ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ І. І. МЕЧНИКОВА

Факультет математики, фізики та інформаційних технологій

Кафедра фізики та астрономії

Кваліфікаційна робота

на здобуття ступеня вищої освіти «бакалавр»

«Автоматизація обліку відвідування занять студентами за допомогою мікроконтролера»

“ Automation of student class attendance accounting using a microcontroller ”

Виконав: здобувач денної форми навчання
спеціальності 151-Автоматизація та комп’ютерно-інтегровані технології
Освітня програма Комп’ютерна обробка та аналіз даних
Ємельянов Євген Михайлович

Керівник к.ф.-м.н.,доц.Сидоров О.Є._____
(підпис)
Рецензент ст.викл.Стукалов С.А.

Рекомендовано до захисту:
протокол засідання кафедри
фізики та астрономії
№ ___ від ____.____. 20___ р.

Завідувач кафедри
_____ Володимир ГОЦУЛЬСЬКИЙ
(підпис)

Захищено на засіданні ЕК № _____
протокол № __від ____.____.20___ р.
Оцінка_____/_____/_____
(за національною шкалою/шкалою ECTS/ бали)

Голова ЕК
_____ Володимир ГОЦУЛЬСЬКИЙ
(підпис)

Одеса 2025

ЗМІСТ

ВСТУП

У сучасний етап розвитку технологій дуже багато речей автоматизовано для швидкого збору інформації та для подальшого аналізу. Однією з проблем у освітньому процесі – облік відвідування занять студентами. Традиційні ручні методи фіксації мають досить суттєвих недоліків, серед яких значні витрати часу викладачів, висока ймовірність помилок при фіксації, а також складність провести подальші дослідження та аналіз накопиченої інформації. Облік відвідування занять студентами потребує автоматизації для швидкого збору інформації та подальшого аналізу. Вирішення цієї актуальної проблеми - це розробка автоматизованої системи обліку на базі сучасних мікроконтролерних технологій. Система повинна бути досить швидкою та ефективною для точної фіксації відвідуваності, особливо в закладах де навчається велика кількість студентів. Сучасні методи обміну даними дозволяють створити таку систему з можливістю подальшого вдосконалення.

Метою даної роботи є розробка комплексного рішення для автоматизації обліку відвідування. Автоматизована система є комбінацією мікронтролерних технологій, звичайних комп’ютерних технологій, програмного забезпечення для обробки даних та інтерфейс для зручного адміністрування. Основні завдання цієї роботи включають аналіз існуючих технологічних рішень у сфері автоматизації обліку відвідування, обґрунтування вибору апаратних засобів та модулів для мікроконтролера, реалізація апаратної частини системи, створення інтегрованої бази даних та програмного забезпечення мікроконтролера. Особливу увагу слід звернути на забезпечення надійності системи, зручності її експлуатації та можливості подальшого масштабування.

1. СИСТЕМИ ОБЛІКУ ВІДВІДУВАННЯ

Сучасні технології пропонують різноманітні засоби для автоматизації обліку відвідування, кожне з яких має свої переваги та недоліки. Найбільш поширеними є RFID-системи, системи на основі біометричних даних та системи на основі QR-кодів.

Самі безпечні та ефективні системи – біометричні системи, такі як сканери відбитків пальців або системи розпізнавання обличчя. Проте, їх впровадження та обслуговування вимагає значно великих витрат – середня вартість одного робочого місця становить близько $500-800 [1]. Також для їх встановлення можуть викликати юридичні складнощі.

Рис. 1. Біометричні засоби ідентифікації людини.

Самі доступні системи за вартістю – системи на основі QR-кодів (вартість впровадження близько $50-100 на точку доступу). Але вони мають серйозні недоліки у сфері безпеки. Дослідження Onecollab (2023) [2] показало, що QR-коди можуть бути легко підроблені або копійовані, що робить їх менш прийнятними для систем обліку відвідування.

