Міністерство освіти і науки України

Луцький національний технічний університет
(повне найменування закладу вищої освіти)

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

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

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

СМАРТ-БУДИЛЬНИК НА ОСНОВІ ARDUINO З ІНТЕГРАЦІЄЮ МОБІЛЬНОГО ДОДАТКУ

ARDUINO-BASED SMART ALARM CLOCK WITH MOBILE APP INTEGRATION

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

освітня програма Комп'ютерна інженерія
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІ-42
Бицюк Володимир Ярославович

(підпис)

Керівник:
ст.викладач
Міскевич Оксана Іванівна

(підпис)

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

(підпис)

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

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

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Бицюку Володимиру Ярославовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Смарт-будильник на основі Arduino з інтеграцією мобільного додатку*

Керівник роботи *ст.викладач Міскевич Оксана Іванівна*

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

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

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
Вступ
Обґрунтування вибору апаратного та програмного забезпечення, методи їх інтеграції
Практична реалізація смарт-будильника і мобільного застосунку
Висновки

5. Перелік графічного (ілюстративного) матеріалу:
Функціональна схема-прототип пристрою смарт-будильника
Інтерфейс додатку смарт-будильника
Інтерфейс додатку при підключені до пристрою та встановлені будильника

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Системний аналіз інформації за темою дослідження	*Міскевич О.І., ст.викладач*		
Обґрунтування вибору апаратного та програмного забезпечення, методи їх інтеграції	*Міскевич О.І., ст.викладач*		
Практична реалізація смарт-будильника і мобільного застосунку	*Міскевич О.І., ст.викладач*		
Нормоконтроль	*Багнюк Н.В., доцент*		
Гарант ОП	*Лавренчук С.В., доцент*		
Показник запозичень тексту	***___%***		
Академічна доброчесність	*Міскевич О.І.,ст.викладач*		

7. Дата видачі завдання 10.01.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	*Огляд літератури із досліджуваної проблеми, аналіз предметної області та наявних рішень*	10.02.2025 р.	Виконано
2.	*Вибір апаратної та програмної бази для проєкту*	02.03.2025 р.	Виконано
3.	*Практична реалізація смарт-будильника і мобільного застосунку*	02.04.2025 р.	Виконано
4.	*Висновки та пропозиції*	10.04.2025 р.	Виконано
5.	*Формування списку використаних джерел*	15.04.2025 р.	Виконано
6.	*Формування додатків*	02.05.2025 р.	Виконано
7.	*Оформлення ілюстративного матеріалу*	10.05.2025 р.	Виконано
8.	*Представлення остаточного варіанту кваліфікаційної роботи керівнику*	15.06.2025 р.	Виконано
9.	*Нормоконтроль*	30.05.2025 р.	Виконано
10.	*Інструментальна перевірка на академічний плагіат*	03.06.2025 р.	Виконано
11.	*Представлення кваліфікаційної та всіх супровідних документів на кафедру*	10.06.2025 р.	Виконано

Здобувач вищої освіти _____ (підпис) Бицюк В.Я. (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ (підпис) Міскевич О.І. (прізвище, ініціали)

АНОТАЦІЯ

Бицюк В.Я. Смарт-будильник на основі Arduino з інтеграцією мобільного додатку. Рукопис.

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

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

У першому розділі було проаналізовано розумні побутові пристрої і обґрунтовано доцільність їх використання у повсякденному житті. Розглянуто основні напрямки розвитку смарт-технологій, зокрема їх взаємодію з мобільними застосунками, а також проведено короткий огляд сучасних рішень і проєктів, пов'язаних із побутовою автоматикою.

Другий розділ присвячено технічному обґрунтуванню обраної концепції пристрою та його завдання. Описано детально складові апаратної частини, включаючи мікроконтролер Arduino, модуль годинника реального часу DS1302, рідкокристалічний LCD дисплей з інтерфейсом I2C та Bluetooth-модуль HC-05. Розглянуто переваги та недоліки окремих елементів пристрою. Багато уваги приділено особливостям програмної частини, яка забезпечує роботу будильника, зберігання часу, відображення інформації та обмін даними з мобільним додатком.

У третьому розділі описано практичну реалізацію проєкту. Представлено загальну архітектуру рішення, схему підключення елементів, а також описано логіку взаємодії між апаратною платформою та мобільним застосунком, створеним у середовищі Android Studio. Проведено базове тестування пристрою, окреслено можливості його подальшого вдосконалення та адаптації для більш широких задач.

Ключові слова: розумний будильник, Arduino UNO, Bluetooth, мобільний застосунок, LCD, HC-05, RTC DS1302.

ANNOTATION

Bytsiuk V. Smart alarm clock based on Arduino with integration of a mobile application. Manuscript.

Qualification work for bachelor's degree in «Computer Engineering», speciality 123 Computer Engineering. Lutsk National Technical University. Lutsk, 2025.

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

The first chapter analyses smart household devices and justifies the feasibility of their use in everyday life. The main areas of smart technology development, including their interaction with mobile applications, are discussed, and a brief overview of modern solutions and projects related to home automation is provided.

The second section is devoted to the technical justification of the chosen device concept and its tasks. The hardware components are described in detail, including the Arduino microcontroller, the DS1302 real-time clock module, the LCD display with I2C interface, and the HC-05 Bluetooth module. The advantages and disadvantages of the individual elements of the device are discussed. Much attention is paid to the features of the software part, which provides the alarm clock, time storage, information display and data exchange with a mobile application.

The third section describes the practical implementation of the project. It presents the overall architecture of the solution, the wiring diagram, and the logic of interaction between the hardware platform and the mobile application created in the Android Studio environment. The basic testing of the device is carried out, and the possibilities of its further improvement and adaptation for wider tasks are outlined.

Keywords: smart alarm clock, Arduino UNO, Bluetooth, mobile application, LCD, HC-05, RTC DS1302.

ЗМІСТ

ВСТУП

У сучасному світі засоби автоматизації та інтелектуальні пристрої відіграють дедалі важливішу роль у повсякденному житті людини. Особливої актуальності набувають технології, пов’язані з інтернетом речей, які дозволяють створювати пристрої, що взаємодіють між собою та з користувачем. Одним із прикладних напрямів є розробка побутових інтелектуальних пристроїв, таких як смарт-будильники. Традиційні будильники поступово відходять на другий план, поступаючись багатофункціональним, персоналізованим пристроям, які враховують індивідуальні параметри користувача та мають інтерактивний інтерфейс. Враховуючи стрімкий розвиток мікроконтролерної техніки та мобільних технологій, розробка смарт-будильника з інтеграцією до мобільного застосунку є актуальним завданням, що має практичну цінність.

Метою роботи є розробка прототипу інтелектуального будильника на базі апаратної платформи Arduino з можливістю віддаленого керування через мобільний застосунок, що забезпечує зручність, гнучкість та розширену функціональність пристрою.

Об’єкт дослідження – системи розумних пристроїв, побудованих на основі мікроконтролерів для автоматизації побутових процесів.

Предмет дослідження – архітектура та функціональна реалізація смарт-будильника з інтеграцією мобільного інтерфейсу на базі апаратної платформи Arduino.

Завдання, які необхідно виконати в межах кваліфікаційної роботи:

– реалізувати апаратну частину смарт-будильника із застосуванням мікроконтролера Arduino, модулів реального часу і виконавчих пристроїв;

– розробити програмне забезпечення для керування пристроєм, включаючи логіку сигналізації, зчитування сенсорних даних та взаємодії з зовнішніми пристроями;

– дослідити можливості інтеграції мобільного застосунку з мікроконтролером через інтерфейс Bluetooth, забезпечивши зручне управління параметрами будильника;

– візуалізувати функціонування системи за допомогою дисплея, а також створити інтерфейс користувача у мобільному додатку;

– спроектувати загальну архітектуру пристрою, включаючи схему підключення модулів, логіку взаємодії між елементами та структурну модель системи;

– запропонувати можливі шляхи подальшої модернізації пристрою з урахуванням актуальних тенденцій у сфері IoT.

РОЗДІЛ 1
СИСТЕМНИЙ АНАЛІЗ ІНФОРМАЦІЇ ЗА ТЕМОЮ ДОСЛІДЖЕННЯ

1.1 Актуальність розвитку смарт-пристроїв у побуті

В сучасному світі інформаційні технології дедалі глибше інтегруються в повсякденне життя людини, зростання популярності концепції «розумного дому» є яскравим свідченням данного процесу. Смарт-пристрої поступово стають не лише технологічною новинкою, а й практичним інструментом для підвищення повсякденного комфорту, безпеки та енергоефективності. Особливо помітною є тенденція персоналізації побутової техніки, яка здатна адаптуватися до індивідуальних потреб кожного користувача.

Інтернет речей, або IoT – це мережа фізичних пристроїв, оснащених програмним забезпеченням, датчиками та іншими механізмами для обміну даними та їх підключення до інших систем і пристроїв.

Технологія інтернет речей функціонує як глобальна система інформаційного суспільства, надаючи модернізованим сервісам можливість об’єднуватися та обмінюватися речами на основі уже існуючих та механізмів зв’язку які ще розвиваються. Крім того, вона надає сумісні дані та може самостійно взаємодіяти без втручання людини.

В результаті, очікується, що ця технологія створюватиме все нові потоки доходів, підвищить ефективність бізнесу та оброблюватиме те, як надаються поточні послуги в різних секторах.

Аналіз ринку інтернет речей: «Глобальний ринок Інтернету речей у 2023 році оцінювався в 595,73 млрд доларів США. Прогнозується, що ринок зросте з 714,48 млрд доларів США у 2024 році до 4062,34 млрд доларів США до 2032 року, демонструючи середньорічний темп зростання CAGR 24,3 % протягом прогнозованого періоду» [1]. Ця галузь розвивається шаленими темпами, вона є дуже перспективною в комерційному плані.

