролерів у різні споживчі та промислові програми.

Годинник – це пристрій, що допомагає людині орієнтуватися у часі, тому не дивно, що з плином розвитку технологій, він також зазнав суттєвих змін.

Перші спроби вимірювання часу з’явилися ще в Давньому Єгипті, де використовували сонячні годинники. Це були дуже прості пристрої – паличка, вкопана в землю, від якої тінь змінювала своє положення в залежності від часу доби. Сонячні годинники служили не тільки для практичних потреб, але й мали культурне значення в різних цивілізаціях.

У середньовічній Європі з'явилися перші механічні годинники, які працювали за допомогою механізмів із колесами та гирями. Вони ставилися на дзвіниці церков, міських ратуш та на важливих громадських будівлях. Адже були великими, громіздкими та потребували постійного заправлення, однак вони стали важливими для соціального життя того часу, бо допомагали синхронізувати час для всіх громадян.

Прорив у годинниковій техніці відбувся в XVI столітті, коли були винайдені пружинні годинники. Вони дозволяли значно зменшити розміри, зробивши їх більш мобільними. Перші такі годинники були настільними, але з часом з’явилися портативні версії, які могли носити з собою заможні люди. Підвищення точності та зручності механізмів дозволило людям ще більше покладатися на цей інструмент у повсякденному житті.

Наприкінці XX століття на зміну механічним годинникам прийшли кварцові годинники, які використовували електричні імпульси для

регулювання часу. Кварцові годинники відрізняються високою точністю, низьким рівнем обслуговування та довгим терміном роботи від батареї. Ці годинники стали дуже популярними завдяки їхній доступності та надійності, і до сьогодні вони займають провідне місце на ринку.

Традиційні годинники еволюціонували від простих механічних пристроїв відліку часу до складних електронних систем, здатних виконувати багато функцій, окрім простого відображення часу. Завдяки прогресу в технології мікроконтролерів та інтеграції датчиків сучасні електронні годинники можуть надати користувачам багато різноманітних функцій. Ці вдосконалення перетворили годинники на гаджети, які сприяють як зручності, так і ефективності в повсякденному житті.

Метою цього дослідження є розробка та впровадження розумного електронного годинника на базі Arduino, який об’єднує основні функції відліку часу з додатковими функціями, такими як функції таймера та будильника, бездротова зарядка телефону, вимірювання температури та взаємодія з користувачем через Bluetooth з’єднання.

Об'єкт дослідження – це технології, що використовуються для створення електронних годинників з функцією синхронізації з мобільними пристроями, а предмет дослідження – процес розробки та вдосконалення функцій годинника.

Таким чином, у результаті виконання роботи буде створено функціональний електронний годинник, який стане не тільки практичним інструментом для вимірювання часу, що забезпечить зручність використання та контроль за часом через мобільний телефон.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної області

У сучасну епоху цифрових технологій значно розширилася роль електронних пристроїв, які впливають на різні аспекти повсякденного життя. Серед численних новинок розумні електронні годинники набули широкої популярності завдяки своїй функціональності та простоті використання. Розвиток технології мікроконтролерів, зокрема завдяки таким платформам, як Arduino, дозволив розробити багатофункціональні розумні годинники, які виходять за рамки простого відліку часу. Ці пристрої містять різні функції, які підвищують зручність для користувачів, що робить їх важливою частиною сучасних цифрових екосистем.

Основною функцією розумного електронного годинника є відображення часу, яке залишається його основним призначенням. Сучасні реалізації цих пристроїв покладаються на високоточні модулі годинника реального часу (RTC) для забезпечення точного відліку часу. На відміну від традиційних механічних або кварцових годинників, електронні версії пропонують більшу гнучкість у синхронізації та налаштуванні часу. Інтеграція модулів RTC із системами на основі Arduino забезпечує автоматичну корекцію часу та підвищує надійність, що робить ці пристрої придатними як для домашнього, так і для професійного середовища.

Ще одна важлива функція сучасних розумних годинників – це можливість вимірювати температуру повітря в приміщенні. Ця функція зазвичай досягається завдяки використанню цифрових датчиків температури, таких як DHT11 або DS18B20, які надають дані про температуру в реальному часі з високою точністю. Зібрані дані можуть відображатися безпосередньо на інтерфейсі годинника, пропонуючи користувачам цінну екологічну

інформацію без додаткових пристроїв. Ця функція покращує роботу користувача, дозволяючи краще керувати кліматом у приміщенні, особливо в середовищах, де контроль температури є критичним.

Інтеграція можливостей бездротової зарядки ще більше розширює корисність розумних електронних годинників. Використовуючи технологію бездротової зарядки Qi, ці пристрої забезпечують зручне рішення для зарядки смартфонів та інших сумісних пристроїв. Це усуває потребу в кількох зарядних кабелях і зменшує безлад у робочих місцях. Впровадження модулів бездротової зарядки в рамках годинника не тільки покращує його функціональність, але й узгоджується з тенденцією підвищення сумісності пристроїв.

Важливим аспектом розумних електронних годинників є їх здатність функціонувати як система будильника та таймера. Встановивши Bluetooth-з’єднання зі смартфоном, користувачі можуть дистанційно налаштовувати будильники та таймери, підвищуючи зручність використання та гнучкість. Ця функція особливо корисна для людей, яким потрібне точне управління часом, як-от студентів, професіоналів і осіб зі структурованим розпорядком дня. Підключення на основі Bluetooth забезпечує безперебійну синхронізацію між годинником і смартфоном, дозволяючи користувачам легко встановлювати сповіщення. Крім того, можливість змінювати будильники без прямої фізичної взаємодії з годинником додає рівень зручності, особливо при інтеграції в більш широку екосистему розумного будинку.

Дизайн та інтерфейс розумних електронних годинників відіграють вирішальну роль у їх прийнятті та зручності використання. Сучасні моделі розроблені з урахуванням естетики та функціональності, поєднуючи різні стилі, кольори та матеріали, що відповідають різним умовам. Деякі пристрої оснащені сенсорними екранами, які забезпечують інтуїтивно зрозумілу взаємодію з користувачем, тоді як інші зберігають класичні цифрові дисплеї для простоти та ефективності. Вибір технології відображення, наприклад РК-

або OLED-екрани, значно впливає на читабельність і енергоспоживання. Ці конструктивні міркування гарантують, що розумні електронні годинники не лише служать функціональними пристроями, але й доповнюють естетику інтер'єру.

Одним з ключових переваг розумних електронних годинників є їх зручність і практичність. Інтегруючи кілька функцій в одному пристрої, користувачі можуть заощадити час і зменшити залежність від окремих гаджетів. Можливість контролювати час, відстежувати температуру, бездротово заряджати пристрої та встановлювати будильники через з'єднання Bluetooth спрощує повсякденні справи та підвищує загальну продуктивність. Крім того, автоматизація певних завдань, таких як розклад будильників та моніторинг температури, сприяє більш ефективному та організованому способу життя.

Незважаючи на численні переваги, розумні електронні годинники також стикаються з певними проблемами та обмеженнями. Однією з головних проблем є автономність акумулятора. Багато моделей покладаються на зовнішні джерела живлення через енергоспоживання їх різноманітних функцій. Хоча деякі пристрої містять акумулятори, потреба в частому підзаряджанні може обмежити їх практичність. Удосконалення технологій акумуляторів і методів оптимізації живлення необхідні для підвищення загальної ефективності та зручності використання цих пристроїв.

Ще одним суттєвим обмеженням є вартість розумних годинників високого класу. Розширені моделі з чудовими функціями та високоякісними компонентами, як правило, дорожчі, що може обмежити доступність для ширшої аудиторії. Включення бездротової зарядки та підключення Bluetooth ще більше збільшує витрати на виробництво, роблячи ці пристрої менш доступними для деяких користувачів. Вирішення цієї проблеми вимагає оптимізації виробничих процесів і вивчення економічно ефективних альтернатив у виборі компонентів.

Дивлячись у майбутнє, очікується, що розробка розумних електронних годинників буде відповідати декільком ключовим технологічним тенденціям. Удосконалення сенсорної технології забезпечить більш точний і надійний збір даних, розширивши такі функції, як моніторинг температури та датчик навколишнього середовища. Удосконалення технології акумуляторів відіграватиме вирішальну роль у збільшенні автономності пристрою, зменшенні залежності від зовнішніх джерел живлення. Крім того, розширена інтеграція з екосистемами розумного будинку ще більше підвищить універсальність цих пристроїв, забезпечуючи безперебійний зв'язок з іншими побутовими приладами та системами автоматизації.

Еволюція розумних електронних годинників, ймовірно, включатиме більший акцент на штучному інтелекті та машинному навчанні. Ці технології можна використовувати для оптимізації взаємодії з користувачем шляхом вивчення моделей поведінки та відповідного налаштування параметрів. Розширені параметри підключення, такі як інтеграція Wi-Fi, можуть додатково розширити можливості цих пристроїв, забезпечуючи синхронізацію в реальному часі з глобальними серверами часу та віддалену конфігурацію через хмарні платформи.

1.2 Огляд вже існуючих рішень

Smart Light Sound Machine – це багатофункціональний пристрій, який поєднує в собі функції нічника, бездротового зарядного пристрою, динаміка Bluetooth і будильника. Розроблений, щоб покращити атмосферу будь-якої кімнати, цей пристрій забезпечує настроюване освітлення, а також пропонує основні функції зручності.

Однією з найяскравіших особливостей Smart Light Sound Machine є його здатність освітлювати простір 12 різними кольорами. Користувачі можуть створити персоналізовану атмосферу, регулюючи яскравість і колір

відповідно до свого настрою чи вподобань. Ця гнучкість робить його ідеальним доповненням до спалень, офісів або зон відпочинку.