Досить широке застосування отримали системи, котрі працюють на RFID ттехнології. Технологія RFID (Radio Frequency Identification) – це метод автоматичної ідентифікації об’єктів, котрий використовує радіохвилі та електромагнітну індукцію для передачі даних між RFID-міткою та зчитувачем. Вона використовується в облікових системах університетів всього світу, таких як MIT та Stanford. Ці системи працюють на основі радіочастотній ідентифікації для автоматичного зчитування даних з карток студентів. За даними дослідження RFID Journal (2023) [3], такі системи працюють досить швидко (обробка одного студента займає менше 0,3 секунди). Основною перевагою RFID є можливість одночасної обробки декількох карток, що дає змогу значно прискорити ефективність роботи системи у майбутньому.

Через ці переваги була обрана RFID технологія. Система зібрана на базі Arduino та RFID-модуля PN532 поєднує переваги цих технологій, мінімізючи їх недоліки. Крім того, вартість одного комплекту обладнання не перевищує 100$, що дешевше у 5-8 разів системи на основі біометричних даних, але ця система досить надійна та також не дорожче систем на основі QR-кодів. Важливою перевагою є можливість інтеграції з популярними системами управління навчанням (LMS), такими як Moodle, через API або експорт даних у стандартних форматах.

За даними дослідження EDUCASE (2024) [4], понад 65% сучасних університетів використовують комбіновані системи контролю доступу з використанням RFID-технологій. Розроблена система відповідає міровим трендам, також мінімізуючи витрати коштів у порівнянні з комерційними аналогами, пропонуючи досить великі можливості та значну безпеку. Головна перевага RFID-технології полягає у відсутності прямого контакту або лінії візуального зв’язку між пристроями. Існують два типи міток: пасивні та активні. Пасивні мітки не мають зовнішнього електричного живлення, але отримують його з зчитувача, котрий генерує електромагнітне поле на певній частоті (13.56 МГц для PN532). З отриманої енергії мітка модулює поле,

передаючи назад свої дані. Перевага пасивних міток – їх низька вартість та простота. Активні мітки навпаки, мають власне живлення й одразу передають сигнал на певній частоті. Перевага активних меток в відстані дії (до 100м).

RFID-система складається з таких компонентів як: транспондер (RFID-мітка) (рис.2), зчитувач (RFID-ридер) та антенна система. Транспондер містить в собі унікальний індетнифікатор (UID), код для аунтетіфікації та блоки пам’яті для запису різної інформації у вигляді 16-річного машиного кода. Зчитувач генерує електромагнітне поле, детектує да декодує сигнал від мітки. Антенна система відповідає за ефективність зв’язку, може визначити дальність зчитування.

Рис.2. Типи безконтактних карток.

Окремо слід розглянути безпеку RFID-технології. Сучасні системи, особливо системи доступу та обліку відвідування, мають кілька рівнів захисту даних, які розглянемо детальніше.

Одним з головоних елементів безепеки в RFID-технології [5] це унікальні ідентифікатори (UID). В кожной мітки міститься заводський ідентифікатор, який фізично «вшитий» у чип під час виробництва. Він не підлагає зміні. Також копіювання такого ідентифікатора вимагає спеціалізованого обладнання та знань, це забезпечує базовий рівень захисту від підробки танспондера. Але слід зазначити, що в деяких простих мітках UID може передаватися у відкритому вигляді, що створює потенційну вразливість до атак типу «replay attack».

Також один з елементів безпеки являються криптографічні алгоритми. Цей алгоритм забезпечує шифрування переданих даних що досить сильно ускладнює їх перехоплення та розшифрування без відповідного ключа. AES-128 (Advanced Encryption Standard) алгоритм, який використовує ключ довжиною 128 біт, що забезпечує високий рівень безпеки. AES-128 реалізується у більш досконалих мітках стандарту Mifare DESFire або Ultralight C. Проте базові мітки, наприклад, Mifare Classic 1K, які є бюджетним рішенням для систем, мають слабший криптографічний захист (алгоритм Crypto-1), який може бути вразливий до взлому за допомогою спеціалізованого ПЗ.