Однією з актуальних задач у цьому напрямі є автоматизація процесу пробудження. Смарт-будильники, на відміну від традиційних механічних або

електронних аналогів, забезпечують гнучкість налаштування, інтеграцію з мобільними пристроями та можливість реагувати на зміни навколишнього середовища. Під час досліджень виявив, що в науковій літературі підкреслюється важливість врахування біоритмів людини для ефективного та менш стресового пробудження [2].

1.2 Технологічна база реалізації пристрою

В основі проєкту лежить використання мікроконтролерної платформи Arduino, яка завдяки відкритій архітектурі, численним бібліотекам і простоті використання широко застосовується в навчальних і прикладних розробках. Вона дозволяє створювати повноцінні прототипи, або ж і повноцінні програмні продукти електронних пристроїв із підтримкою сенсорів, дисплеїв, модулів зв’язку тощо.

Задля точного вимірювання часу в системі смарт-будильника доцільно використовувати модулі реального часу RTC, наприклад, DS1302, або ж DS1307. Модуль реального часу дозволяє забезпечити незалежність системи від живлення та високу точність сигналу. Виведення даних реалізується за допомогою дисплея, зокрема розглядаю варіант LCD-екрана, або ж семисегментний екран з чотирма цифрами, що забезпечує зручне відображення часу. Мобільне керування здійснюється через Bluetooth-модуль типу HC-05 або HC-06, які забезпечують двосторонню передачу даних між мікроконтролером та смартфоном. При необхідності це можна буде використати в подальших оновленнях пристрою.

Зі зростанням популярності бездротових технологій з’являється можливість використовувати альтернативні методи інтеграції, зокрема Wi-Fi або протоколи IoT, проте Bluetooth залишається найбільш доступним та енергоефективним для базових систем. Також базовий варіант проєкту поки не потребує такої технології.

1.3 Значення мобільної інтеграції

Використання мобільного додатку значно підвищує зручність взаємодії з пристроєм, розширює функціонал та надає користувачеві можливість змінювати налаштування безпосередньо зі смартфона. А це в свою чергу є невід'ємною складовою якісного продукту. Також смартфон дозволяє уникнути додавання фізичних кнопок до будильника, через нераціональність. Для реалізації простих застосунків часто використовують середовище MIT App Inventor, яке дозволяє створювати Android-додатки у візуальному конструкторі, або редакторі без глибоких знань програмування. Або ж гарною альтернативою є – Android Studio, додаток дозволяє з допомогою не складного програмного коду реалізувати чудовий додаток. Такий підхід є особливо корисним у навчальному процесі та для швидкого створення прототипів [3].

Водночас у середовищі Android Studio можна реалізовувати більш складні проєкти з використанням мов Java або Kotlin. Дане програмне забезпечення дозволяє з допомогою не складного програмного коду реалізувати чудовий додаток. В цій прграмі можжна створювати як повноцінні додатки зі складною логікою та дизайном так і прості, або середньої сладності проєкти. Зокрема, використання мобільного інтерфейсу в домашній автоматизації дозволяє вивести зручність користувача на якісно новий рівень.

1.4 Аналіз комерційних рішень і дослідницьких проєктів

На ринку присутні численні моделі розумних будильників – від базових електронних пристроїв із функцією голосового керування до багатофункціональних систем з інтеграцією в екосистеми «розумного дому». Наприклад, Philips SmartSleep (рис. 1.1) має функцію імітації природного освітлення, тоді як пристрої типу Amazon Echo Spot поєднують можливості голосового помічника з годинником та динаміком.

Рисунок 1.1 – Philips SmartSleep [4]

Попри це, більшість таких пристроїв є дорогими та закритими для змін і модернізації. Саме тому рішення, засновані на відкритих апаратно-програмних платформах, таких як Arduino, дозволяють створювати доступні альтернативи з гнучкою конфігурацією та можливістю індивідуального доопрацювання.

Окремо слід відзначити, що дослідники у сфері побутової автоматизації все частіше звертаються до теми розумних будильників як прикладу компактних систем інтелектуального керування. Так, у публікації на IEEE Explore [5] розглядається впровадження багаторежимної системи пробудження з врахуванням змін кліматичних умов у кімнаті, що підтверджує актуальність теми науково та практично.

1.5 Невирішені аспекти та перспективи подальших досліджень

Незважаючи на наявність значного прогресу, існує чимало викликів, пов’язаних із подальшим розвитком систем розумного будинку, включаючи смарт-пробудження. По-перше, це необхідність врахування індивідуальних біоритмів користувача, що потребує впровадження сенсорів пульсу, дихання

або руху, а також алгоритмів обробки цих даних. По-друге, актуальним залишається питання енергозбереження, оскільки більшість пристроїв працює від акумуляторів та і з врахуванням енергетичної кризи в країні питання енергонезалежності стоїть більш гостро, ніж раніше. І по-третє, подальші дослідження повинні зосереджуватись на безпеці з'єднання, особливо в умовах підключення до мережі Інтернет. Зі зростанням важливості застосунків інтернету речей також зростає необхідність аналізу та обробки величезних обсягів даних. Штучний інтелект передбачає використання механізмів машинного навчання для генерації нових даних і, таким чином, став перспективною технологією, яка може допомогти вирішити різні проблеми. Також зараз багато користувачів піклуються про свій здоровий та комфортний сон, а також пробудження. Доступні дослідження штучний інтелект допомагають будь-яким людям краще підходити до питання якісного сну. Одним із помічників буде – смарт-будильник.

У межах даної кваліфікаційної роботи планується реалізувати лише базову версію пристрою, однак із урахуванням подальшого потенціалу його розширення.

1.6 Конкретизація завдань кваліфікаційної роботи

З огляду на проведений аналіз літератури та практичних реалізацій, у межах кваліфікаційної роботи формуються завдання.

По-перше, необхідно реалізувати апаратну частину пристрою з використанням Arduino, модуля реального часу, генератора сигналу, дисплея та Bluetooth-модуля [6]. Цей етап передбачає складання схеми, налаштування живлення, перевірку з'єднань.

Далі слід розробити програмне забезпечення для мікроконтролера, яке забезпечуватиме основний функціонал будильника: зчитування поточного часу, генерацію сигналу, виведення інформації на дисплей, також прийом сигналів по Bluetooth та виконання команд.

Розробка мобільного додатку із зручним та зрозумілим інтерфейсом, що дозволить керувати будильником дистанційно використовуючи Bluetooth.

Наступним етапом стане дослідження взаємодії зі смартфоном, тобто реалізація Bluetooth-зв’язку, який дозволить керувати пристроєм через мобільний застосунок.

Особливу увагу буде приділено проєктуванню архітектури пристрою, включаючи логіку обміну даними між компонентами та послідовність виконання функцій.

РОЗДІЛ 2
ОБҐРУНТУВАННЯ ВИБОРУ АПАРАТНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, МЕТОДИ ЇХ ІНТЕГРАЦІЇ

2.1 Ідея та новизна проєкту

Смарт-будильники не є технологічною новинкою на ринку, однак більшість з них – це комерційні пристрої з обмеженою можливістю персоналізації та розширення функціоналу. Запропонований у моїй роботі підхід полягає у створенні відкритого та гнучкого прототипу, який користувач може модифікувати, вдосконалювати або адаптувати під власні потреби. Новизна реалізації полягає в поєднанні простоти апаратної частини з можливістю індивідуального налаштування будильника через мобільний застосунок. А також у створенні простого додатку для керування будильником. Такий підхід дозволяє використовувати переваги інтернету речей на рівні побутового пристрою, зберігаючи низьку вартість і доступність реалізації.

Ключовою ідеєю проєкту є забезпечення інтерактивного способу керування будильником, що дозволяє не лише встановлювати час сигналу будильника, а й бачити в реальному часі інформацію на дисплеї, адаптувати функціональність без фізичного втручання в пристрій. Завдяки цьому користувач отримує не просто сигналізатор, а мініатюрну систему автоматизації, доступну в навчальних цілях, для домашнього використання або як базу для розширених IoT-рішень.

2.2 Обґрунтування вибору апаратної платформи

На етапі планування проєкту я розглянув декілька варіантів апаратних платформ: Arduino, ESP32 (рис. 2.1), STM32 (рис. 2.2) та Raspberry Pi (рис. 2.3). Остаточний вибір зупинився на Arduino Uno R3. Цей вибір обґрунтовано як технічними характеристиками, так і практичними міркуваннями.

Рисунок 2.1 – Плата ESP32 [7]

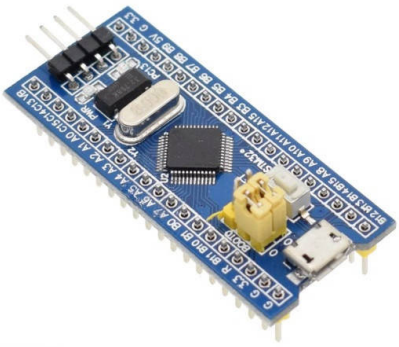

Рисунок 2.2 – Плата STM32 [8]

Рисунок 2.3 – Мікрокомп’ютер Raspberry Pi 5 [9]

«Arduino забезпечує високу стабільність роботи у реальному часі, оскільки не потребує повноцінної операційної системи. Це дозволяє уникнути затримок, характерних для платформ на базі Linux (як у випадку з Raspberry Pi)» [10]. Arduino має мінімальне енергоспоживання, що важливо для автономного пристрою. Крім того, Arduino ідеально підходить для навчальних і прототипових задач – він має просту архітектуру, широку підтримку середовищ розробки, доступну документацію та велику кількість бібліотек, а також простий у використанні, має багато доступних додаткових модулів які можна використати для вдосконалень.