Можливість бездротової та USB зарядки ще більше підвищує практичність пристрою. Повна інтеграція цієї функції гарантує, що пристрої завжди будуть зарядженими та готовими до використання.

Крім того, годинник містить високоякісний динамік Bluetooth, що дозволяє користувачам транслювати музику безпосередньо зі своїх смартфонів або інших пристроїв з підтримкою Bluetooth. Для відпочинку, продуктивності чи розваги динамік забезпечує насичений звук.

Функція будильника гарантує, що користувачі можуть почати свій день за розкладом. Завдяки налаштуванням будильника користувачі можуть вибирати різні мелодії будильника, регулювати гучність і використовувати функцію повтору. Параметрами будильника можна керувати за допомогою програми, що додає ще один рівень зручності.

Рисунок 1.1 – Smart Light Sound Machine

Наступним конкурентом є розумний будильник LEXEGO – це багатофункціональний годинник, призначений для покращення повсякденної роботи користувача шляхом об’єднання різноманітних функцій в одному компактному пристрої. Цей продукт пропонує поєднання бездротової зарядки, високоякісного динаміка, нічника, FM-радіо та інтелектуальної функції будильника. Його елегантний і сучасний дизайн робить його придатним для будь-якої спальні або робочого місця, забезпечуючи зручність і комфорт в одному пристрої.

Однією з визначних особливостей цього будильника є можливість бездротової зарядки. Використовуючи технологію бездротової зарядки Qi, користувачі можуть легко заряджати свої смартфони без заплутаних кабелів. Просто помістивши сумісний пристрій на верхню панель, починається процес заряджання, гарантуючи, що телефон готовий до використання без будь-яких проблем.

Крім того, вбудований динамік забезпечує чудову якість звуку, дозволяючи користувачам насолоджуватися улюбленою музикою або брати участь у телефонних розмовах без використання рук. Підключивши смартфон через Bluetooth, користувачі можуть передавати високоякісне аудіо.

Вбудований нічник ще більше сприяє його багатофункціональності. Розроблений для створення затишної та розслаблюючої атмосфери, нічник можна використовувати для м’якого освітлення, що робить його ідеальним для нічного читання або як м’яке навколишнє освітлення для спокійного сну.

Крім того, наявність FM-радіо дозволяє користувачам починати свій день із улюбленої музики, новин або ток-шоу. Функція радіо легко регулюється, забезпечуючи безперебійне прослуховування.

Нарешті, функція інтелектуального будильника розроблена з урахуванням різних уподобань пробудження. Користувачі можуть встановлювати кілька будильників на різні дні тижня, вибирати власні мелодії будильника, регулювати гучність і використовувати функцію відкладення

одним дотиком. Ці функції роблять пробудження більш керованим і персоналізованим, задовольняючи індивідуальні потреби та вподобання.

Рисунок 1.2 – Розумний будильник LEXEGO

Останнім та найпростішим варіантом є розумний годинник Miaomiaoce. Він поєднує точне вимірювання температури та вологості з основними функціями годинника. Розроблений із РК-дисплеєм, цей розумний годинник надає дані про навколишнє середовище в реальному часі, зберігаючи при цьому елегантний і сучасний вигляд.

Функції термометра та гігрометра є одними з найважливіших аспектів цього пристрою. Розумний годинник Miaomiaoce точно вимірює температуру в діапазоні від 0 °C до +60 °C з точністю до 0,1 °C, забезпечуючи надійні показники для внутрішнього середовища. Крім того, функція гігрометра з однаковою точністю визначає рівень вологості від 0% до 99%. Дані оновлюються щохвилини, надаючи користувачам актуальну інформацію для підтримки оптимальних умов у приміщенні.

РК-дисплей розроблений для чіткості та ефективності, має

висококонтрастний екран, який відображає температуру, вологість, час, дату та функцію будильника. Унікальним аспектом дисплея є індикатор на основі смайлів, який візуально відображає рівень комфорту в кімнаті на основі показників температури та вологості. Ця функція пропонує інтуїтивно зрозумілий спосіб оцінити умови в приміщенні з першого погляду.

Живлення пристрою складається з двох батарей CR2032, які забезпечують до року роботи без необхідності частої заміни батареї.

Крім того, функція інтелектуального будильника покращує його використання, дозволяючи користувачам встановлювати будильники через спеціальну програму. Функція пам'яті записує дані за останні 24 години, зберігаючи мінімальні та максимальні значення температури та вологості.

Рисунок 1.3 – Розумний годинник Miaomiaoce

1.3 Порівняльна характеристика

Порівнюючи ці три продукти, до уваги береться кілька факторів. Розумний будильник LEXEGO зосереджений на традиційних функціях будильника, інтегруючи бездротову зарядку, FM-радіо та нічник. Навпаки,

розумний годинник Miaomiaoce спеціалізується на точному моніторингу навколишнього середовища, пропонуючи дані про температуру та вологість у реальному часі. З іншого боку, Smart Light Sound Machine наголошує на створенні атмосфери за допомогою освітлення, динаміка Bluetooth і можливостей розумного керування. Кожен пристрій виконує певну мету, задовольняючи різні потреби користувачів.

Таблиця 1.1 – Порівняльна характеристика оглянутих рішень

Особливість	LEXEGO	Miaomiaoce	Smart Light Sound Machine
Наявність зарядки	+	-	+
Функції нічника	+	-	+
Наявність моніторингу температури	-	+	-
Функції будильника	+	+	+
Наявність застосунку	-	+	+

1.4 Постановка задачі

Після ознайомлення із завданням та вивчення існуючих рішень, представлених на ринку, сформуємо завдання на виконання проекту та вимоги до виконаного пристрою:

- функція таймеру;
- функція будильнику;
- вбудована безпровідна зарядка;
- можливість керування через телефон;
- показ температури в приміщенні.

2 РОЗРОБКА АПАРАТНОЇ ЧАСТИНИ ПРОЄКТУ

2.1 Arduino

Мікроконтролерна платформа Arduino з відкритим кодом стала наріжним каменем сучасних вбудованих систем і прототипів електроніки. Завдяки своїй простоті, гнучкості та доступності він широко використовується в освітніх середовищах, промислових додатках і проєктах DIY.

Фундаментальною характеристикою Arduino є його апаратна архітектура, яка складається з мікроконтролера ATmega, набору вхідних і вихідних контактів, схем керування живленням і комунікаційних інтерфейсів, таких як USB і UART. До найпопулярніших плат належать Arduino Uno, Mega та Nano, кожна з яких адаптована для різних рівнів складності проектів.

Функціонально Arduino дозволяє користувачам розробляти широкий спектр додатків, від простих систем автоматизації до складних рішень Інтернету речей (IoT). Його середовище програмування, Arduino IDE, підтримує кодування у спрощеній версії C/C++, що робить його доступним як для початківців, так і для досвідчених розробників. Користувачі можуть писати ескізи та завантажувати їх на мікроконтролер через USB, забезпечуючи взаємодію в режимі реального часу з підключеними апаратними компонентами, такими як датчики, дисплеї та комунікаційні модулі. Крім того, сумісність Arduino з різними бібліотеками програмування значно прискорює розробку, надаючи попередньо написаний код для загальних функцій.

Переваги Arduino включають його доступність, простоту використання та широку підтримку спільноти. Природа апаратного та програмного забезпечення з відкритим вихідним кодом заохочує інновації та співпрацю, результатом чого є величезне сховище проектних ідей, навчальних посібників та посібників з усунення несправностей. Його здатність «plug-and-play»

дозволяє швидко створювати прототипи, не вимагаючи глибоких знань електроніки, що робить його ідеальним інструментом як для студентів, любителів, так і для професіоналів.

Однак Arduino також має обмеження. У порівнянні з більш просунутими мікроконтролерами або одноплатними комп'ютерами, такими як Raspberry Pi, він має нижчу обчислювальну потужність і пам'ять, що робить його менш придатним для складних обчислювальних завдань. Крім того, незважаючи на те, що він чудово керує апаратним забезпеченням та інтеграцією датчиків, йому не вистачає вбудованої здатності запускати операційну систему, що обмежує його використання в програмах, які вимагають виконання програмного забезпечення високого рівня. Іншим недоліком є енергоспоживання, оскільки деякі плати Arduino можуть бути не оптимізовані для енергоефективних програм із живленням від акумулятора.

Незважаючи на ці обмеження, Arduino залишається потужною та універсальною платформою, яка продовжує формувати сфери робототехніки, автоматизації та розробки IoT. Його простота використання, потужна підтримка спільноти та адаптивність забезпечують його місце як провідного інструменту для створення електронних прототипів та інновацій.

2.2 Arduino Uno

Arduino Uno – одна з найпопулярніших і широко використовуваних плат мікроконтролерів в екосистемі Arduino. Вона є основним інструментом для людей будь-якого рівня знань, пропонуючи баланс між простотою використання та універсальністю. Розроблений для розробки вбудованих систем, робототехніки та автоматизації, Arduino Uno забезпечує доступну точку входу у світ мікроконтролерів та електронного прототипування.

Arduino Uno базується на мікроконтролері ATmega328P, який працює на тактовій частоті 16 МГц. Він має 32 КБ флеш-пам'яті для зберігання програм,

2 КБ SRAM для виконання операцій і 1 КБ EEPROM для енергонезалежного зберігання даних. Плата працює при стандартній напрузі 5 В, з рекомендованим діапазоном вхідної напруги від 7 В до 12 В при живленні від зовнішнього джерела. Мікроконтролер побудований на 8-розрядній архітектурі RISC, що дозволяє ефективно виконувати команди з мінімальним енергоспоживанням.

Розміри плати становлять приблизно 68,6 мм × 53,4 мм, що робить її компактною та придатною для інтеграції в різні проекти. Він містить порт USB Type-B для зв’язку з комп’ютером, роз’єм зовнішнього живлення, роз’єм ICSP і кнопку скидання. Інтерфейс USB управляється мікроконтролером ATmega16U2, який діє як міст між основним мікроконтролером і головною системою.