Критично важливим елементом системи безпеки є процес аутентифікації перед операціями зчитування/запису. Принцип цього алгоритму такий: перед тим як поділитися даними, зчитувач та мітка обмінюються випадковими числами. Якщо після виконання криптографічних дій з цими числами ключі аутентифікації співпадають, то аутентифікація пройшла успішно та відкривається доступ до запису/зчитування даних. Стандартний ключ має такий вигляд: **0xFF 0xFF 0xFF 0xFF 0xFF 0xFF.** Але в умовах експлуатації рекомендується використовувати унікальні ключі для кожного примірника картки.

Додаткові механізми захисту:

1. Контроль цілісності даних за допомогою CRC (Cyclic Redundancy Check)

2. Обмеження кількості спроб аутентифікації
3. Можливість блокування міток після виявлення підрозрілих спроб доступу або взлому
4. Функція «антиколізії», яка запобігає одночасному зчитуванню кількох міток.

Слід зазначити, що система безпеки RFID-технології буде справно працювати тільки тоді, коли дані також добре захищені в інших елементах загальної системи, передачі даних до бази даних та в самій базі даних.

2. АПАРАТНА РЕАЛІЗАЦІЯ

2.1. Вибір компонентів

Мікроконтролер Arduino UNO (рис. 3.1) був обраний як основа системи з кількох ключових причин. Це максимально універсальна основа, на яку є безліч різноманітних рішень, через свою простоту та велику спільноту розробників. Згідно з офіційною документацією Arduino (2023) [5] плата оснащена мікроконтролером Atmega328P з тактовою частотою 16 МГц, що цілком достатньо для нашої задачі. Arduino UNO має 14 цифрових входів/виходів, 6 аналогових входів, що дозволяє підключити додаткові модулі та датчики при необхідності.

Рис. 3.1. Мікроконтролер Arduino UNO.

NFC-модуль PN532 (рис. 3.2) був обраний через максимальну сумісність з Arduino та підтримку різних протоколів комунікації. Згідно з офіційними даними [6], цей модуль підтримує:

1. Роботу з картками типу Mifare Classic 1K
2. Діапазон роботи до 5 см
3. Швидкість передачі даних до 424 кбіт/с

Рис. 3.2. NFC-модуль PN532.

Найголовнішою перевагою є підтримка інтерфейсу I2C, який дозволяє підключати модуль лише двома провідниками (SDA та SCL), що спрощує монтаж, та дозволяє підключити одночасно декілька таких модулів.

Додаткові компоненти системи:

1. Світлодіоди для візуальної індикації стану системи.
2. Релейний модуль виконує декілька функцій, а саме: імітує роботу турнікету при успішній ідентифікації, може керувати зовнішніми пристроями.
3. Джерело живлення – для стаціонарного використання рекомендовано стабілізований блок живлення 5V/2A, для мобільних рішень можна використати Power Bank з USB-виходом.

Звернемо увагу на те, що всі ці компоненти широко поширені на ринку, що спрощує їхнє придбання та заміну у разі необхідності, також вони всі є стандартом, немає ніяких проблем з сумісністю.

2.2 Схема підключення

Для реалізації апаратної частини системи потрібно з’єднати всі компоненти в одну схему. Її основою виступає мікроконтролер Arduino UNO, до якої підключається NFC/RFID модуль PN532, а також додаткові периферійні пристрої.