У порівнянні з ESP32, який має більші апаратні можливості: двоядерний процесор, Wi-Fi, Bluetooth Low Energy; Arduino виграє завдяки простоті. Використання ESP32 могло б ускладнити як розробку, так і тестування, особливо для завдань, де не потрібна мережна передача даних. У свою чергу STM32 вимагає глибших знань в області мікроконтролерного програмування, а Raspberry Pi – значно більших ресурсів і джерела живлення, що робить його недоцільним для простого будильника. З огляду на всі вищезазначені аспекти, вибір саме Arduino Uno є логічним та виправданим. Ця плата дозволяє швидко перейти від ідеї до реалізації, не витрачаючи зайвого часу на налаштування середовища або вирішення апаратних конфліктів. Завдяки простоті та надійності, Arduino зменшує ймовірність помилок як у коді, так і в апаратному забезпеченні, що особливо важливо для проєктів з обмеженими часовими рамками чи навчальних цілей. Крім того, Arduino Uno має дружнє середовище розробки Arduino IDE, підтримується великою спільнотою розробників, працює у реальному часі без потреби в ОС і є доступним за ціною. Arduino Uno забезпечує достатній функціонал для реалізації основних завдань без надмірної складності, що робить його оптимальним вибором у цьому контексті [11].

Таким чином, вибір Arduino є зваженим компромісом між функціональністю, простотою та стабільністю. Його переваги у цьому проєкті полягають у швидкому реагуванні, низькому споживанні енергії, широкій підтримці модулів і мінімальних вимогах до ресурсів ну і звісно в низькій ціні.

2.2.1 Порівняння плат Arduino

Серед багатьох плат Arduino особливе місце посідає Arduino Uno. Макетна плата, яка вважається найбільш збалансованим варіантом для реалізації широкого кола проєктів. На відміну від компактнішої плати Arduino Nano (рис. 2.4), Uno забезпечує зручніший доступ до виводів, краще підходить для роботи з макетними платами та має повнорозмірний USB-порт для живлення і програмування. Хоча Nano повторює технічні характеристики Uno, її форм-фактор створює певні незручності при створенні прототипів, особливо без навичок пайки.

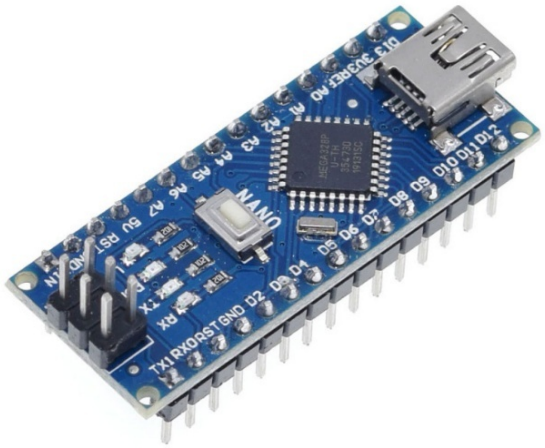

Рисунок 2.4 – Плата Arduino Nano [12]

Arduino Mega, у свою чергу, пропонує значно більшу кількість портів та обсяг пам’яті, що робить її доречною для великих та складних систем з численними периферійними пристроями. Однак великі габарити, вищі вимоги до енергоспоживання та недоцільність її використання у проєктах середнього рівня складності знижують її універсальність. Подібна ситуація з Arduino Leonardo: незважаючи на цікаву можливість емулювати HID-пристрої, ця плата має специфічні відмінності в роботі з USB, що може створювати труднощі при використанні сторонніх модулів.

Одним варіантів є Arduino Pro Mini, який добре підходить для вбудованих рішень, однак вимагає окремого програматора та досвіду роботи з зовнішнім обладнанням. Така особливість значно ускладнює початкову розробку та тестування.

У підсумку, Arduino Uno (рис. 2.5) залишається найбільш універсальним та практичним вибором для створення, бо поєднує достатні технічні характеристики, доступність, легкість використання, широку підтримку з боку спільноти розробників, а також стабільну роботу з більшістю готових модулів та шилдів. Саме ці властивості роблять її оптимальним рішенням як для початкових етапів навчання, так і для створення повноцінних прототипів.

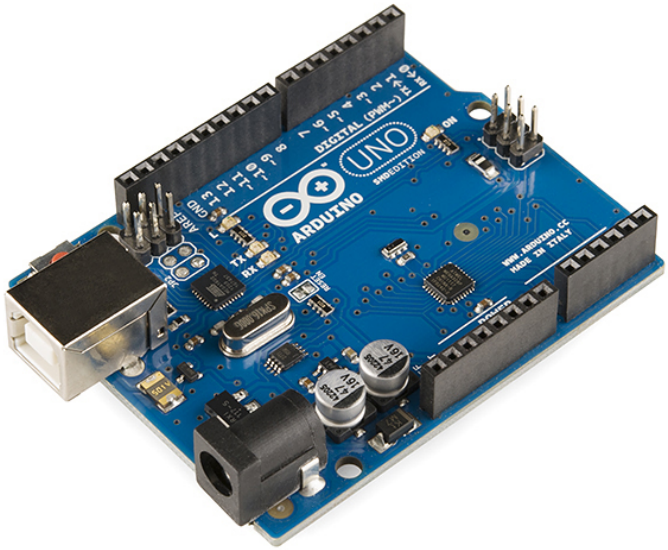

Рисунок 2.5 – Плата Arduino Uno [13]

2.2.2 Характеристика та структура плати Arduino Uno

Arduino Uno є однією з найвідоміших і найпоширеніших моделей серед усіх представників сімейства мікроконтролерних платформ Arduino. Свою популярність вона здобула завдяки поєднанню доступності, зручності у використанні, широкої підтримки спільноти та великої кількості навчальних матеріалів. Цю плату часто використовують як у навчальних цілях, так і для

створення хобійних або напівпрофесійних проєктів, зокрема в автоматизації, побутовій електроніці, робототехніці та системах моніторингу.

Основою плати є мікроконтролер ATmega328P, який забезпечує виконання основної логіки та обробку програмного коду, написаного користувачем. Цей мікроконтролер працює з тактовою частотою 16 мегагерц і має у своєму розпорядженні 32 кілобайти флеш-пам’яті для зберігання програми, з яких приблизно пів кілобайта зайнято завантажувачем. Для тимчасового зберігання даних під час виконання коду використовується 2 кілобайти оперативної пам’яті SRAM, а для збереження постійних налаштувань чи параметрів, які не повинні стиратись при вимкненні живлення, передбачено 1 кілобайт EEPROM-пам’яті.

Плата Arduino Uno містить 14 цифрових портів, які можуть працювати як у режимі входу, так і в режимі виходу. Деякі з них, зокрема D3, D5, D6, D9, D10, D11 підтримують функцію широтно-імпульсної модуляції, що дозволяє керувати аналогоподібними величинами – наприклад, яскравістю світлодіодів або рівнем звукового сигналу. Окремо варто зазначити порти D0 і D1, які слугують для обміну даними через інтерфейс UART і використовуються під час серійної комунікації, зокрема при підключенні до комп’ютера.

Окрім цифрових портів, Arduino Uno має 6 аналогових входів, які позначаються як A0-A5. Ці входи призначені для зчитування аналогових сигналів у межах від 0 до 5 вольт. Застосування таких входів особливо корисне при підключенні сенсорів, що видають неперервні значення, таких як датчики температури, освітлення або вологості. Аналогові входи забезпечують 10-бітну розрядність перетворення, тобто сигнал перетворюється у числове значення в діапазоні від 0 до 1023.

Для живлення Arduino Uno можна використовувати декілька варіантів. Живлення може подаватися через USB-кабель, який одночасно слугує для програмування та обміну даними з комп’ютером, або через зовнішній роз’єм живлення, що дозволяє підключити адаптер з напругою від 7 до 12 вольт. На платі присутні окремі пінові виходи: VIN для подачі зовнішньої

нестабілізованої напруги, пін 5V для зняття стабілізованого живлення 5 вольт, який може використовуватись для живлення зовнішніх компонентів, а також пін 3.3V, який видає напругу від 3.3 вольта до 50 мА струму. Загальні контакти GND слугують для з'єднання із землею. Є також вивід AREF, який використовується для задання опорної напруги при точному зчитуванні аналогових сигналів, хоча в більшості проєктів він залишається незадіяним.

На платі також передбачено кілька світлодіодів для індикації. Один із них підключений до цифрового порту D13 і часто використовується у тестових прикладах, таких як миготіння blink. Індикатори TX і RX сигналізують про передавання та приймання даних через UART-інтерфейс, а світлодіод ON вказує на наявність живлення на платі.

З погляду програмування, Arduino Uno підтримується середовищем Arduino IDE, у якому користувач пише код мовою, схожою на C/C++. Програма завантажується у мікроконтролер через USB-кабель за допомогою попередньо прошитого завантажувача, що спрощує процес розробки. Уся логіка реалізується через зрозумілі функції, як-от digitalWrite, digitalRead, analogRead, delay, тощо, що робить цю платформу зручною навіть для початківців.

Таким чином, Arduino Uno є універсальним інструментом для створення прототипів електронних пристроїв. Вона забезпечує достатню функціональність для реалізації більшості побутових та навчальних завдань, і в той самий час є максимально відкритою, прозорою та доступною у використанні. Саме тому дана плата була обрана як основа для реалізації смарт-будильника, описаного у цій роботі.

2.3 Опис апаратної частини та її компонування

2.3.1 Модуль вимірювання часу DS1302

Центральним елементом пристрою виступає мікроконтролер Arduino, до якого підключаються всі периферійні модулі. На рисунку 2.6 зображено модуль вимірювання поточного часу DS1302, який забезпечує високу точність і

продовжує відлік навіть при відключенні живлення завдяки вбудованій батареї. У системах, де необхідно забезпечити збереження та точне відображення поточного часу незалежно від наявності живлення, особливо важливим є використання модулів реального часу, або Real Time Clock. У даному проєкті для реалізації функції точного хронометрування використовується модуль DS1302 – досить компактний і надійний мікросхемний блок, який здатен безперервно відраховувати години, хвилини, секунди, дні, місяці та роки.