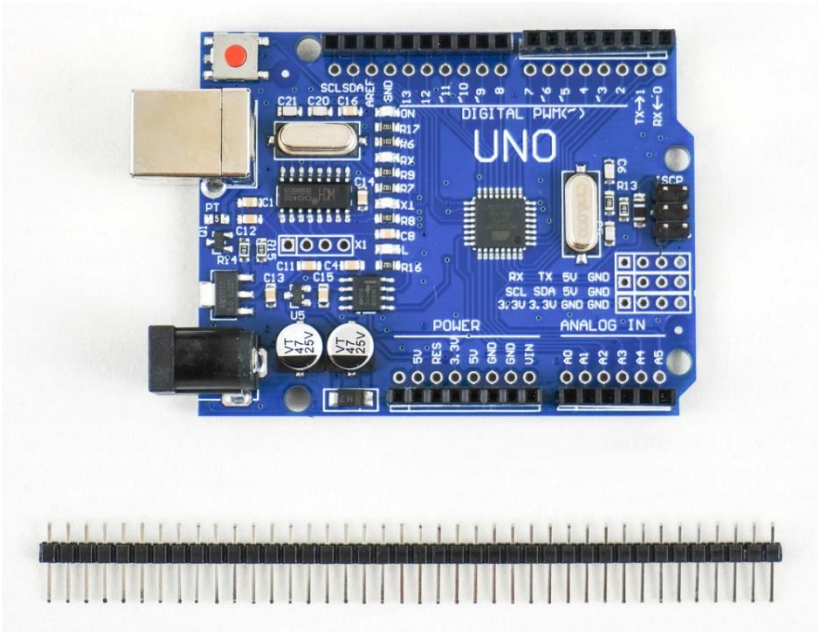

Рисунок 2.1 – Arduino Uno

Arduino Uno має 14 контактів цифрового входу/виводу, шість з яких можна використовувати як виходи PWM. Ці контакти забезпечують гнучкість у керуванні такими пристроями, як світлодіоди, двигуни та інші приводи. Крім

того, плата містить шість аналогових вхідних контактів (A0-A5), здатних зчитувати сигнали змінної напруги від датчиків та інших аналогових компонентів. Аналогові входи використовують 10-бітний АЦП, що дозволяє зчитувати значення з роздільною здатністю від 0 до 1023.

Кожен цифровий і аналоговий контакт підтримує максимальне споживання струму 40 мА, що робить його придатним для взаємодії із зовнішніми компонентами без додаткових схем. Загальний максимальний струм, що споживається на всіх контактах вводу/виводу, обмежується регуляторами напруги на платі, які забезпечують стабільну роботу.

Комунікаційні інтерфейси включають UART, SPI і I2C що забезпечує безперебійне з’єднання з іншими мікроконтролерами, датчиками, дисплеями та модулями зв’язку. Ці протоколи зв’язку забезпечують ефективний обмін даними між Arduino Uno та периферійними пристроями.

Arduino Uno залишається дуже впливовою та широко використовуваною платою мікроконтролера завдяки своїй доступності, універсальності та сильній підтримці спільноти. Він служить чудовою платформою для вивчення вбудованих систем, швидкого створення прототипів і розробки практичних програм у різних областях. Хоча він має обмеження щодо обчислювальної потужності та пам’яті, його доступність і адаптивність роблять його ідеальним вибором для освітніх цілей, проектів DIY і невеликих систем автоматизації.

2.3 Модуль годинника реального часу DS3231SN

DS3231SN – це високоточний модуль RTC із низьким енергоспоживанням, який об’єднує кристалічний генератор із температурною компенсацією і годинник/календар реального часу. Цей пристрій оптимізовано для додатків, які потребують точного відліку часу у вбудованих системах, особливо тих, що розгортаються в середовищах, де коливання температури можуть впливати на точність відліку часу.

DS3231SN працює в діапазоні від 2,3 В до 5,5 В, що робить його сумісним з різними платформами мікроконтролерів. Він має інтерфейс I2C, що забезпечує безперебійний зв’язок із мікроконтролерами при мінімальному споживанні енергії. Модуль також включає функцію резервного живлення від батареї за допомогою літієвої батареї типу «таблетка», що забезпечує безперервний відлік часу навіть за відсутності основного живлення.

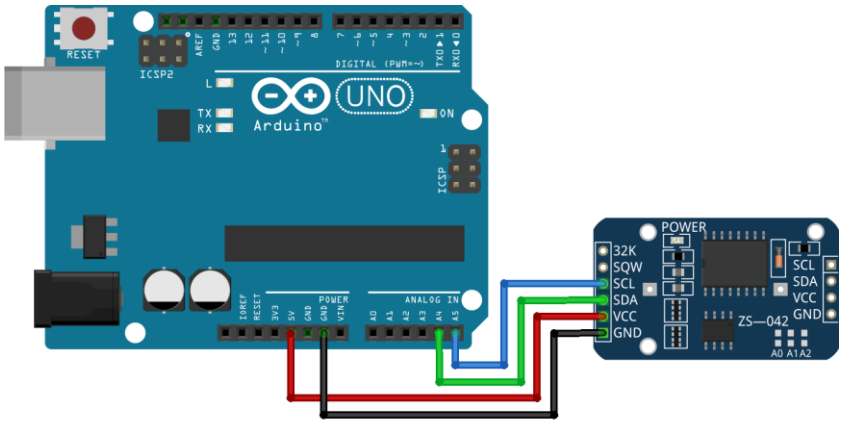

Рисунок 2.2 – Підключення DS3231SN до Arduino Uno

Крім того, DS3231SN оснащено 32 байтами енергонезалежної оперативної пам’яті, яку можна використовувати для зберігання даних користувача, які повинні зберігатися між циклами живлення. Ця додаткова пам’ять підвищує універсальність модуля в додатках, які вимагають постійного зберігання даних, таких як реєстрація часових позначок для показань датчиків або тригерів подій у системах автоматизації.

2.4 Модуль Bluetooth HC-06

HC-06 – це модуль послідовного зв'язку Bluetooth 2.0, призначений для бездротової передачі даних між мікроконтролерами та зовнішніми пристроями. Цей модуль працює в діапазоні ISM 2,4 ГГц і використовує FHSS

для пом’якшення перешкод і підвищення цілісності даних. Він використовує SPP, що забезпечує бездротовий зв’язок UART зі швидкістю передачі даних, яка налаштовується від 1200 до 115200 біт/с.

Модуль працює при напрузі живлення 3,3 В, але включає внутрішній зсув рівня, що дозволяє безпечно взаємодіяти з мікроконтролерами 5 В. Він оснащений трансивером Bluetooth 2 класу, що забезпечує діапазон зв’язку до 10 метрів у відкритому середовищі. Модуль HC-06 має вбудований світлодіодний індикатор стану, який забезпечує візуальний зворотний зв'язок щодо стану підключення.

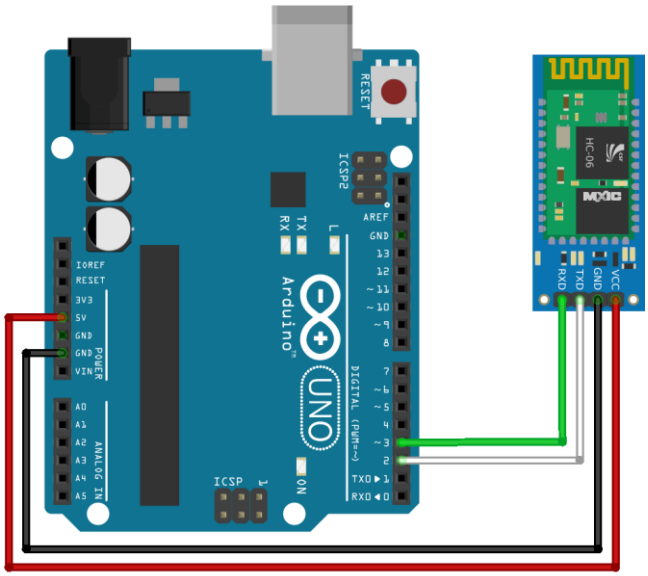

Рисунок 2.3 – Підключення HC-06 до Arduino Uno

Однією з ключових переваг модуля HC-06 є його простота інтеграції. Він підтримує AT-команди для зміни налаштувань пристрою, наприклад зміни назви пристрою, налаштування швидкості передачі даних і зміни параметрів з’єднання. Модуль зазвичай використовується в програмах, які вимагають бездротового обміну даними, включаючи промислову автоматизацію, системи IoT і рішення для дистанційного керування.

2.5 Цифровий датчик температури DS18B20

DS18B20 – це високоточний цифровий датчик температури, який використовує протокол зв’язку 1-Wire, що дозволяє підключати кілька пристроїв до одного контакту мікроконтролера. Цей датчик забезпечує зчитування температури з роздільною здатністю, яка налаштовується між 9 і 12 бітами, що дозволяє проводити вимірювання з точністю ±0,5°C у діапазоні від -10°C до +85°C.