Головні елементи схеми підключення:

1. Модуль PN532 підключається за допомогою інтерфейсу I2C, що дозволяє значно спростити схему та зменшити кількість необхідних з’єднань. Пін TXD модуля PN532 підключається до аналового входу A4 на Arduino UNO. Цей провідник передає дані між контролером та модулем. Пін RXD підключається до аналового виводу A5 на Arduino. Ця лінія відповідає за синхронізацію передачі даних. Живлення модуля (VCC) підключається до відповідного виводу на Arduino. Важливо врахувати, що PN532 вимагає живлення 3.3V, а не 5V.
2. Також підключається релейний модуль, який імітує роботу турнікету або керує електромагнітним замком. Керуючий вхід реле S підключається до цифрового виходу 7 на Arduino. Також керуюча частина реле вимагає живлення 3.3V, куди и підключається відповідний провідник. Живлення силової частини реле підключається до окремого джерела живлення.
3. Підключається світлодіодна система індикації. Жовтий світлодіод підключається до виходу 2, він сигналізує о готовності запису інформації на карту. Білий світлодіод, підключений до вихіду 8, попереджає про помилку запису та/або зчитування інформації. До реле під’єднані ще два світлодіода, червоний, що вказує на те, що турнікет закритий, та зелений, котрий вказує на успішну ідентифікацію та відкриття турнікету. Всі світлодіоди підключаються через обмежувальний резистор (220-470 Ом).