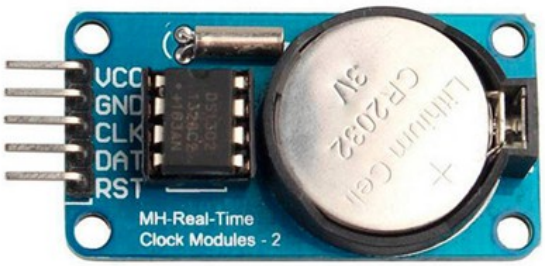

Рисунок 2.6 – Модуль DS1302 для вимірювання поточного часу [14]

Мікросхема DS1302 була розроблена компанією Dallas Semiconductor і є однією з класичних реалізацій модуля реального часу в недорогих мікроконтролерних пристроях. Однією з найважливіших її переваг є наявність вбудованого енергетично незалежного джерела живлення у вигляді батареї типу CR2032. Завдяки цьому модуль може зберігати інформацію про час навіть при повному знеструмленні основного живлення плати Arduino. Як показує практика, енергоспоживання DS1302 у режимі очікування настільки низьке, що одного елемента живлення вистачає на декілька років безперервної роботи.

Функціонально DS1302 виконує відлік часу в двадцятичотирьох або дванадцяти годинному форматі з індикацією AM/PM, що дає змогу гнучко адаптувати його до вимог конкретного проєкту. Модуль має три основні лінії підключення до мікроконтролера: це виводи для передачі даних I/O, синхронізації SCLK та вибору пристрою CE. Такий підхід забезпечує простоту взаємодії з мікроконтролерами, які не мають окремих апаратних інтерфейсів

для роботи з RTC, адже обмін даними відбувається через власний простий послідовний протокол, що легко реалізується програмно.

З технічного боку, DS1302 має вбудований кварцовий генератор з частотою 32,768 кГц, що є стандартом для годинникових пристроїв і забезпечує точний відлік з мінімальним дрейфом. При нормальних умовах експлуатації похибка становить близько 1 хвилини на місяць, що є прийнятним значенням для більшості побутових і навчальних застосувань. Для стабільної роботи мікросхема підтримує діапазон напруги живлення від 2,0 до 5,5 Вольт, що дозволяє зручно поєднувати її з більшістю контролерів, включаючи Arduino Uno, який працює на 5 В.

Окрім поточного часу, DS1302 також містить регістри для збереження дати, включаючи день тижня, місяць, рік і навіть можливість коригування високосних років. Це особливо важливо для довготривалих проєктів, які повинні коректно працювати протягом декількох років без ручного втручання. Крім того, у мікросхемі передбачено 31 байт статичної оперативної пам’яті, яку можна використовувати для зберігання допоміжної інформації, параметрів або налаштувань, що не мають безпосереднього відношення до відліку часу. Ця функція хоч і рідко використовується в простих застосуваннях, проте може бути корисною в складніших системах з обмеженими ресурсами пам’яті.

Під час розробки пристрою типу смарт-будильника важливою перевагою DS1302 є його автономність: користувачеві не потрібно щоразу вручну встановлювати час після втрати живлення, оскільки модуль зберігає поточні показники навіть при відключенні плати Arduino. Це створює враження «розумного» пристрою, що пам’ятає про свої функції й відновлює їх миттєво після перезапуску. Окремо варто згадати і про низький рівень складності інтеграції з Arduino: існує безліч готових бібліотек, які дозволяють зчитувати й записувати час лише за допомогою декількох рядків коду, що значно спрощує процес програмування і робить модуль особливо привабливим для початківців.

У результаті, модуль DS1302 став оптимальним вибором для реалізації точного годинникового функціоналу в межах смарт-будильника. Він забезпечує

стабільність, автономність та простоту використання, а також демонструє гарну сумісність з апаратною платформою Arduino Uno, що дозволяє досягнути поставлених цілей проєкту з мінімальними витратами ресурсів і часу.

2.3.2 Вибір модуля зв'язку

Для зв'язку з мобільним застосунком використовується модуль HC-05 (рис. 2.7), що реалізує передачу даних по Bluetooth-каналу. Завдяки цьому модуль легко сполучається з більшістю смартфонів на Android без потреби у складному налаштуванні. «У порівнянні можливостей та простоти ми рекомендуємо використовувати пристрій HC-05, оскільки його набагато простіше програмувати, ніж HC-06, він має набагато більше можливостей і коштує лише трохи дорожче. З точки зору безпеки, HC-05 дозволяє використовувати пароль довжиною до 16 буквено-цифрових символів, тоді як HC-06 підтримує лише 4-значний PIN-код» [15].

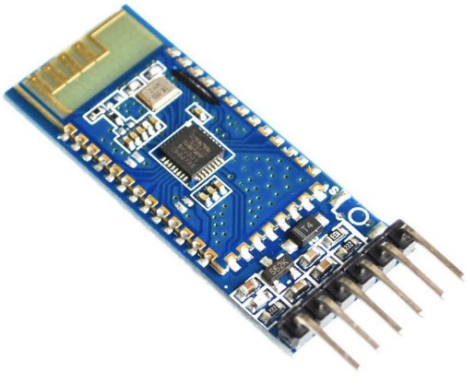

Рисунок 2.7 – Модуль HC-05 для передачі даних по Bluetooth [16]

HC-05 має 6 контактів і може бути як головним, так і веденим пристроєм Bluetooth, тоді як HC-06 має 4 контакти і може бути лише веденим пристроєм. HC-05 має розширений набір AT-команд для програмування, тоді як HC-06 дуже обмежений, дозволяючи змінювати лише швидкість передачі даних, назву та PIN-код. HC-05 має спеціальний режим для надсилання AT-команд для програмування, тоді як HC-06 можна програмувати за допомогою AT-команд будь-коли, коли він не підключений до материнської плати. Здається, що HC-06

є простішим варіантом для програмування, проте для інтерактивного програмування потрібно використовувати послідовну консоль, яка надсилає всю команду за один раз протягом 1 секунди.

2.3.3 Обґрунтування вибору I2C LCD-екрана 1602

Візуальна індикація є невід'ємною складовою будь-якого сучасного електронного пристрою, зокрема й у контексті смарт-будильника. Надзвичайно важливо, щоб користувач мав змогу швидко та зручно отримувати основну інформацію про поточний час, активність будильника, стан підключення чи навіть підтвердження про виконання тієї чи іншої команди. Саме для цього до складу пристрою було інтегровано рідкокристалічний дисплей, який працює за протоколом I2C. У розробці використано дисплей формату 1602, тобто такий, що може відображати до 16 символів у 2 рядках. Це оптимальний варіант для компактних систем, які не потребують виведення великого обсягу графічної інформації.

Інформація про час, статус будильника або інші повідомлення виводиться на компактний LCD-дисплей розміром 0.96 дюйма (рис. 2.8). Він підключений через I2C, що спрощує схему з'єднання. В якості виконавчого елементу обрано п'єзозумер, який подає звуковий сигнал у встановлений час. Також розглядався варіант застосування семисегментного індикатора на чотири цифри, проте його функціонал більш обмежений. На відміну від семисегментного індикатора, I2C LCD-дисплей дозволяє виводити текстову інформацію в зручному та зрозумілому форматі, включаючи повідомлення типу «Alarm ON», «Set: 07:30», «Connected», тощо. Такий спосіб подачі інформації робить пристрій значно зручнішим і зрозумілішим у користуванні, знижує імовірність помилок і підвищує загальну ергономіку.

Завдяки підтримці програмного керування затримками та зчитуванням часу, DS1302 чудово поєднується з іншими модулями у складі системи, не створюючи конфліктів у роботі. Його компактні розміри та мінімальна кількість необхідних виводів дозволяють інтегрувати модуль навіть у найбільш обмежених за простором пристроях. У поєднанні з бібліотекою RTClib

реалізація функціоналу годинника займає мінімум ресурсів мікроконтролера, залишаючи достатньо пам’яті для інших задач.

Рисуноку 2.8 – Дисплей LCD 1602 [17]

Рішення застосувати саме дисплей із інтерфейсом I2C продиктоване насамперед зручністю підключення та економією апаратних ресурсів мікроконтролера. Традиційні дисплеї 1602 без плати-перехідника вимагають підключення щонайменше шести виводів до Arduino, що суттєво обмежує можливості під’єднання додаткових компонентів. У випадку ж використання I2C-адаптера до дисплея достатньо лише двох сигнальних ліній – SDA та SCL, які передають дані та тактові імпульси відповідно. Це дозволяє значно спростити загальну електричну схему, зробити її більш охайною та полегшити налагодження всієї системи в цілому.

Обраний дисплей підтримує типову напругу живлення в 5 В та може функціонувати у широкому температурному діапазоні, що гарантує стабільність роботи як у побутових, так і в дещо складніших умовах експлуатації. Пристрій також має потенціометр, за допомогою якого можна відрегулювати контрастність зображення, що є додатковою перевагою з точки зору користувацького досвіду. Підсвітка екрану на основі світлодіодів дозволяє зчитувати інформацію навіть при поганому освітленні, що особливо важливо

для пристрою, який часто використовується у нічний час або в затемненому приміщенні.

Більшість компонентів з’єднані з допомогою макетної плати, що дозволяє гнучко змінювати схему, тестувати альтернативні конфігурації та вдосконалювати систему без створення друкованої плати на початковому етапі та що головне без пайки.

Загалом, вибір на користь LCD 1602 з I2C-інтерфейсом є зваженим і повністю виправданим з технічної, функціональної та користувацької точки зору. Такий дисплей дозволяє виводити потрібну інформацію компактно, чітко і без перевантаження плати зайвими провідниками. Це забезпечує зручну взаємодію між людиною і пристроєм, що є основною вимогою до будь-якого смарт-девайсу, особливо в рамках концепції Інтернету речей та персональних розумних систем.