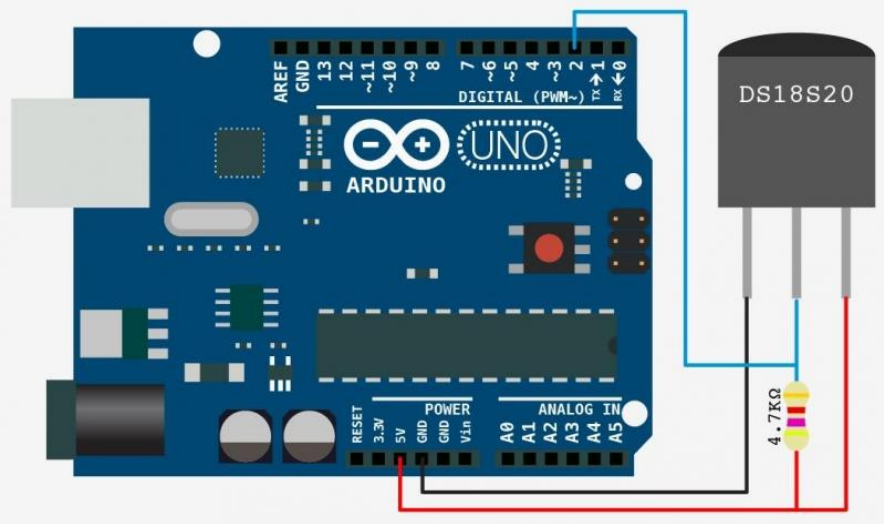

Рисунок 2.4 – Підключення DS18B20 до Arduino Uno

Працюючи при напрузі від 3,0 В до 5,5 В, DS18B20 сумісний з більшістю мікроконтролерів. Датчик видає дані про температуру в цифровій формі, усуваючи потребу у зовнішніх компонентах формування сигналу, таких як операційні підсилювачі або аналого-цифрові перетворювачі. Його паразитний режим живлення дозволяє працювати лише з двома проводами – для передачі даних і заземлення – що робить його ідеальним для додатків із низьким енергоспоживанням і обмеженим простором.

2.6 LCD 1602 I2C символьний дисплей

LCD 1602 – це 16-символьний 2-рядковий рідкокристалічний дисплей, який використовує інтерфейс I2C для ефективного зв'язку з мікроконтролерами. Цей дисплей базується на контролері HD44780, який надає широкі можливості відображення тексту, включаючи генерацію нестандартних символів, керування курсором і функції вирівнювання тексту.

Інтерфейс I2C зменшує кількість необхідних контактів GPIO з 6-8 (паралельний режим) до лише двох (SDA та SCL), значно спрощуючи підключення та інтеграцію мікроконтролера. Дисплей працює від напруги 5 В і має блакитне підсвічування з білими символами, що забезпечує відмінну читаність як в яскравому, так і в тьмяному середовищі. Він підтримує регулювання контрастності за допомогою потенціометра, що дозволяє налаштовувати на основі умов навколишнього освітлення.

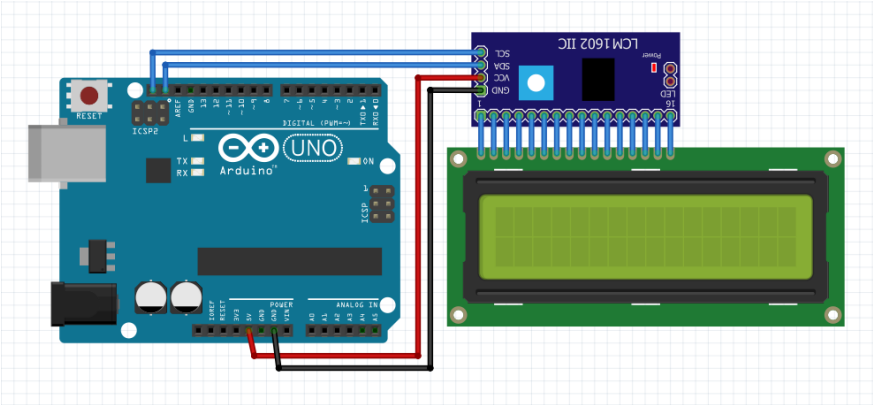

Рисунок 2.5 – Підключення LCD 1602 I2C до Arduino Uno

РК-дисплей 1602 оснащений буфером внутрішньої пам'яті, який зберігає відображуваний текст навіть за відсутності живлення, забезпечуючи стабільність у програмах відображення в реальному часі. Він підтримує функції прокручування тексту, миготливі курсори та налаштування символів,

що дозволяє розробникам створювати інтуїтивно зрозумілі інтерфейси користувача. Частота оновлення дисплея забезпечує роботу без мерехтіння, що робить його придатним для безперервної візуалізації даних у системах моніторингу та автоматизації.

2.7 Безпровідна зарядка

Безпровідна передача електроенергії є однією з перспективних технологій сучасної електроніки, яка дозволяє здійснювати зарядку пристроїв без фізичного електричного контакту між джерелом живлення та споживачем. Основним принципом безпровідної зарядки є використання явища електромагнітної індукції, при якому енергія передається від передавальної котушки до приймальної шляхом створення змінного магнітного поля.

У проєктах на основі мікроконтролерів Arduino безпровідна зарядка реалізується переважно за допомогою індуктивних модулів, таких як стандартні Qi-сумісні модулі або простіші рішення на основі пар індуктивних котушок та драйверів типу TP4056, XL6009 або BQ51050B. Arduino може виступати як контролер для моніторингу параметрів процесу зарядки, таких як напруга, струм, температура приймального елемента або стан акумулятора.

У межах реалізації проєкту було застосовано спрощений метод організації безпровідної зарядки. Для цього використано модуль стандарту MagSafe, розроблений компанією Apple, який функціонує за принципом індуктивної безпровідної передачі енергії.

Хоча Arduino Uno не має вбудованих функцій генерації високочастотного змінного струму, він може слугувати як логічний елемент управління подачею живлення на передавальний модуль. У даному випадку Arduino Uno підключено до MagSafe-передавача через перехідник з роз’єму Type-C на відповідні цифрові піни.

2.8 Buzzer

Buzzer – це звуковий сигнальний пристрій, який видає гучний, привертаючий увагу звук. Він зазвичай використовується в різних сферах застосування для позначення сповіщень, попереджень або підтверджень. Buzzer можна знайти в усьому: від побутової техніки, такої як мікрохвильові печі та пральні машини, до промислового обладнання, систем пожежної сигналізації та навіть іграшок.

Buzzer буває різних типів, включаючи механічні, електромеханічні та електронні (п'єзоелектричні) версії. Звук зазвичай створюється вібрацією діафрагми або п'єзоелектричного матеріалу під час подачі електричного струму.

У багатьох системах buzzer служить простим, але ефективним способом сповіщення користувачів про подію, наприклад, про закінчення часу таймера, натискання кнопки або помилку. Завдяки компактним розмірам, низькій вартості та високій надійності, зумери широко використовуються в різних галузях промисловості.

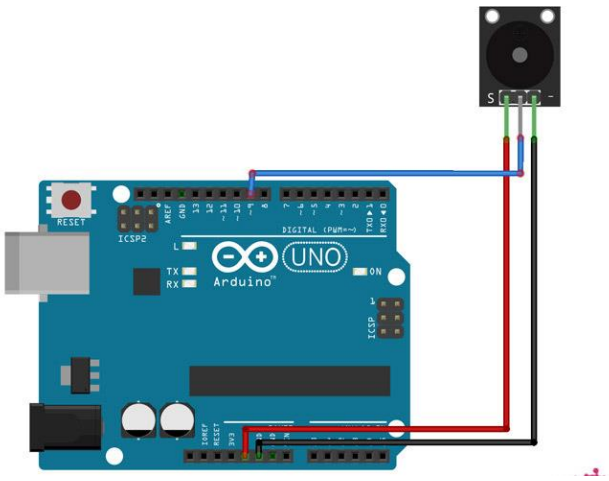

Рисунок 2.6 – Підключення Buzzer до Arduino Uno

2.9 Розробка апаратної частини системи

Апаратна частина була побудована на основі Arduino Uno із мікроконтролером ATmega328P. Також для забезпечення збору, передачі та відображення даних було підключено декілька модулів, які були описані в цьому розділі, а саме: LCD 1602 I2C символьний дисплей, цифровий датчик температури DS18B20, модуль годинника реального часу DS3231SN, модуль Bluetooth HC-06, buzzer та перехідник до MagSafe. Схему підключення компонентів до Arduino Uno зображено на рисунку 2.7.

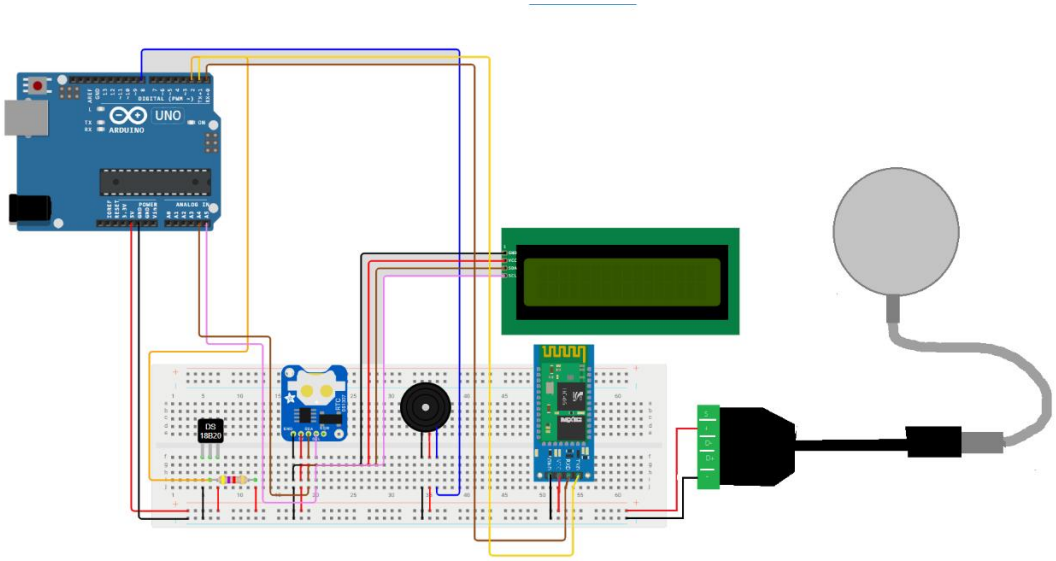

Рисунок 2.7 – Схема підключень компонентів системи

Також для зручності складання схеми з реальних компонентів було використано відладкову плату і багато проводів перемичок дюпон для прототипування.