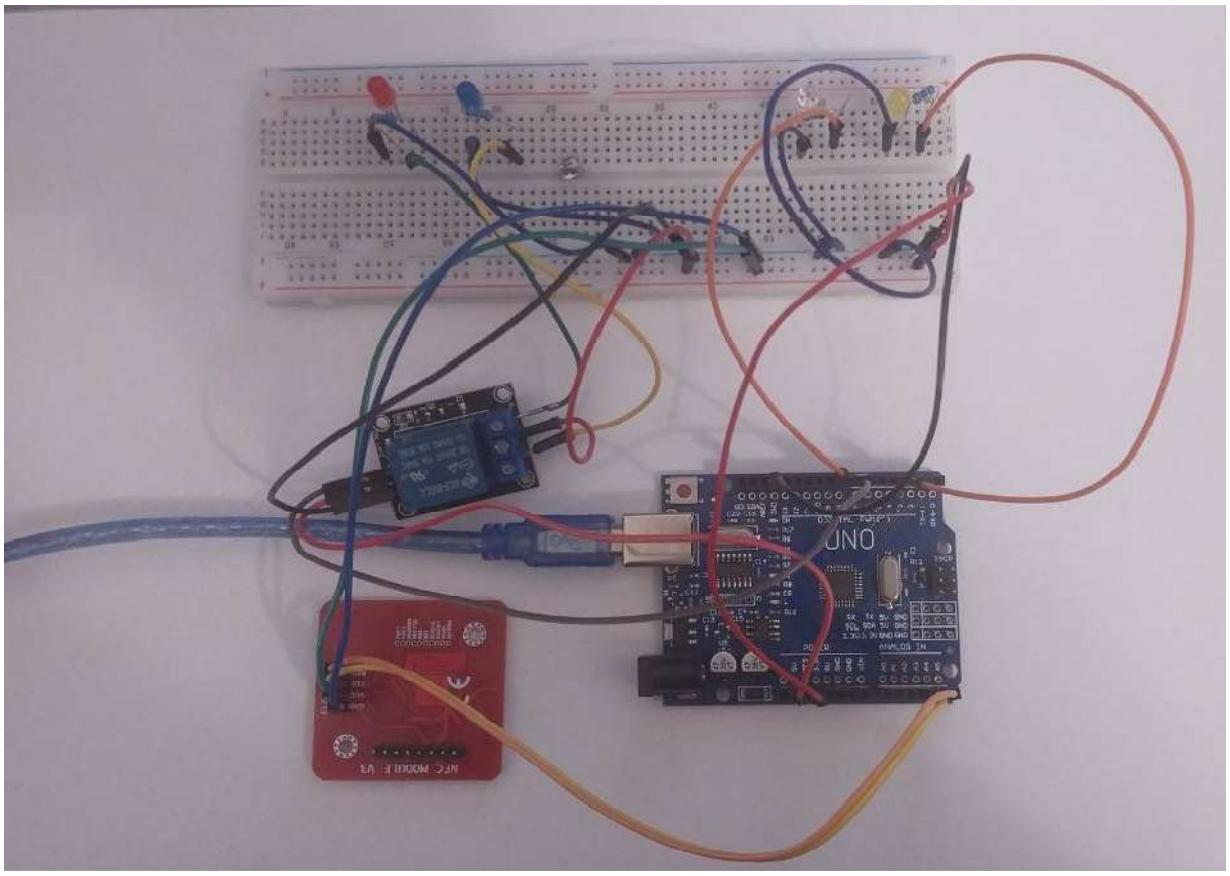

Рис. 4.1. Фото схеми підключення.

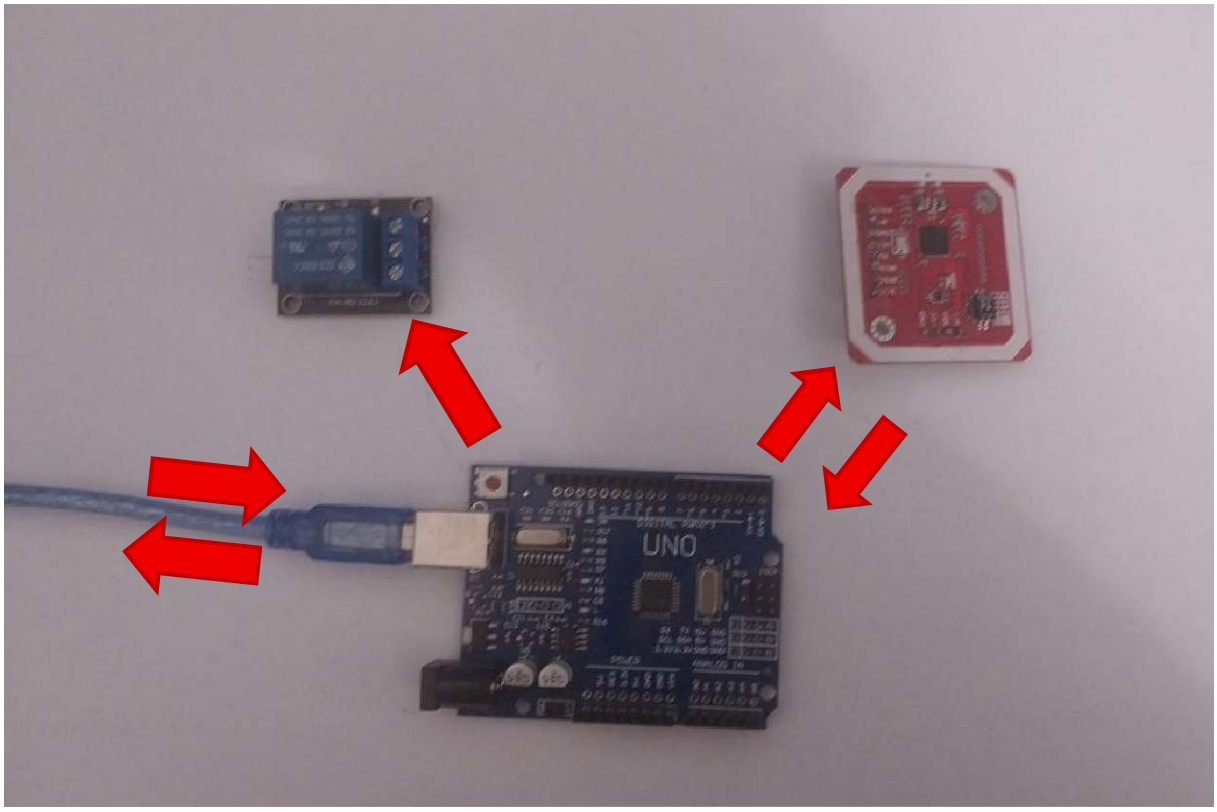

Рис4.3. Фото схеми без провідників. З потоками даних.