2.4 Програмне забезпечення: логіка та реалізація

Програмне забезпечення смарт-будильника розроблено у середовищі Arduino IDE з використанням мови програмування C++. Основна мета програмної частини полягає в реалізації керування функціоналом будильника: зчитування поточного часу з модуля реального часу, порівняння його з заданим часом сигналу, відображення актуальних даних на LCD-дисплеї та прийом команд із мобільного застосунку через Bluetooth-з’єднання.

У процесі реалізації було використано як стандартні бібліотеки, так і сторонні – зокрема, бібліотеки Wire, LiquidCrystal_I2C, DS1302 та SoftwareSerial. Бібліотека Wire відповідає за роботу з шиною I2C, необхідною для спілкування з LCD-дисплеєм, а LiquidCrystal_I2C спрощує роботу з цим дисплеєм. DS1302 – спеціалізована бібліотека для взаємодії з модулем годинника реального часу, що забезпечує точне відображення часу й дати. За обробку Bluetooth-з’єднання відповідає бібліотека SoftwareSerial, яка дозволяє

використовувати програмну емуляцію послідовного порту для обміну даними з модулем HC-05 (або аналогічним).

Архітектура логіки передбачає постійний цикл перевірки поточного часу та стану будильника. Мікроконтролер щосекунди звертається до модуля RTC і зчитує час. Якщо виявляється, що поточний час збігається з встановленим часом сигналу, мікроконтролер активує п’єзобузер, формуючи звуковий сигнал із заданою частотою ввімкнення/вимкнення. При цьому користувач бачить відповідне повідомлення на дисплеї. Для зупинки сигналу достатньо скористатися мобільним застосунком, який передає відповідну команду через Bluetooth.

Окрему увагу приділено відображенню даних на дисплеї. Користувач у режимі реального часу бачить актуальну годину, хвилини, секунди, а також дату. Якщо будильник активовано, у правому верхньому куті екрана з’являється спеціальний символ, який вказує на наявність встановленого сигналу.

Мобільний застосунок створено мовою Java з використанням Android Studio. Він забезпечує простий, інтуїтивно зрозумілий інтерфейс, через який користувач може встановлювати час сигналу, вмикати або вимикати його вручну, а також отримувати інформацію про статус з'єднання з пристроєм. Передача команд здійснюється у вигляді коротких текстових повідомлень, таких як SET:09:22, ON, OFF або STATUS. Після запуску застосунку здійснюється пошук доступних Bluetooth-пристроїв, і при встановленні з’єднання користувач отримує повідомлення про успішне підключення. У випадку виникнення помилки, наприклад, якщо Bluetooth на телефоні або пристрої вимкнений, застосунок видає відповідне повідомлення й блокує можливість встановлення часу будильника до моменту відновлення з’єднання.

Комунікація між застосунком та Arduino відбувається по простому протоколу – текстові команди, що надсилаються з Android-пристрою, приймаються Arduino та аналізуються. У разі отримання команди, наприклад, «T0922», мікроконтролер розпізнає її як інструкцію встановити будильник на

09:22. Команда «off» вимикає активний сигнал і скидає час будильника, повертаючи пристрій у звичайний режим.

Загалом логіка роботи програмного забезпечення смарт-будильника є послідовною, надійною та орієнтованою на максимальну зручність для користувача. Вона поєднує як локальні обчислення мікроконтролера, так і інтерактивну взаємодію з мобільним застосунком, що дозволяє ефективно контролювати пристрій і адаптувати його під індивідуальні потреби.

2.5 Інтеграція апаратної та програмної частин

Система спроектована з урахуванням принципу мінімального з'єднання, що дозволяє значно знизити складність апаратної реалізації та одночасно забезпечити високу стабільність взаємодії між модулями. Основні апаратні компоненти – такі як датчики, виконавчі механізми та модулі зв'язку – комунікують з центральним мікроконтролером через два основні інтерфейси: I2C Inter-Integrated Circuit та UART Universal Asynchronous Receiver-Transmitter.

I2C використовується для підключення такого пристрою, як LCD-дисплеї. Цей інтерфейс дозволяє під'єднувати кілька пристроїв до однієї шини, використовуючи лише два проводи SDA і SCL, що суттєво скорочує загальну кількість з'єднань на макетній платі та мінімізує можливість помилок під час монтажу. Крім того, у I2C передбачено унікальну адресацію кожного підключеного пристрою, що виключає конфлікти на шині.

UART, у свою чергу, застосовується для бездротового обміну даними через модуль Bluetooth наприклад, HC-05. Він забезпечує надійний, асинхронний зв'язок між мікроконтролером і мобільним додатком користувача. Формат обміну – прості текстові команди у вигляді рядків, що легко зчитуються як на стороні Arduino, так і в інтерфейсі користувача додатку.

Усі взаємозв'язки між модулями координуються в головному скетчі Arduino, який виконує роль центрального програмного хабу. Саме він відповідає за обробку вхідних повідомлень, синхронізацію часу, реагування на

події, оновлення дисплея та керування вихідними сигналами. Логіка написана таким чином, що передбачено обробку не лише коректних команд, але й обробку можливих помилок чи некоректних вхідних даних. Це підвищує надійність системи та забезпечує безпечну експлуатацію.

Обмін інформацією між додатком і пристроєм відбувається асинхронно, що дозволяє уникнути блокування виконання основного коду під час прийому чи передачі даних. Команди обробляються у спеціальному програмному буфері, реалізованому на базі об’єктів типу String або char. Після надходження команда парситься, розпізнається її тип і викликається відповідна функція-обробник. Наприклад, команда зчитується, розбивається на ключові елементи інструкція та параметри і записує новий час будильника в пам’ять. Аналогічно, запит STATUS повертає користувачу поточну інформацію про стан пристрою: температуру, залишок заряду, час тощо.

Такий підхід забезпечує високу інтерактивність пристрою, адже користувач не лише керує системою, а й отримує від неї зворотний зв’язок у зручному форматі. Це створює враження повного контролю над усіма аспектами роботи пристрою, роблячи взаємодію інтуїтивною та комфортною. Окрім того, використання асинхронної моделі дозволяє в подальшому масштабувати функціонал, не порушуючи загальної архітектури.

РОЗДІЛ 3
ПРАКТИЧНА РЕАЛІЗАЦІЯ СМАРТ-БУДИЛЬНИКА ТА МОБІЛЬНОГО ЗАСТОСУНКУ

3.1 Загальна архітектура проєкту

Смарт-будильник побудований на базі мікроконтролера Arduino Uno, який координує роботу всіх апаратних модулів та реалізує алгоритм керування пристроєм. Архітектура передбачає взаємодію між модулем реального часу DS1302, LCD-дисплеєм для виводу інформації, Bluetooth-модулем HC-05 для бездротового керування та п’єзодинаміком, що виконує функцію звукового оповіщення.

Взаємозв’язки між компонентами побудовані з урахуванням особливостей кожного з них: обмін даними між дисплеєм і мікроконтролером здійснюється по інтерфейсу I2C, тоді як HC-05 взаємодіє через послідовний UART. Така структура дозволила реалізувати паралельну роботу всіх компонентів без конфліктів.

Прототип архітектури зображено на рисунку 3.1, де було змодельовано за допомогою сервісу «Wokwi». З допомогою цієї схеми можна було протестувати роботу пристрою, це дозволило визначити необхідні компоненти та їх кількість, також перевіряти з’єднання [18].

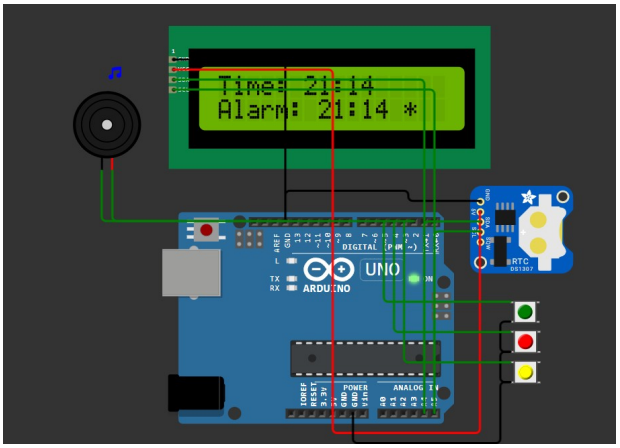

Рисунок 3.1 – Схема-прототип пристрою смарт-будильник

3.2 Апартне забезпечення та схема підключення

Живлення усієї системи здійснюється через USB або зовнішнє джерело 5В. До стабільності живлення приділено особливу увагу, оскільки перебої можуть спричинити збої в роботі модуля часу та втрату з’єднання з додатком.

Збірка пристрою була виконана на макетній платі, що надала можливість оперативно тестувати функціональність системи, вносити зміни до схеми та швидко усувати потенційні помилки без потреби пайки. Такий підхід особливо зручний на етапі розробки прототипу, коли важливо забезпечити гнучкість у конфігурації апаратних з’єднань.

Як основу обрано контролер Arduino Uno, який є одним із найпоширеніших і найзручніших мікроконтролерів для розробки навчальних і прототипних електронних пристроїв. Його вибір зумовлений декількома факторами: простота використання, велика кількість супровідної документації, активна спільнота, а також підтримка широкого спектра бібліотек для роботи з популярними датчиками та модулями. Крім того, Arduino Uno має достатню кількість цифрових та аналогових входів/виходів для підключення всіх необхідних компонентів без використання додаткових розширювачів портів.

LCD-дисплей підключено через шину I2C до контактів A4 – SDA і A5 – SCL. Модуль реального часу DS1302 підключений до контактів A0 – CLK, A1 – DAT, A2 – RST [19]. Це дозволило залишити інші входи/виходи вільними для додаткових модулів і сенсорів.