Схема була зібрана, вигляд прототипу можна побачити на рисунку 2.8. Перевірку на справність було пройдено, адже всі елементи, які мають діоди

відображення підключення, засвітилися, а модуль Bluetooth HC-06 можна було знайти у списку пристроїв до підключення на смартфоні.

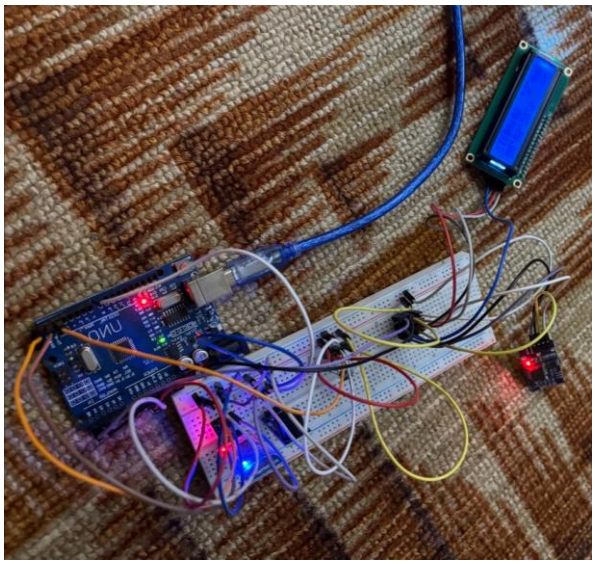

Рисунок 2.8 – Вигляд зібраної схеми

Окремо було протестована безпровідна зарядка, адже цей елемент не потребує програмної частини для функціонування. Як видно з рисунка 2.9 телефон заряджається від модуля MagSafe.

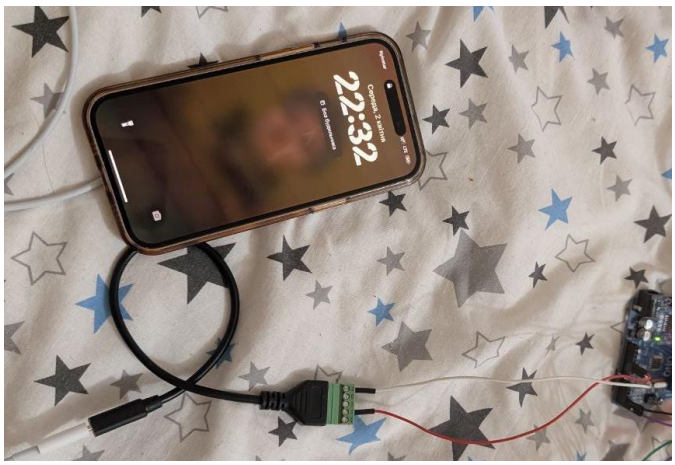

Рисунок 2.9 – Робота безпровідної зарядки

3 РОЗРОБКА ПРОГРАМНОЇ ЧАСТИНИ ПРОЕКТУ

3.1 Arduino IDE

Інтегроване середовище розробки (IDE) Arduino – це потужна крос-платформна програмна програма, призначена для написання, компіляції та завантаження коду в мікроконтролери Arduino. Він служить основним інтерфейсом для розробників, які працюють з платами Arduino, пропонуючи зручне, але багатофункціональне середовище, яке підходить як для початківців, так і для досвідчених інженерів вбудованих систем.

За своєю суттю Arduino IDE забезпечує спрощений інтерфейс програмування на основі мов програмування C і C++. Він містить вбудовані функції, що полегшує взаємодію з апаратними компонентами, такими як датчики, виконавчі механізми та комунікаційні модулі. IDE містить простий текстовий редактор із підсвічуванням синтаксису, автоматичним відступом і пропозиціями коду, що забезпечує ефективне кодування та налагодження.

Одним із ключових аспектів Arduino IDE є його сумісність із широким спектром плат, зокрема Arduino Uno, Mega, Nano та Due тощо. Він підтримує послідовний зв’язок, дозволяючи розробникам відстежувати дані в реальному часі з датчиків або налагоджувати програми за допомогою Serial Monitor. Вбудований компілятор і компонувальник перетворює зрозумілий людині код у машинно-виконувані двійкові файли, які потім можна завантажити на мікроконтролер через USB або послідовне з’єднання.

IDE спрощує процес розробки, пропонуючи інтегрований менеджер бібліотек, що дозволяє користувачам легко встановлювати зовнішні бібліотеки та керувати ними. Вони розширюють функціональність Arduino, надаючи попередньо написаний код для конкретних апаратних компонентів. Наявність численних сторонніх бібліотек ще більше підвищує гнучкість

середовища розробки.

Іншою фундаментальною функцією є диспетчер плат, який дозволяє користувачам додавати підтримку різних архітектур мікроконтролерів за межі стандартних плат Arduino. Ця функція особливо корисна для роботи з альтернативними платформами розробки, такими як ESP8266, ESP32 і мікроконтролери на базі ATmega.

Природа Arduino IDE з відкритим кодом призвела до широкої підтримки спільноти з розгалуженою онлайн-екосистемою форумів, документації та проектів з відкритим кодом. Розробники можуть скористатися цією підтримкою для вирішення проблем, оптимізації коду та відкриття нових інтеграцій обладнання.

До того ж, Arduino IDE 2.0 представляє сучасні вдосконалення, такі як покращений інтерфейс користувача, можливості налагодження в реальному часі та вбудований послідовний плотер для візуалізації даних датчиків. Ці вдосконалення задовольняють більш просунуті потреби розробки, зберігаючи при цьому простоту, яка робить Arduino доступним.

Незважаючи на свої переваги, Arduino IDE має деякі обмеження. У ньому відсутні розширені засоби налагодження, такі як точки зупину та перевірка змінних у реальному часі, які доступні в професійних середовищах розробки, таких як Atmel Studio або PlatformIO. Крім того, відсутність власного рефакторинга коду та інтеграції контролю версій може ускладнити керування великомасштабними проектами.

3.2 Бібліотека microDS18B20

Бібліотека microDS18B20 легка та ефективна, спеціально розроблена для взаємодії з датчиком температури. Цей датчик взаємодіє через протокол 1-Wire, що дозволяє кільком пристроям використовувати одну лінію передачі даних. Бібліотека вирізняється своїм компактним та ефективним дизайном, що

робить її особливо придатною для програм, що працюють на мікроконтролерах з обмеженими ресурсами.

Вона мінімізує накладні витрати, уникаючи непотрібних абстракцій та роздутої функціональності, що зустрічається у великих бібліотеках датчиків. Незважаючи на свою легкість, вона зберігає підтримку ключових функцій DS18B20, таких як асинхронні запити температури, налаштування роздільної здатності та ідентифікація окремих датчиків за їх 64-бітною адресою ROM.

Одним з недоліків microDS18B20 є те, що йому може бракувати деяких розширених функцій зручності або ширшої сумісності з пристроями, які надають більш комплексні бібліотеки, такі як бібліотеки OneWire або DallasTemperature. Варто також зазначити, що через свою орієнтацію на мінімалізм, він передбачає, що користувач має певний рівень знайомства з тим, як працює 1-Wire зв'язок і як поводяться датчики DS18B20.

3.3 Бібліотека microDS3231

MicroDS3231 – це мінімалістична бібліотека, створена для взаємодії з модулем годинника реального часу (RTC) DS3231, високоточним пристроєм відліку часу. Її основна мета – запропонувати швидкий та ефективний з точки зору пам'яті інтерфейс для доступу до даних про дату, час і температуру з DS3231, особливо в системах з обмеженою пам'яттю або обмеженими обчислювальними ресурсами.

Ця бібліотека забезпечує основні функції, такі як зчитування та встановлення поточного часу, отримання температури з внутрішнього датчика чіпа та форматування часу та дати у вигляді рядків для відображення. Завдяки своїй легкій конструкції microDS3231 особливо підходить для компактних проектів, таких як пристрої з батарейним живленням, реєстратори даних та дисплеї годинників, які не потребують складних розрахунків часових поясів або розширеної функціональності календаря.

Однак, одним з обмежень є відсутність надійної обробки помилок та розширеної функціональності порівняно з більшими бібліотеками RTC, такими як RTClib, які краще підходять для більш вимогливих програм.

3.4 Бібліотека LiquidCrystal_I2C

Бібліотека LiquidCrystal_I2C є однією з найпоширеніших бібліотек для керування символьними РК-дисплеями (такими як моделі 16x2 або 20x4) через протокол зв'язку I2C. I2C значно зменшує кількість необхідних контактів на мікроконтролері, дозволяючи зв'язок лише через дві лінії (SDA та SCL), що робить цю бібліотеку особливо популярною для проектів Arduino з обмеженою доступністю GPIO.

Ця бібліотека, по суті, є модифікованою версією стандартної бібліотеки LiquidCrystal, адаптованою для роботи з модулями I2C backpack, які використовують розширювач вводу/виводу PCF8574. Вона підтримує подібний інтерфейс програмування, що забезпечує легкий перехід для користувачів, які вже знайомі з оригінальними РК-дисплеями з паралельним керуванням. Вона підтримує стандартні операції відображення символів, включаючи друк тексту, очищення екрана, керування курсором та визначення власних символів.

Основною перевагою LiquidCrystal_I2C є зручність та простота, які вона пропонує для роботи з дисплеями з підтримкою I2C.

В інтернеті існує кілька версій цієї бібліотеки, що часто призводить до плутанини та проблем сумісності, особливо коли різні розгалужені версії використовують дещо різні API або параметри ініціалізації. У деяких випадках продуктивність також може постраждати через повільнішу швидкість зв'язку I2C порівняно з паралельним, особливо під час частого оновлення великих обсягів тексту.