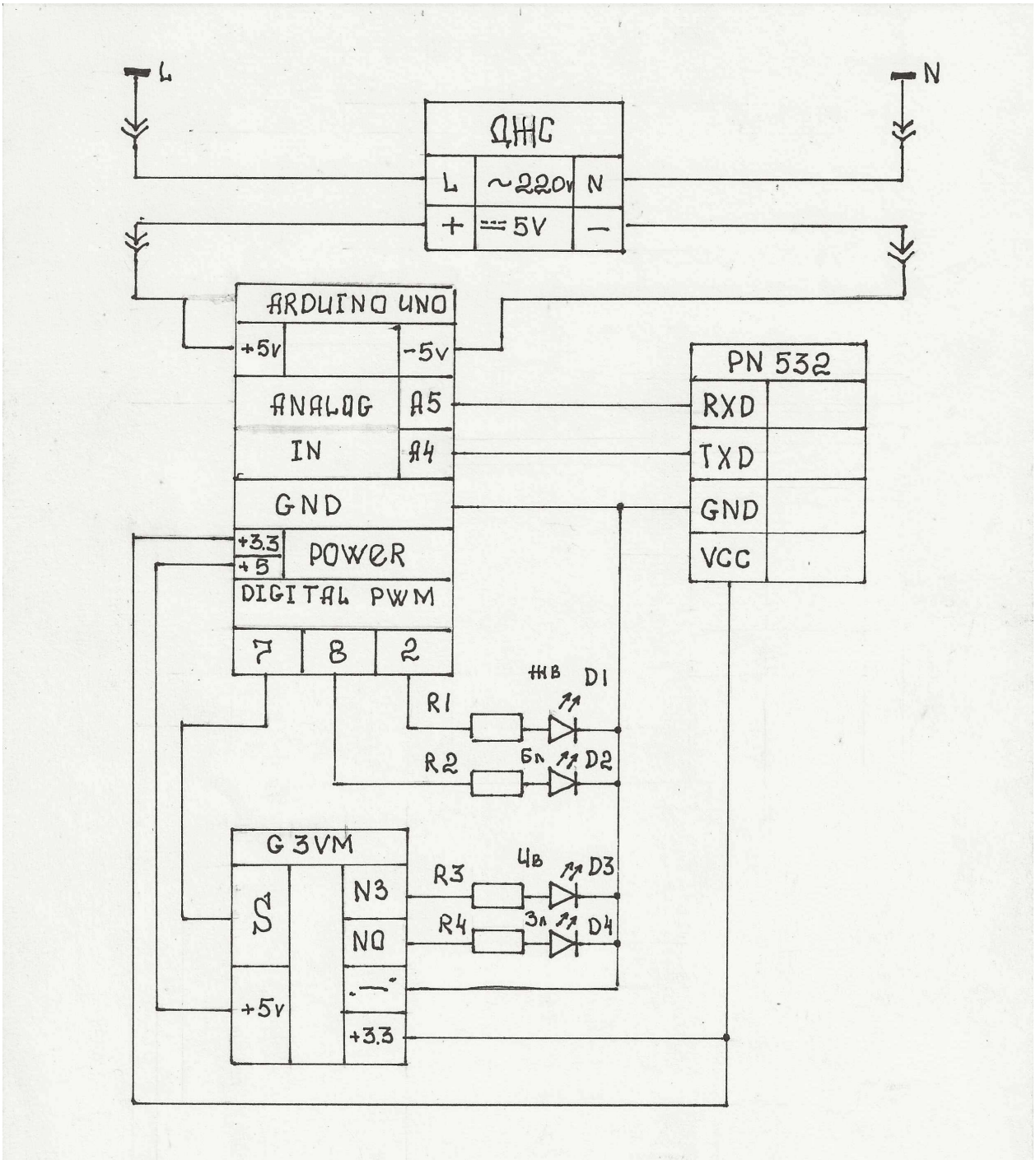

Рис. 4.2. Схема підключення.