Реалізація функції відстеження часу здійснюється за допомогою модуля реального часу DS1302. Його підключено до аналогових контактів Arduino: лінія CLK – до A0, DAT – до A1, RST – до A2. Незважаючи на наявність внутрішнього таймера у самому мікроконтролері, застосування окремого RTC-модуля дозволяє забезпечити точність відліку часу навіть у випадку перезапуску або відключення живлення, оскільки модуль має власну батарею.

Комунікація між пристроєм та смартфоном реалізується через Bluetooth-модуль HC-05. Для підключення використано стандартні UART-лінії Arduino:

TX модуля з’єднано з RX Arduino, а RX модуля – з TX Arduino, із врахуванням потреби у відповідному зниженні напруги для уникнення пошкодження модуля.

Для подачі звукового сигналу використано бузер, який підключено до цифрового порту D13. Другий контакт бузера з’єднаний із загальним проводом GND. Цей компонент виконує функцію будильника, забезпечуючи подачу звукового сигналу з достатньою гучністю.

Уся система живиться через USB-порт Arduino або від зовнішнього джерела постійного струму 5 В, що може бути реалізоване через гніздо живлення плати. До живлення було висунуто підвищені вимоги стабільності, оскільки його перебої можуть призвести до некоректної роботи окремих модулів. Зокрема, нестабільна напруга живлення негативно впливає на модуль реального часу, що може втратити точність ходу, а також на роботу Bluetooth-модуля, який може втратити з’єднання із застосунком, що, в свою чергу, знижує надійність усього пристрою.

Усі компоненти були інтегровані у функціональну схему, яка дозволяє користувачу встановлювати час будильника, контролювати стан пристрою, а також отримувати візуальну та звукову інформацію без необхідності використання додаткового комп’ютера чи зовнішніх систем.

3.3 Програмне забезпечення

Програмна частина реалізована у середовищі Arduino IDE. Було використано відповідні бібліотеки: Wire.h та DS1302.h для роботи з годинником, LiquidCrystal_I2C.h для роботи з LCD-дисплеєм через I2C [20-21]. Bluetooth-модуль обробляє вхідні команди від мобільного додатку, розробленого в середовищі Android Studio.

Основні функції прошивки: отримання та відображення поточного часу на екрані, прийом команд на встановлення часу будильника на конкретний час, запуск звукового сигналу з бузера в заданий час. Після отримання сигналу з

модуля RTC про збіг поточного часу з встановленим будильником, вмикається п’єзодинамік.

Усі ключові частини програмного коду, що реалізують логіку керування в апаратній частині, наведено в додатку А.

3.4 Інтеграція компонентів

Після поетапного тестування кожного модуля окремо система була об’єднана в єдиний пристрій. Початкова перевірка здійснювалася у середовищі симуляції «Wokwi», що дозволило знайти та усунути помилки на рівні логіки. Після цього було здійснено тестування з фізичними модулями. У процесі інтеграції важливою була синхронізація роботи програмної частини з апаратними подіями.

Для перевірки точності та стабільності були проведені багатогодинні тести на реальному пристрої. Часова точність модуля DS1302 підтвердилася на практиці – протягом доби фіксував час на модулі та порівнював з реальним за допомогою serial monitor в Arduino IDE, зміщення не спостерігалося (лістинг 3.1). Передача команд з мобільного додатку з мінімальними затримками, або взагалі без затримок, звуковий сигнал спрацьовував вчасно із бажаною послідомністю та частотою.

Лістинг 3.1 – код для тестування часової точності RTC модуля

```
#include <DS1302.h>
// вказую піни для RST, DAT, CLK
const int RST_PIN = 16;    //A2
const int DAT_PIN = 15;    //A1
const int CLK_PIN = 14;    //A0
// створюю RTC
DS1302 rtc(RST_PIN, DAT_PIN, CLK_PIN);
void setup() {
  Serial.begin(9600);
```

```
  rtc.halt(false);   // запускає RTC
  rtc.writeProtect(false);
}
void loop() {   // кожні 10 секунд дата та час оновлюються
  Time t = rtc.getTime();
  Serial.print(«Час: »);   // вивід часу
  Serial.print(t.hour); Serial.print(«:»);
  Serial.print(t.min); Serial.print(«:»);
  Serial.println(t.sec);

  Serial.print(«Дата: »);  // вивід дати
  Serial.print(t.date); Serial.print(«.»);
  Serial.print(t.mon); Serial.print(«.»);
  Serial.println(t.year);
  delay(10000); //задання часу оновлення
}
```

кінець лістингу 3.1

3.5 Розробка мобільного додатку

Для керування смарт-будильником було створено мобільний додаток на платформі Android із використанням середовища розробки Android Studio. Основною метою додатку стало забезпечення користувача інтуїтивно зрозумілим інтерфейсом для керування пристроєм в особливості встановлення часу будильника, а також передача цієї інформації до мікроконтролера Arduino через Bluetooth-з’єднання [22].

Інтерфейс реалізовано за допомогою елементів TimePicker для вибору години й хвилин та кнопки для надсилання обраних данних. Було створено окремий Java-клас для роботи з Bluetooth, у якому ініціалізується з’єднання з модулем HC-05, перевіряється наявність дозволів і виконується передача повідомлень до пристрою. Команди формуються у вигляді форматованих рядків, які Arduino інтерпретує відповідно до запрограмованої логіки [23].

Наприклад, рядок A0815 означає встановлення будильника на 08:15. Додаток може надіслати команду B, будильник сприймає цю команду як – вказівку вимкнути будильник, при цьому час та дату на дисплеї все ще показуватиме. При ввімкнені будильника на екрані буде показано про це, за допомогою символу «*».

Інтерфейс добре підтримує як світлу тему пристрою так і темну. Також додаток показує статус підключення, також пристрій до якого здійснено підключення. При потребі вимкнути будильник наявна додаткова кнопка «Вимкнути будильник».

У процесі розробки також було реалізовано обробку можливих помилок: при втраті з’єднання або невдалому з’єднанні користувачу виводиться повідомлення з відповідною інформацією. У разі успішної синхронізації на екрані з’являється підтвердження, що час будильника змінено. Також користувач бачить статус підключення до пристрою (рис. 3.2). В разі, якщо пристрій не підключено то додаток буде показувати попереджувальні повідомлення про це, а також не дозволить встановити будильник (рис. 3.3).

Під час розробки виникали труднощі з дозволами використання функцій блютуз в особливості для пристроїв з новими версіями андроїд вище 12 версії. Проте у файлі «AndroidManifest.xml» я надав дозволи для використання базових функцій, розширених функцій та для нових версій андроід.

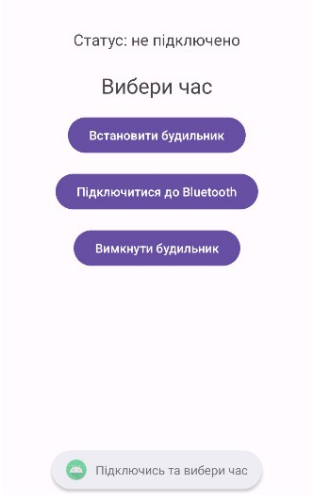

Рисунок 3.2 – Приклад статусу підключення до пристрою

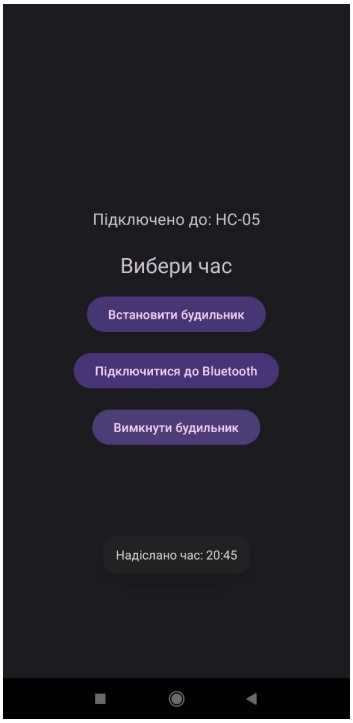

Рисунок 3.3 – Приклад зміни статусу підключеного додатку та сповіщення встановленого будильника

3.6 Тестування мобільного додатку

На етапі розробки системи особливу увагу було приділено тестуванню мобільного додатку, оскільки саме він виступає основним інтерфейсом взаємодії користувача з апаратною частиною будильника. Перевірка його працездатності, стабільності та сумісності проводилася в реальних умовах на фізичних пристроях із різними версіями операційної системи Android, що дозволило виявити потенційні помилки та забезпечити стабільну роботу в широкому діапазоні середовищ.

Застосунок тестувався на смартфонах з Android 10 та Android 13, зокрема на пристроях із прошивкою MIUI Xiaomi, яка характеризується посиленою політикою безпеки щодо доступу до системних ресурсів, зокрема Bluetooth-модуля. Це дозволило перевірити поведінку додатку в умовах обмеженого доступу до бездротових інтерфейсів, що є типовим для сучасних мобільних операційних систем.

Тестування охоплювало кілька ключових аспектів. Насамперед перевірялась стабільність з’єднання між додатком і Bluetooth-модулем HC-05 під час перемикання між програмами, блокування екрана та зміни контексту системи. Особливу увагу було приділено правильності передачі даних, а саме – здатності застосунку безпомилково надсилати команди у відповідному форматі, зокрема встановлення часу будильника та його вимкнення. Оцінювався також час реакції пристрою, тобто проміжок часу між відправкою команди та фактичною її обробкою мікроконтролером. Ще одним важливим критерієм стала коректна взаємодія з апаратною частиною пристрою – правильність активації звукового сигналу відповідно до заданих параметрів.