3.5 Розробка програмної частини системи

У рамках програмної частини було реалізовано функції таймеру та будильнику, показ температури, можливість взаємодії з користувачем через Bluetooth для встановлення дати і часу та керування вище згаданими функціями годинника.

Була використана модульна та безпереривна архітектура з періодичним опитуванням та логікою на основі станів, що забезпечує сумісність з неблокуючими шаблонами виконання, необхідними для вбудованих систем з обмеженими ресурсами.

Скетч починається з включення необхідних бібліотек: microDS18B20 для цифрового датчика температури DS18B20, microDS3231 для модуля RTC DS3231SN та LiquidCrystal_I2C для взаємодії з LCD-дисплеєм I2C. Вони абстрагують протоколи зв'язку низького рівня та надають спрощені функції високого рівня для взаємодії з датчиками та периферійними пристроями.

Лістинг 3.1 – Підключення бібліотек

```
#include <microDS18B20.h>
#include <microDS3231.h>
#include <LiquidCrystal_I2C.h>
```

Опісля оголошуються глобальні об'єкти для представлення кожного апаратного компонента. Екземпляр MicroDS3231 керує відліком часу, тоді як шаблон MicroDS18B20<2> налаштовує датчик температури на цифровому виводі D2. Для керування логікою будильника та станами таймера оголошуються різні логічні прапорці та змінні позначок часу, тоді як цілі числа зберігають значення часу тривоги, а тривалість таймера зберігається як unsigned long у мілісекундах для точності.

Лістинг 3.2 – Константи для модулів та функцій

```
MicroDS3231 rtc;
MicroDS18B20<2> sensor;
LiquidCrystal_I2C lcd(0x27, 16, 2);

bool alarmEnabled = false;
bool alarmTriggered = false;
bool alarmStopped = false;
bool alarmPlaying = false;
unsigned long lastAlarmTime = 0;
unsigned long alarmStartTime = 0;

int alarmHour = -1;
int alarmMinute = -1;

unsigned long timerDuration = 0;
unsigned long timerStart = 0;
bool timerRunning = false;
```

Функція setup ініціалізує апаратні компоненти. Час модулю RTC встановлюється за допомогою макросу, який компілює поточну позначку часу в прошивку. Послідовний зв'язок запускається на швидкості 9600 бод для налагодження Bluetooth. РК-дисплей ініціалізується з увімкненим підсвічуванням, а контакт 8 налаштовано як вихід для зумера.

Лістинг 3.3 – Setup проєкту

```
void setup() {
 rtc.setTime(COMPILE_TIME);
 Serial.begin(9600);
 lcd.init();
 lcd.backlight();
 pinMode(8, OUTPUT);
}
```

Функція loop реалізує логіку, керовану часом, використовуючи неблокуючі методи. Програмний таймер перевіряє інтервал 800 мс для оновлення дисплея. Якщо доступне нове значення температури, значення

температури друкується на РК-дисплеї у фіксованому місці, а потім символ градуса та одиниця вимірювання Цельсія. Одночасно видається новий запит на перетворення температури для мінімізації затримки. Поточний час і дата безперервно отримуються з RTC та відображаються на РК-дисплеї.

Лістинг 3.4 – Функція loop

```
void loop() {
 static uint32_t tmr;
 if (millis() - tmr >= 800) {
  tmr = millis();

  if (sensor.readTemp()) {
   lcd.setCursor(10, 1);
   lcd.print(sensor.getTemp());
   lcd.print("C");
  }

  sensor.requestTemp();

  lcd.setCursor(0, 1);
  lcd.print(rtc.getTimeString());

  lcd.setCursor(0, 0);
  lcd.print(rtc.getDateString());
 }

 handleBluetooth();
 checkAlarm();
 checkTimer();
 stopAlarmAfterDuration();
}
```

Функція handleBluetooth спрощує розбір команд через послідовний інтерфейс. Вхідні дані зчитуються та обрізаються для видалення пробілів. Ескіз підтримує кілька команд: `

- ALARM:hh:mm встановлює будильник на вказаний час;
- ALARM_OFF негайно зупиняє зумер і скидає прапорці будильника;
- DATETIME:hh:mm:ss:dd:MM:yyyy встановлює RTC на певну

позначку часу;

– TIMER:<секунди> запускає таймер на вказану кількість секунд.

Визначення виконується за допомогою sscanf для структурованого вилучення числових параметрів, а прапорці або лічильники оновлюються відповідно для відстеження стану кожної функції.

Лістинг 3.5 – Функція handleBluetooth

```
void handleBluetooth() {
 if (Serial.available()) {
  String command = Serial.readStringUntil('\n');
  command.trim();
  if (command.startsWith("ALARM:")) {
   sscanf(command.c_str(),       "ALARM:%d:%d",       &alarmHour,
&alarmMinute);
   alarmEnabled = true;
   alarmStopped = false;
   alarmTriggered = false;
   Serial.println("Alarm set.");
  }
  else if (command == "ALARM_OFF") {
   noTone(8);
   digitalWrite(8, LOW);
   alarmStopped = true;
   alarmPlaying = false;
   Serial.println("Alarm stopped manually.");
  }
  else if (command.startsWith("DATETIME:")) {
   int h, m, s, d, mo, y;
   sscanf(command.c_str(), "DATETIME:%d:%d:%d:%d:%d:%d", &h, &m,
&s, &d, &mo, &y);
   rtc.setTime(s, m, h, d, mo, y);
   Serial.println("DateTime set.");
  }
  else if (command.startsWith("TIMER:")) {
   int sec;
   sscanf(command.c_str(), "TIMER:%d", &sec);
   timerDuration = sec * 1000UL;
   timerStart = millis();
   timerRunning = true;
   Serial.println("Timer started.");
  }}}
```

Функція будильника обробляється функцією checkAlarm, яка оцінює, чи відповідає поточний час встановленому часу будильника, і чи увімкнено будильник, а не зупинено вручну. Якщо на початку хвилини виявляється збіг (секунда дорівнює нулю), зумер активується. Записується позначка часу для полегшення подальшого контролю. Будильник також повторно спрацьовує через фіксовані інтервали, якщо він вже був запущений, але не зупинений, створюючи поведінку повторного сповіщення.

Лістинг 3.6 – Функція checkAlarm

```
void checkAlarm() {
 if (!alarmEnabled || alarmStopped) return;

 int currentH = rtc.getHours();
 int currentM = rtc.getMinutes();
 int currentS = rtc.getSeconds();

 if (!alarmTriggered && currentH == alarmHour && currentM == 
alarmMinute && currentS == 0) {
  tone(8, 1000);
  alarmPlaying = true;
  alarmStartTime = millis();
  alarmTriggered = true;
  lastAlarmTime = millis();
  Serial.println("Alarm triggered.");
 }
 if (alarmTriggered && !alarmStopped && millis() - lastAlarmTime 
> 300000 UL) {
  tone(8, 1000);
  alarmPlaying = true;
  alarmStartTime = millis();
  lastAlarmTime = millis();
  Serial.println("Alarm repeated.");
 }
}
```

Функція stopAlarmAfterDuration гарантує, що зумер не звучатиме нескінченно. Вона відстежує час, що минув з моменту запуску будильника, і вимикає зумер через п'ять секунд.

Лістинг 3.7 – Функція stopAlarmAfterDuration

```
void stopAlarmAfterDuration() {
 if (alarmPlaying && millis() - alarmStartTime >= 5000UL) {
  noTone(8);
  digitalWrite(8, LOW);
  alarmPlaying = false;
  Serial.println("Alarm stopped after 5 seconds.");
 }
}
```

Функція checkTimer керує функцією таймера зворотного відліку. Коли таймер активний і заданий час минув, він запускає короткий звуковий сигнал на дві секунди та зупиняє таймер, скидаючи прапорець керування. Ця функція гарантує, що таймер залишається точним, і надає звуковий сигнал після закінчення терміну дії.

Лістинг 3.8 – Функція checkTimer

```
void checkTimer() {
 if (timerRunning && millis() - timerStart >= timerDuration) {
  tone(8, 1500);
  delay(2000);
  noTone(8);
  timerRunning = false;
  Serial.println("Timer ended.");
 }
}
```

4 РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ

4.1 Android Studio

Android Studio – це офіційне IDE для розробки застосунків для Android, схвалене та підтримуване Google у співпраці з JetBrains. Воно побудоване на платформі IntelliJ IDEA та забезпечує надійну та масштабовану платформу для розробки програм.

З технічної точки зору, Android Studio вирізняється глибоко інтегрованим інструментарієм, який спрощує весь життєвий цикл розробки програмного забезпечення. Воно включає в себе редагування коду, компіляцію, тестування, налагодження та упаковку додатків для Android. IDE підтримує програмування на Java, Kotlin та C++.

Однією з найважливіших особливостей Android Studio є система збірки на базі Gradle. Це забезпечує детальний контроль над конфігурацією збірки та підтримує варіанти збірки, управління залежностями та безперервну інтеграцію, що забезпечує модуляризацію, масштабованість та відтворюваність у великомасштабних проектах Android.

Android Studio також включає розширений редактор макетів, який дозволяє розробникам створювати користувацькі інтерфейси за допомогою візуального інтерфейсу перетягування або за допомогою точної маніпуляції з XML. Цей редактор підтримує адаптивний дизайн завдяки ConstraintLayout, що забезпечує динамічні та адаптивні інтерфейси, що підходять для різних розмірів екрана.

З точки зору тестування, Android Studio інтегрує кілька фреймворків для тестування, включаючи JUnit для модульного тестування, Espresso для тестування інтерфейсу користувача та AndroidX Test для інструментальних тестів на реальних пристроях або емуляторах. IDE підтримує віртуальні

пристрої Android (AVD), які імітують фізичні пристрої для тестування в різних умовах, таких як різні рівні API, специфікації обладнання та географічні локалізації.