На рис.4.2. ДЖС – джерело живлення стабілізаційне. G3VM – реле-перемикач. Жв – жовтий, Бл – білий, Чв – червоний, Зл – зелений.

Схема підключення виявилася надійною, але через універсальність Arduino цю схему можна швидко змінити, якщо це потрібно для подальшого монтажу. Для стабільної роботи системи рекомендується джерело живлення 5V з струмом не менше 1А.

3. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

Програмне забезпечення складається з декількох блоків програмного коду. Воно включає у собі програму написану на мові C++ для Arduino, базу даних Microsoft Access з написаними скриптами та PowerShell-скрипти для зв’язку між базою даних та Arduino.

3.1 Реалізація програмного коду для Arduino

Програмна частина системи розроблена на мові C++ для платформи Arduino IDE та складаэться з декількох ключових функцій, які забезпечують повний цикл роботи з RFID-картками, а саме запис та зчитування даних на картки.

Архітектура програми:

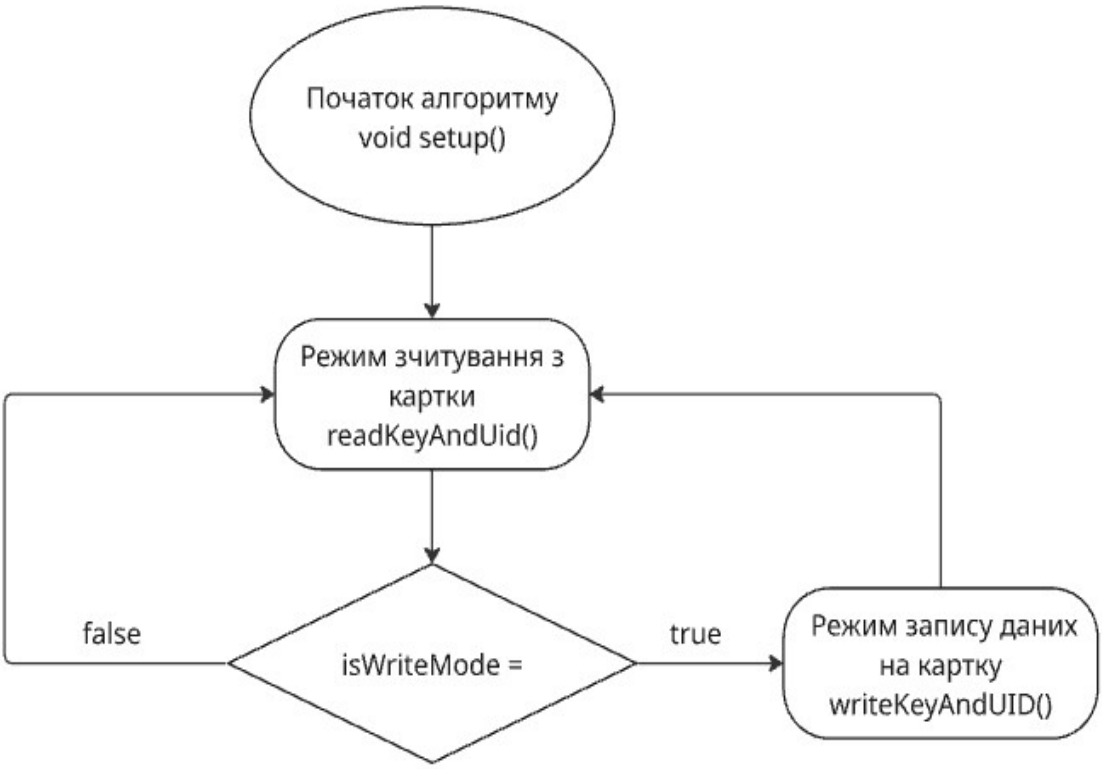

Рис. 5.1. Загальна блок-схема програми.