У процесі тестування було виявлено кілька характерних проблем. Наприклад, модуль HC-05 не з’являвся у списку доступних пристроїв на деяких смартфонах, що було пов’язано з відсутністю необхідних дозволів. Це стосувалося зокрема дозволів ACCESS_FINE_LOCATION та BLUETOOTH_CONNECT, які є обов’язковими для роботи Bluetooth на нових версіях Android. Внаслідок цього у додаток було додано перевірку дозволів під час запуску та механізм повторного запиту у разі відмови користувача.

Для перевірки правильності розпізнавання команд Arduino використовувалися серійний монітор та LCD-дисплей. Це дозволяло зіставити надіслані з додатку рядки з тими, що приймає мікроконтролер, і візуально переконатися у правильності зчитування та подальшої обробки даних. Було виявлено, що навіть незначні відхилення у форматі, зокрема зайві пробіли чи символи переносу рядка, можуть призводити до ігнорування команди. Тому код програми було доповнено механізмом суворої перевірки отриманих даних перед збереженням часу в оперативну пам’ять.

У межах тестування також моделювались ситуації з втратою зв’язку та повторним підключенням. Усі спроби з’єднання були успішними, що підтверджує надійність роботи з модулем HC-05 та правильність реалізації програмної логіки Bluetooth-обміну. У процесі моделювання спрацювання будильника підтверджено стабільну роботу системи: пристрій активував

звуковий сигнал у встановлений час, а після отримання відповідної команди – вимикав його без затримки.

У підсумку проведені тести підтвердили, що мобільний додаток повністю справляється зі своєю функцією, забезпечуючи стабільну та надійну взаємодію зі смарт-будильником. Система готова до використання в реальних умовах і відповідає вимогам до сучасних інтерактивних пристроїв.

3.7 Взаємодія додатку з апаратною частиною

Одним із ключових етапів реалізації проєкту стало забезпечення стабільної взаємодії між мікроконтролером Arduino та мобільним додатком через Bluetooth-з’єднання [24]. Задля цього було обрано модуль HC-05 – надійне та перевірене рішення для бездротового зв’язку, яке підтримує стандарт Bluetooth 2.0 SPP (Serial Port Profile), сумісний із більшістю Android-пристроїв. Інтеграція цієї технології дозволила перенести все управління будильником у мобільне середовище, тим самим значно підвищивши зручність користування пристроєм.

З’єднання між додатком і Arduino ініціюється вручну з інтерфейсу мобільного застосунку. Після успішного спарення HC-05 встановлює послідовне з’єднання, через яке передаються текстові команди. Комунікація базується на простому та зрозумілому протоколі: команда для встановлення будильника надсилається у форматі ТГГХХ\n, де ГГ – години, а ХХ – хвилини. Наприклад, рядок T0730\n означає встановлення будильника на 07:30 ранку. У разі потреби вимкнути активний сигнал будильника застосунок надсилає коротку команду B\n.

На стороні мікроконтролера реалізовано обробник вхідних команд, що аналізує отримані дані. Якщо повідомлення починається з літери T, наступні чотири символи перетворюються в числові значення для зберігання в змінних годин і хвилин. Ці значення порівнюються з часом, що надходить від модуля

реального часу DS1302, і у разі збігу активується звуковий сигнал. Команда B викликає негайне припинення сигналу та скидає активний стан тривоги.

Завдяки такій структурі обміну даними вдалося досягти високої надійності та передбачуваності роботи. Інтерфейс додатку мінімалістичний і водночас інтуїтивно зрозумілий: користувач має змогу встановити час спрацьовування будильника через зручний інструмент TimePicker, а також одним натисканням вимкнути сигнал.

Варто зазначити, що обробка Bluetooth-команд реалізована так, аби забезпечити толерантність до помилок – наприклад, у разі втрати з’єднання користувач отримає відповідне повідомлення. Такий підхід дозволяє уникнути збоїв та забезпечує надійну роботу пристрою навіть у нестабільних умовах зв’язку.

У підсумку, мобільний застосунок не лише виконує роль інтерфейсу взаємодії, але й фактично замінює традиційні апаратні кнопки. Це дозволяє зменшити кількість фізичних елементів керування на самому пристрої, спростити конструкцію та зробити використання будильника зручнішим і більш адаптованим до сучасних цифрових звичок користувачів.

3.8 Відповідність вимогам та оцінка результатів

Система повністю відповідає функціональним вимогам, сформованим на етапі проєктування. Досягнуто головну мету – створення смарт-будильника з можливістю встановлення часу сигналу через мобільний пристрій, виводу актуальної інформації на дисплей та звукового оповіщення у заданий час.

Водночас були виявлені деякі обмеження. Наприклад, при тривалому живленні від нестабільного джерела деякі модулі могли тимчасово втрачати працездатність, що вирішувалося додатковими конденсаторами на лініях живлення.

Проєкт виявився повністю працездатним. Усі компоненти стабільно взаємодіють, а користувач має зручний інтерфейс для керування. Цілі

кваліфікаційної роботи виконано, і створений пристрій може слугувати базою для подальшої розробки більш фукціональних моделей – з підтримкою Wi-Fi, кількох будильників чи сенсорного дисплея. Програмне забезпечення теж можна вдосконалити, додавши більше налаштувань. Також в подальшому можна замінити буззер на динамік, який зможе генерувати не тільки пищання, а й приємні звуки та музику.

ВИСНОВКИ

Здійснено побудову апаратної частини смарт-будильника на базі мікроконтролера Arduino Uno. Було використано модуль реального часу DS1302, Bluetooth-модуль HC-05, LCD-дисплей 1602 із інтерфейсом I2C та п’єзодинамік. Компоненти інтегровано з допомогою макетної плати, що дозволяє легко протестувати та змінювати конфігурацію пристрою. Перевірка показала стабільну роботу усіх модулів у спільній системі.

Розроблено програмне забезпечення для Arduino у середовищі Arduino IDE із використанням бібліотек для роботи з дисплеєм та модулем часу. Реалізовано основну логіку сигналу будильника, синхронізацію часу та обробку команд, що надходять через Bluetooth.

Виконано інтеграцію мікроконтролера з мобільним застосунком. Застосунок створено у середовищі Android Studio мовою Java. Основна функція – передача вибраного часу будильника з телефону на Arduino. Інтерфейс є простим і зручним для користувача.

Візуалізовано роботу будильника через LCD-дисплей, який у реальному часі показує поточну годину та дату, також містить позначку коли ввімкнено будильник.

Спроектовано та реалізовано загальну архітектуру смарт-будильника, яка включає логічну структуру взаємодії між модулями, а також електричну схему підключення компонентів. Архітектура забезпечує незалежну, але синхронізовану роботу всіх елементів системи. У якості середовища попереднього тестування було використано симулятор Wokwi, що дозволило перевірити правильність підключень та попередню функціональність пристрою ще до фізичної реалізації.

На основі проведених досліджень запропоновано кілька напрямів для подальшої модернізації пристрою. Серед них – інтеграція Wi-Fi модуля наприклад, ESP8266 для керування через інтернет, реалізація кількох будильників, розширення функціональності дисплея до сенсорного OLED, а

також додавання функцій моніторингу навколишнього середовища температури, вологості тощо. Це дозволить перетворити пристрій на багатофункціональний IoT-гаджет, що відповідає сучасним тенденціям розумної автоматизації побуту.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Internet of things market size, share, growth, trends, 2032. Fortune Business Insight Global Market Research Reports & Consulting. URL: https://surl.li/lenxau (дата звернення: 10.03.2025).

2. Blum J. Exploring arduino: tools and techniques for engineering wizardry. URL: https://surl.lu/djvwgt (дата звернення: 10.03.2025).

3. 7 top ides for android development in 2024. Lanars LLC. URL: https://lanars.com/blog/ide-for-android-development (дата звернення: 10.03.2025).

4. Philips. URL: https://surl.li/tmozhw (дата звернення: 10.04.2025).

5. Smart alarm clock for effective sleep health. URL: https://doi.org/10.1109/ICCI57424.2023.10112383 (дата звернення: 19.03.2025).

6. Arduino bluetooth basic tutorial. Hackster.io. URL: https://surl.li/febmxf (дата звернення: 19.03.2025).

7. WIZTECH інтернет магазин. URL: https://wiztech.com.ua/wp-content/uploads/2024/04/ESP32-wroom-32D.webp (дата звернення 10.04.2025).

8. Prom. URL: https://surl.li/xhqvil (дата звернення: 10.04.2025).

9. E-server. URL: https://surl.li/auvzwo (дата звернення: 10.04.2025).

10. Віктор Олександрович. Arduino Робототехніка для початківців. Що таке Arduino? Мікрокомп’ютер та його можливості, 2024. YouTube. URL: https://surl.li/vwinew (дата звернення: 23.03.2025).

11. Udara T. Selecting the best embedded board for your project: an in-depth guide. Medium. URL: https://surl.li/binuzx (дата звернення: 10.04.2025).

12. Arduino в Києві, Україна. URL: https://surl.li/rojfcp (дата звернення: 10.04.2025).

13. Arduino в Києві, Україна. URL: https://surl.li/jibvvx (дата звернення: 10.04.2025).

14. Prom. URL: https://surl.lu/wkplod (дата звернення: 10.04.2025).

15. HC-05/06 bluetooth modules DCC-EX model railroading documentation. DCC-EX Model Railroading DCC-EX Model Railroading documentation. URL: https://surl.lu/ennfcd (дата звернення: 10.04.2025).

16. Rozetka. URL: https://surl.li/fegulr (дата звернення: 10.04.2025).

17. Arduino в Києві, Україна. URL: https://surl.li/qpgiaa (дата звернення: 10.04.2025).

18. Arduino Based Digital Alarm Clock - Wokwi Arduino Simulator. Wokwi World’s most advanced Arduino Simulator. URL: https://surl.gd/pxnjbd (дата звернення: 17.04.2025).