Системи контролю версій, такі як Git, Mercurial та Subversion, бездоганно інтегровані в Android Studio, що сприяє спільній розробці, стратегіям розгалуження та процесам перевірки коду. Це особливо корисно для команд розробників, які практикують Agile, DevOps або інші ітеративні методології програмної інженерії.

Android Studio також підтримує архітектуру на основі плагінів, що дозволяє налаштовувати середовище розробки відповідно до конкретних вимог проекту. Розробники можуть використовувати величезну екосистему плагінів для лінтингу, локалізації, дизайну інтерфейсу користувача та інтеграції зі сторонніми сервісами, такими як Firebase, який пропонує хмарні функції бекенду, аналітику, звіти про збої та автентифікацію користувачів.

4.2 Flutter

Flutter – це набір інструментів для розробки програмного забезпечення з відкритим кодом, створений та підтримуваний Google для створення власно скомпільованих додатків з єдиної кодової бази. Він дозволяє розробляти додатки для різних платформ, включаючи Android, iOS, веб, настільні комп'ютери (Windows, macOS, Linux) та вбудовані системи, використовуючи єдину модель програмування та візуальну архітектуру. З моменту свого стабільного випуску в 2018 році Flutter отримав значну популярність серед розробників та підприємств завдяки своїй продуктивності, гнучкості та орієнтованим на розробників інструментам.

В основі Flutter лежить мова програмування Dart, яка також розроблена Google. Це об'єктно-орієнтована мова програмування на основі класів, яка поєднує функції JavaScript, Java та C#, пропонуючи надійну безпеку null,

асинхронне програмування, а також компіляцію just-in-time (JIT).

Однією з визначальних інновацій Flutter є його віджет-орієнтована архітектура. У Flutter кожен візуальний, структурний та функціональний елемент є віджетом. Ця ієрархічна, компонована модель надає розробникам декларативний підхід до дизайну інтерфейсу користувача, полегшуючи створення складних, реактивних інтерфейсів з мінімальним імперативним кодом. Механізм рендерингу Skia дозволяє Flutter обходити нативні компоненти інтерфейсу користувача та натомість малювати кожен піксель на екрані, забезпечуючи узгоджений вигляд та відчуття на різних платформах.

З точки зору робочого процесу розробки, Flutter робить акцент на швидкій ітерації та візуальному зворотному зв'язку завдяки своєму механізму гострого перезавантаження з відстеженням стану. Ця функція дозволяє розробникам вносити зміни коду в запущену програму без перезапуску середовища з відстеженням стану, що значно пришвидшує розробку та налагодження інтерфейсу користувача. У поєднанні з інтегрованим інтерфейсом командного рядка та сумісністю з основними IDE (такими як Android Studio, IntelliJ IDEA та Visual Studio Code), Flutter забезпечує продуктивний та гнучкий досвід розробки.

Flutter також включає потужний набір інструментів для тестування та контролю якості, підтримку модульного, віджетного та інтеграційного тестування. Він пропонує вбудовану підтримку конвеєрів безперервної інтеграції (CI) та розгортання (CD) і зазвичай використовується разом з Firebase – мобільною платформою Google для бекенд-сервісів, аналітики та синхронізації даних у режимі реального часу. Екосистема включає репозиторій pub.dev, який містить тисячі пакетів та плагінів, що розширюють можливості Flutter, починаючи від обробки платежів і закінчуючи зв'язком Bluetooth.

Незважаючи на численні переваги, Flutter також має певні недоліки. Залежність фреймворку від власного механізму рендерингу, хоча й корисна

для узгодженості, призводить до більших бінарних файлів програм порівняно з повністю нативними програмами. Крім того, хоча спільнота та екосистема швидко зростають, Flutter не завжди може мати негайну або зрілу підтримку для кожної платформо-специфічної функції або стороннього SDK, особливо в таких сферах, як розширена доступність або інтеграції на рівні підприємства.

4.3 Бібліотеки

У цьому проєкті інтегровано три зовнішні бібліотеки Flutter для підтримки основної функціональності та покращення взаємодії з користувачем: flutter_bluetooth_serial, permission_handler та lottie. Кожна з цих бібліотек відіграє певну та незамінну роль в архітектурі програми. flutter_bluetooth_serial сприяє зв'язку Bluetooth, дозволяючи програмі підключатися та взаємодіяти з зовнішнім годинником. permission_handler відповідає за керування дозволами виконання, які є критично важливими для доступу до Bluetooth та місцезнаходження в сучасних системах Android. lottie використовується для створення візуально привабливої анімації, покращення зворотного зв'язку з користувачем та інтерактивності інтерфейсу. Хоча всі три бібліотеки надають цінні можливості, вони також вносять певні компроміси та міркування щодо підтримки платформи, продуктивності та складності розробки.

Бібліотека flutter_bluetooth_serial є центральною для цієї програми, забезпечуючи базовий рівень зв'язку Bluetooth Classic. Вона дозволяє програмі виявляти пристрої Bluetooth поблизу, ініціювати послідовні з'єднання, обмінюватися потоками даних та керувати станами з'єднань – все це в середовищі Flutter. Однією з ключових переваг цієї бібліотеки є її відносна простота та відповідність стандартній моделі розробки Flutter, яка використовує асинхронні потоки та ф'ючерси для обробки виявлення та зв'язку пристроїв. Вона абстрагує значну частину платформно-специфічної

складності, яка зазвичай пов'язана з роботою з Bluetooth на Android, і дозволяє розробникам взаємодіяти з пристроями Bluetooth, використовуючи знайомі парадигми Flutter.

Однак, flutter_bluetooth_serial має деякі помітні обмеження. По-перше, вона створена спеціально для Bluetooth Classic, що означає, що вона не підтримує Bluetooth Low Energy (BLE) – більш сучасний та енергоефективний протокол, який використовується багатьма новими пристроями. Це обмеження може стати перешкодою, якщо додатку в майбутньому потрібно буде взаємодіяти з ширшим спектром обладнання. Крім того, ця бібліотека наразі підтримується лише на Android, без офіційної сумісності з iOS, що обмежує кросплатформні можливості додатка. Підтримка та обслуговування спільноти також помірні; хоча бібліотека функціональна та достатньо задокументована, темпи її розробки можуть не відповідати швидкому розвитку Flutter, що може призвести до проблем сумісності з новими версіями SDK. Незважаючи на ці недоліки, для проектів, які спеціально потребують зв'язку Bluetooth Classic на Android, flutter_bluetooth_serial залишається одним із найефективніших рішень, доступних в екосистемі Flutter.

Бібліотека permission_handler вирішує критичний аспект сучасної мобільної розробки: керування динамічними запитами на дозволи під час виконання. Зі зростанням обмежень конфіденційності та платформо-специфічними відмінностями в моделях дозволів, особливо на Android 10 та вище, правильне керування дозволами Bluetooth та місцезнаходження є надзвичайно важливим. permission_handler абстрагує значну частину складності платформи та надає єдиний API для запиту, перевірки та обробки станів дозволів як у середовищах Android, так і iOS. Він бездоганно інтегрується з асинхронною моделлю програмування Flutter, що спрощує перевірку або запит дозволів перед виконанням операцій, які від них залежать, таких як сканування Bluetooth або підключення до пристрою.

Однією з головних переваг permission_handler є його міжплатформна

узгодженість. Він обробляє розбіжності, пов'язані з дозволами, між Android та iOS з мінімальним втручанням розробника, що покращує підтримку коду та зменшує ризик помилок під час виконання. Він також підтримує широкий спектр типів дозволів, окрім Bluetooth та місцезнаходження, що робить його універсальним інструментом для програм, яким потрібен доступ до датчиків, сховища медіа або системних налаштувань.

Однак бібліотека має кілька недоліків. Процес інтеграції бібліотеки в існуючу кодову базу може призвести до ускладнень під час збірки, якщо ним не керувати належним чином, особливо в проектах, що використовують інші нативні плагіни. Незважаючи на ці складнощі, permission_handler широко вважається стандартним рішенням для програм Flutter, які потребують розширеного керування дозволами, а його надійність та стабільність зробили його фаворитом галузі.

Бібліотека Lottie використовується в цьому проєкті для рендерингу анімації станів Bluetooth-з'єднання. Ці анімації можна легко інтегрувати в інтерфейси Flutter без значних накладних витрат на продуктивність, забезпечуючи вишуканий та динамічний візуальний досвід, який покращує залученість користувачів та ефективніше передає зміни стану програми, ніж статичні значки чи текст.

Однією з ключових переваг використання Lottie в проєкті Flutter є можливість включати складні векторні анімації без суттєвого впливу на розмір програми чи продуктивність під час виконання. Анімації Lottie не залежать від роздільної здатності, що означає, що вони чудово масштабуються на пристроях з різними розмірами екрана та щільністю пікселів. Крім того, пакет Lottie Flutter надає гнучкий API, який підтримує керування відтворенням анімації, циклічне завантаження та логіку умовного відображення. Це робить його корисним для передачі динамічної інформації у режимі реального часу.

4.4 Розробка мобільного застосунку

Основна мета застосунку – забезпечити інтуїтивно зрозумілий та адаптивний інтерфейс користувача, який дозволяє налаштовувати час та дату годинника відповідно внутрішнього часу пристрою або відправляти власне значення, встановлювати та вимикати будильники та запускати таймери за допомогою бездротового зв'язку.

З технічної точки зору, застосунок використовує пакет flutter_bluetooth_serial для встановлення та керування класичними з'єднаннями Bluetooth, а також пакет permission_handler для динамічного запиту та обробки необхідних дозволів на виконання для сканування, підключення та служб визначення місцезнаходження Bluetooth, що особливо важливо для пристроїв Android, що працюють із сучасними рівнями API.