19. MG ProjectHub. Arduino bluetooth basic, 2016. YouTube. URL: https://surl.lu/fiaezr (дата звернення: 18.04.2025).

20. DS1302 Rinky-Dink Electronics. Rinky-Dink Electronics. URL: http://www.rinkydinkelectronics.com/library.php?id=5 (дата звернення: 23.04.2025).

21. Which LiquidCrystal_I2C library. Arduino Forum. URL: https://forum.arduino.cc/t/which-liquidcrystal-i2c-library/1017770 (дата звернення: 23.04.2025).

22. Android Studio що це таке і для чого потрібна. AndroidAyuda. URL: https://surl.li/hpntta (дата звернення: 30.04.2025).

23. Віталій Мартинюк. Як зробити android додаток, 2023. YouTube. URL: https://surl.li/hvgull (дата звернення: 30.04.2025).

24. Керуємо arduino зі смартфона android за допомогою bluetooth. MikroTik-NTUU-KPI. URL: https://surl.lu/qdlqez (дата звернення: 10.05.2025).

ДОДАТКИ

Додаток А

Код прошивки для Arduino

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <SoftwareSerial.h>
#include <DS1302.h>

#define BUZZER_PIN 13
#define BUZZ_INTERVAL 500  // мілісекунди

SoftwareSerial BT(10, 11); // RX, TX

// LCD на I2C — адреса 0x27, 16 символів × 2 рядки
LiquidCrystal_I2C lcd(0x27, 16, 2);

// DS1302
#define RST_PIN 16  // CE
#define DAT_PIN 15  // I/O
#define CLK_PIN 14  // SCLK

DS1302 rtc(RST_PIN, DAT_PIN, CLK_PIN);

int alarmHour = -1;
int alarmMinute = -1;
bool isBuzzing = false;
bool buzzerState = false;
unsigned long lastBuzzToggle = 0;

void setup() {
  pinMode(BUZZER_PIN, OUTPUT);
  digitalWrite(BUZZER_PIN, LOW);

  Serial.begin(9600);
  BT.begin(9600);

  rtc.halt(false);
  rtc.writeProtect(false);

  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print(«Smart Alarm Ready»);

  Serial.println(«Смарт-будильник готовий до роботи»);
  delay(2000);
  lcd.clear();
}

void loop() {
  if (BT.available()) {
```

```
    String command = BT.readStringUntil('\n');
    command.trim();

    Serial.print(«Команда: »);
    Serial.println(command);

    if (command == «A») {
      isBuzzing = true;
      Serial.println(«Бузер увімкнено вручну»);
    } else if (command == «B» || command == «off») {
      digitalWrite(BUZZER_PIN, LOW);
      isBuzzing = false;
      buzzerState = false;
      alarmHour = -1;
      alarmMinute = -1;
      Serial.println(«Бузер вимкнено»);
    } else if (command.startsWith(«T») && command.length() == 5) {
      alarmHour = command.substring(1, 3).toInt();
      alarmMinute = command.substring(3, 5).toInt();
      Serial.print(«Будильник на: »);
      Serial.print(alarmHour);
      Serial.print(«:»);
      Serial.println(alarmMinute);
    }
  }

  Time now = rtc.time();

  // Показуємо час на першому рядку LCD
  lcd.setCursor(0, 0);
  lcd.print(«Time: »);
  printTimeLCD(now);

  // Якщо будильник встановлено — додаємо зірочку в кут
  if (alarmHour != -1) {
    lcd.setCursor(15, 0);  // останній символ першого рядка
    lcd.print('*');
  } else {
    lcd.setCursor(15, 0);
    lcd.print(' '); // стираємо зірочку
  }

  // Показуємо дату на другому рядку LCD
  lcd.setCursor(0, 1);
  printDateLCD(now);

  if (alarmHour != -1 && now.hr == alarmHour && now.min == alarmMinute &&
!isBuzzing) {
    isBuzzing = true;
    lastBuzzToggle = millis();
    Serial.println(«Час будильника! Увімкнено сигнал»);
  }
```

```
  if (isBuzzing) {
    if (millis() - lastBuzzToggle >= BUZZ_INTERVAL) {
      buzzerState = !buzzerState;
      digitalWrite(BUZZER_PIN, buzzerState ? HIGH : LOW);
      lastBuzzToggle = millis();
    }
  }

  delay(500);
}

void printTimeLCD(Time t) {
  if (t.hr < 10) lcd.print('0');
  lcd.print(t.hr);
  lcd.print(':');
  if (t.min < 10) lcd.print('0');
  lcd.print(t.min);
  lcd.print(':');
  if (t.sec < 10) lcd.print('0');
  lcd.print(t.sec);
  lcd.print(« »);
}

void printDateLCD(Time t) {
  if (t.date < 10) lcd.print('0');
  lcd.print(t.date);
  lcd.print('.');
  if (t.mon < 10) lcd.print('0');
  lcd.print(t.mon);
  lcd.print('.');
  lcd.print(2000 + t.yr);
}
```

Додаток В

Основний код андроід додатку

```
package com.example.smart;

import android.app.TimePickerDialog;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.os.Bundle;
import android.util.Log;
import android.widget.Button;
import android.widget.TextView;
import android.widget.TimePicker;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;

import java.io.IOException;
import java.io.OutputStream;
import java.util.Calendar;
import java.util.Set;
import java.util.UUID;

public class MainActivity extends AppCompatActivity {

    private static final String TAG = «MainActivity»;
    private static final String DEVICE_NAME = «HC-05»;  // Назва модуля
    private static final UUID SPP_UUID = UUID.fromString(«00001101-0000-
1000-8000-00805F9B34FB»);

    private BluetoothAdapter bluetoothAdapter;
    private BluetoothSocket bluetoothSocket;
    private OutputStream outputStream;

    private Button connectBtn, setAlarmBtn, offAlarmBtn;
    private TextView statusText;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        connectBtn = findViewById(R.id.connectBtn);
        setAlarmBtn = findViewById(R.id.setAlarmBtn);
        offAlarmBtn = findViewById(R.id.offAlarmBtn);
        statusText = findViewById(R.id.statusText);

        bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

        connectBtn.setOnClickListener(v -> startClassicScanAndConnect());
        setAlarmBtn.setOnClickListener(v -> sendTimeToArduinoClassic());
```

```
        offAlarmBtn.setOnClickListener(v -> sendOffCommandClassic());
    }

    private void startClassicScanAndConnect() {
        if (bluetoothAdapter == null) {
            Toast.makeText(this, «Bluetooth не підтримується»,
Toast.LENGTH_SHORT).show();
            return;
        }

        if (!bluetoothAdapter.isEnabled()) {
            bluetoothAdapter.enable();  // краще відкрити діалог, але для
простоти вмикаємо напряму
        }

        Set<BluetoothDevice> pairedDevices =
bluetoothAdapter.getBondedDevices();
        if (pairedDevices != null) {
            for (BluetoothDevice device : pairedDevices) {
                Log.d(TAG, «Знайдено пристрій: » + device.getName());
                if (DEVICE_NAME.equals(device.getName())) {
                    connectToDeviceClassic(device);
                    return;
                }
            }
        }

        statusText.setText(«Пристрій не знайдено»);
        Toast.makeText(this, «Пристрій не знайдено серед спарених»,
Toast.LENGTH_SHORT).show();
    }

    private void connectToDeviceClassic(BluetoothDevice device) {
        try {
            bluetoothSocket =
device.createRfcommSocketToServiceRecord(SPP_UUID);
            bluetoothSocket.connect();
            outputStream = bluetoothSocket.getOutputStream();
            statusText.setText(«Підключено до: HC-05»);
            Toast.makeText(this, «Підключено», Toast.LENGTH_SHORT).show();
        } catch (IOException e) {
            Log.e(TAG, «Помилка підключення», e);
            statusText.setText(«Помилка підключення»);
            Toast.makeText(this, «Не вдалося підключитися»,
Toast.LENGTH_SHORT).show();
        }
    }
    private void sendTimeToArduinoClassic() {
        if (outputStream == null) {
            Toast.makeText(this, «Немає з'єднання»,
Toast.LENGTH_SHORT).show();
            return;
        }
```

```
        Calendar calendar = Calendar.getInstance();
        int hour = calendar.get(Calendar.HOUR_OF_DAY);
        int minute = calendar.get(Calendar.MINUTE);
        TimePickerDialog timePickerDialog = new TimePickerDialog(this,
(TimePicker view, int selectedHour, int selectedMinute) -> {
            String command = String.format(«T%02d%02d\n», selectedHour,
selectedMinute);
            try {
                outputStream.write(command.getBytes());
                Toast.makeText(this, «Надіслано час: « + selectedHour + «:»
+ selectedMinute, Toast.LENGTH_SHORT).show();
            } catch (IOException e) {
                Log.e(TAG, «Помилка надсилання», e);
                Toast.makeText(this, «Помилка надсилання»,
Toast.LENGTH_SHORT).show();
            }
        }, hour, minute, true);

        timePickerDialog.show();
    }

    private void sendOffCommandClassic() {
        if (outputStream == null) {
            Toast.makeText(this, «Немає з'єднання»,
Toast.LENGTH_SHORT).show();
            return;
        }

        try {
            outputStream.write(«B\n».getBytes());
            Toast.makeText(this, «Будильник вимкнено»,
Toast.LENGTH_SHORT).show();
        } catch (IOException e) {
            Log.e(TAG, «Помилка надсилання», e);
            Toast.makeText(this, «Помилка надсилання»,
Toast.LENGTH_SHORT).show();
        }
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        try {
            if (outputStream != null) outputStream.close();
            if (bluetoothSocket != null) bluetoothSocket.close();
        } catch (IOException e) {
            Log.e(TAG, «Помилка закриття з'єднання», e);
        }
    }
}
```