Функція main служить точкою входу програми. Вона починає з перевірки правильної ініціалізації всіх прив'язок Flutter за допомогою WidgetsFlutterBinding.ensureInitialized(), що є вирішальним кроком під час виконання асинхронних операцій перед runApp().

Лістинг 4.1 – Функція main

```
void main() async {
 WidgetsFlutterBinding.ensureInitialized();
 await requestBluetoothPermissions();
 runApp(MyApp());
}
```

Після цього вона викликає requestBluetoothPermissions() для отримання необхідного доступу до Bluetooth та місцезнаходження. Ця функція виконується під час запуску програми, щоб переконатися, що програма має необхідні привілеї перед початком будь-яких дій, пов'язаних з Bluetooth.

Лістинг 4.2 – Функція requestBluetoothPermissions

```
Future<void> requestBluetoothPermissions() async {
 await [
  Permission.bluetoothScan,
  Permission.bluetoothConnect,
  Permission.location
 ].request();
}
```

Для забезпечення коректної роботи з дозволами їх потрібно додати до файлу маніфесту.

Лістинг 4.3 – Прописані дозволи

```
  <uses-permission android:name="android.permission.BLUETOOTH" />
  <uses-permission
android:name="android.permission.BLUETOOTH_ADMIN" />
  <uses-permission
android:name="android.permission.BLUETOOTH_CONNECT" />
  <uses-permission
android:name="android.permission.BLUETOOTH_SCAN" />
  <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
  <uses-permission
android:name="android.permission.BLUETOOTH_ADVERTISE" />

<queries>
    <intent>
      <action
android:name="android.bluetooth.device.action.REQUEST_DISCOVERAB
LE" />
    </intent>
    <intent>
      <action
android:name="android.bluetooth.adapter.action.REQUEST_ENABLE" />
    </intent>
  </queries>
```

Логіка підключення інкапсульована в асинхронні методи, які намагаються підключитися до попередньо визначеної MAC-адреси, що відповідає цільовому пристрою Bluetooth. Якщо підключення успішне,

програма прослуховує вхідні дані та відповідно оновлює інтерфейс користувача, щоб відобразити стан підключення. Відключення обробляється коректно, забезпечуючи належне вивільнення ресурсів.

Лістинг 4.4 – Функції, зо відповідають за роботу з Bluetooth

```
 BluetoothConnection? connection;
 bool isConnected = false;
 bool isConnecting = false;

 Future<void> _connectToDevice() async {
  setState(() => isConnecting = true);
  try {
   connection                          =                          await
BluetoothConnection.toAddress("00:23:09:01:6C:EC");
   setState(() {
    isConnected = true;
    isConnecting = false;
   });
   connection!.input!.listen((data) {}).onDone(() {
    setState(() => isConnected = false);
   });
  } catch (e) {
   setState(() {
    isConnecting = false;
    isConnected = false;
   });
  }
 }

 void _disconnect() {
  connection?.dispose();
  connection = null;
  setState(() => isConnected = false);
 }

 void _toggleConnection() {
  if (isConnected) {
   _disconnect();
  } else {
   _connectToDevice();
  }
 }
```

Зв'язок із зовнішнім пристроєм реалізується шляхом надсилання

текстових команд через послідовний вихідний потік Bluetooth. Команди форматуються відповідно до попередньо визначеного протоколу, зрозумілого годиннику, адже саме такі команди було запрограмовано до плати Arduino. Наприклад, для встановлення часу та дати використовується рядок типу DATETIME:HH:MM:SS:DD:MM:YYYY, ініціювання будильника за допомогою ALARM:HH:MM, його зупинка за допомогою ALARM_OFF або запуск таймера із загальною кількістю секунд, що встановлюється як TIMER:<секунди >.

Лістинг 4.5 – Функції комунікації з годинником

```
 void            sendAlarm(String            timeStr)            =>
_sendCommand("ALARM:$timeStr");

 void stopAlarm() => _sendCommand("ALARM_OFF");
 void sendTimer(int seconds) => _sendCommand("TIMER:$seconds");

 void sendDateTime(DateTime dt) {
  final command =
    "DATETIME:${dt.hour.toString().padLeft(2,
'0')}:${dt.minute.toString().padLeft(2,
'0')}:${dt.second.toString().padLeft(2,
'0')}:${dt.day.toString().padLeft(2,
'0')}:${dt.month.toString().padLeft(2, '0')}:${dt.year}";
  _sendCommand(command);
 }
```

Функція _sendCommand приймає рядкову команду як вхідні дані, форматує її з символом нового рядка, перетворює на Uint8List та записує у вихідний потік Bluetooth. Вона також очікує на надсилання всіх даних за допомогою allSent, забезпечуючи цілісність повідомлення. Ця функція формує комунікаційну основу програми та використовується кількома іншими функціями для надсилання структурованих команд на зовнішній пристрій.

Лістинг 4.6 – Функція _sendCommand

```
void _sendCommand(String command) {
  if (connection != null && connection!.isConnected) {

connection!.output.add(Uint8List.fromList("$command\n".codeUnits
));
   connection!.output.allSent;
  }
 }
```

Взаємодія з користувачем здійснюється за допомогою модальних вікон які дозволяють синхронізувати годинник з поточним часом телефону або дозволяє вручну вибирати власну дату та час. Аналогічно, конфігурації будильника та таймера представлені за допомогою інтуїтивно зрозумілих та адаптивних діалогових вікон, що спрощують вибір часу та ініціювання команд. Наприклад, функція _showAlarmModal запускає діалогове вікно вибору часу, щоб користувачі могли вибрати час будильника. Після вибору час форматується та передається до sendAlarm.

Лістинг 4.7 – Функція _showAlarmModal

```
 void _showAlarmModal() async {
   TimeOfDay selectedTime = TimeOfDay.now();

   final pickedTime = await showTimePicker(
    context: context,
    initialTime: selectedTime,
   );

   if (pickedTime != null) {
    selectedTime = pickedTime;
    sendAlarm("${selectedTime.hour.toString().padLeft(2,
'0')}:${selectedTime.minute.toString().padLeft(2, '0')}");
   }
 }
```

ВИСНОВКИ

У ході виконання кваліфікаційної роботи були успішно реалізовані всі поставлені завдання. Було спроєктовано та створено електронний годинник на базі мікроконтролера Arduino Uno, який виконує не лише базові функції відображення часу, але й має можливість працювати як будильник, таймер, вимірює температуру та здійснює обмін даними з телефоном через Bluetooth.

Для візуалізації інформації та зручної взаємодії з користувачем було обрано дисплей LCD 1602 I2C, що дозволяє економно використовувати виводи плати Arduino та забезпечує чітке зображення даних. Також було використано модуль реального часу DS3231SN, цифровий температурний датчик DS18B20, звуковий модуль buzzer та Bluetooth-модуль HC-06.

Окрім апаратної частини, важливим етапом проєкту стала розробка мобільного застосунку на платформі Flutter, який забезпечує зручний інтерфейс керування годинником. За допомогою Bluetooth-з'єднання мобільний застосунок дозволяє налаштовувати час, керувати будильником та запускати таймер.

Програмне забезпечення для мікроконтролера було реалізовано у середовищі Arduino IDE, де відпрацьовано логіку роботи годинника, зчитування показників з датчиків, опрацювання сигналів та передача даних на мобільний застосунок.

Після завершення розробки обидві частини – апаратна та програмна пройшли етап тестування. Усі модулі взаємодіють коректно, функції виконуються згідно з вимогами, інтерфейс мобільного застосунку є зручним для кінцевого користувача.

У результаті роботи було створено повноцінну інтегровану систему: електронний годинник із розширеним функціоналом та мобільним застосунком для керування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Бокселл Дж. Arduino Workshop: A Hands-On Introduction with 65 Projects. – San Francisco : No Starch Press, 2013. – 392 с.

2. Tooley M. Electronic applications and the Arduino // Electronic Circuits. – 2019. – С. 371–399. – URL: https://www.taylorfrancis.com/chapters/mono/electronic-applications-arduino-mike-tooley/ (дата звернення: 01.05.2025)

3. Баран В. С., Власюк Г. Г., Оникієнко Ю. О., Смоленська О. І. Основи мікропроцесорної техніки. – Київ : КПІ ім. Ігоря Сікорського, 2019. – 140 с.

4. Путятін Є. П., Любченко В. А., Кобилін О. А., Руденко Д. О. Основи програмування мовою С++ : навч. посіб. – 2018. – 282 с.

5. Arduino Uno Specifications [Електронний ресурс] – URL: https://spiceman.net/arduino-uno/ (дата звернення: 01.05.2025)

6. Офіційна документація Arduino [Електронний ресурс] – URL: https://www.arduino.cc/ (дата звернення: 01.05.2025)

7. Devadze D. Vector-Deductive Memory-Based Transactions for Fault-As-Address Simulation / D. Devadze, Z. Davitadze, A. Hahanova //12th International Conference on Dependable Systems, Services and Technologies (DESSERT), Athens, Greece, 2022: proceedings. – IEEE, 2022. – P. 1–6.

8. Hahanov V. Vector-Logical In-Memory Simulation of Faults as Truth Table Addresses / V. Hahanov, E. Litvinova, H. Hahanova, S. Chumachenko, Z. Davitadze, I. Hahanova, H. Kulak, V. Ponomarova, V. H. Abdullayev // 2024 IEEE East-West Design & Test Symposium (EWDTS), Yerevan, Armenia, 2024, pp. 1-6,

9. Intelligent Computing for Cybersocial Space / [V. Hahanov, E. Litvinova, Z. Davitadze, S. Chumachenko, D. Devadze, M. Abashidze] // International Black Sea Conference on Communications and Networking (BlackSeaCom), Tbilisi, Georgia, 24-27 June 2024: proceedings. – IEEE, 2024. – P. 108–113.