Як позначено на рис. 2, після запуску програмия Arduino одразу входить у режим зчитування даних з картки, тобто з початку виконування програми isWriteMode = false, тоді виконується блок програми

readKeyAndUID(), а саме зчитує 16 символьний ключ та UID. Але користувач може перемкнути систему у режим запису даних для реєстрації нових студентів, тобто якщо isWriteMode = true, виконується блок програми writeKeyAndUID(). Ці два режима роботи розглянемо детальніше.

3.1.1. Архітектура readKeyAndUID()

Першочергове програма зчитує фізичний UID карти (рис. 5.2) та виводить його у консоль. Після чого починає аунтетифікацію 4 та 5 блока пам’яті за допомогою ключа доступу. Якщо аунтетифікація пройшла успішно, то зчитуються дані з блоків пам’яті, а саме з 4 блоку 16-символьний ключ та з 5 блоку студентський UID, після чого отримаємо ось такий результат у консолі:

Encrypted Key (ASCII): (саме тут розташовується ключ)

String stored UID = (UID студентської карти).

Після чого йде перевірка на відкриття умовного турнікету. В програмі є масив студентських UID (allowedUIDs[i]), яким дозволен доступ до аудіторії. Якщо зчитаний UID співпадає з одним елементом масива, то виконується перемикання реле та вмикається зелений світлодіод, який сигналізує об успішності ідентифікації. Всі інші UID не будуть давати доступу до аудіторії, турнікет залишиться закритим.

Якщо один з процесів був прерван або, через деякі обставини, не виконався до кінця, то білий світлодіод буде сигналізувати о том, що виникла помилка. Також в консоль буде трансліруватися саме який з процесів не пройшов до кінця.

Важливо зазначити, що червоний світлодіод буде ввімкнений з самого початку виконання алгоритму, сигналізуючи о том, що турнікет закритий та доступ до аудіторії заборонен.

3.1.2. Архітектура writeKeyAndUID().

У користувача є можливість перемкнути режим зчитування на режим запису даних. Якщо користувач натисне кнопку «1», то система переключиться у режим запису, видав у консоль:

Switched to WRITE mode.

Важливо зазначити, що червоний світлодіод буде ввімкнений з самого початку виконання алгоритму, сигналізуючи о том, що турнікет закритий та доступ до аудіторії заборонен.

Архітектура блока програму вибору та запису ключа на RFID-карту writeKeyAndUID().

У користувача є можливість перемкнути режим зчитування на режим запису даних. Якщо користувач натисне кнопку «1» то система переключиться у режим запису (рис.5.3), сигналізує у консоль:

Switched to WRITE mode.

Після чого буде чекати вводу 16-символьного ідентифікаціонного ключа студента. При введенні 16-символьного ключа система перевіряє кількість символів, якщо кількість символів дорівнює 16, то система сигналізує о готовності запису даних на карту, вмикається жовтий світлодіод та в консоль виводиться повідомлення:

Received new key for writing. Please tap the card.

Після чого визивається блок програми writeKeyAndUID(), де починається процес запису даних на RFID-карту студента. А саме знову зчитується фізичний UID картки, після чого виводиться в консоль повідомлення:

Card detected. Authenticating...

Де починається процес аутентіфікації для отримання доступу до 4 та 5 блоку пам’яті для запису даних в них. Якщо аутентифікація пройшла успішно виводиться повідомлення:

Authentication successful. Writing key and UID...

Далі записується 16-символьний ключ, котрий ми ввели раніше, в 4 блок пам’яті:

```
if (nfc.mifareclassic_WriteDataBlock(4, keyData)) {
    Serial.println("Key written.");
```

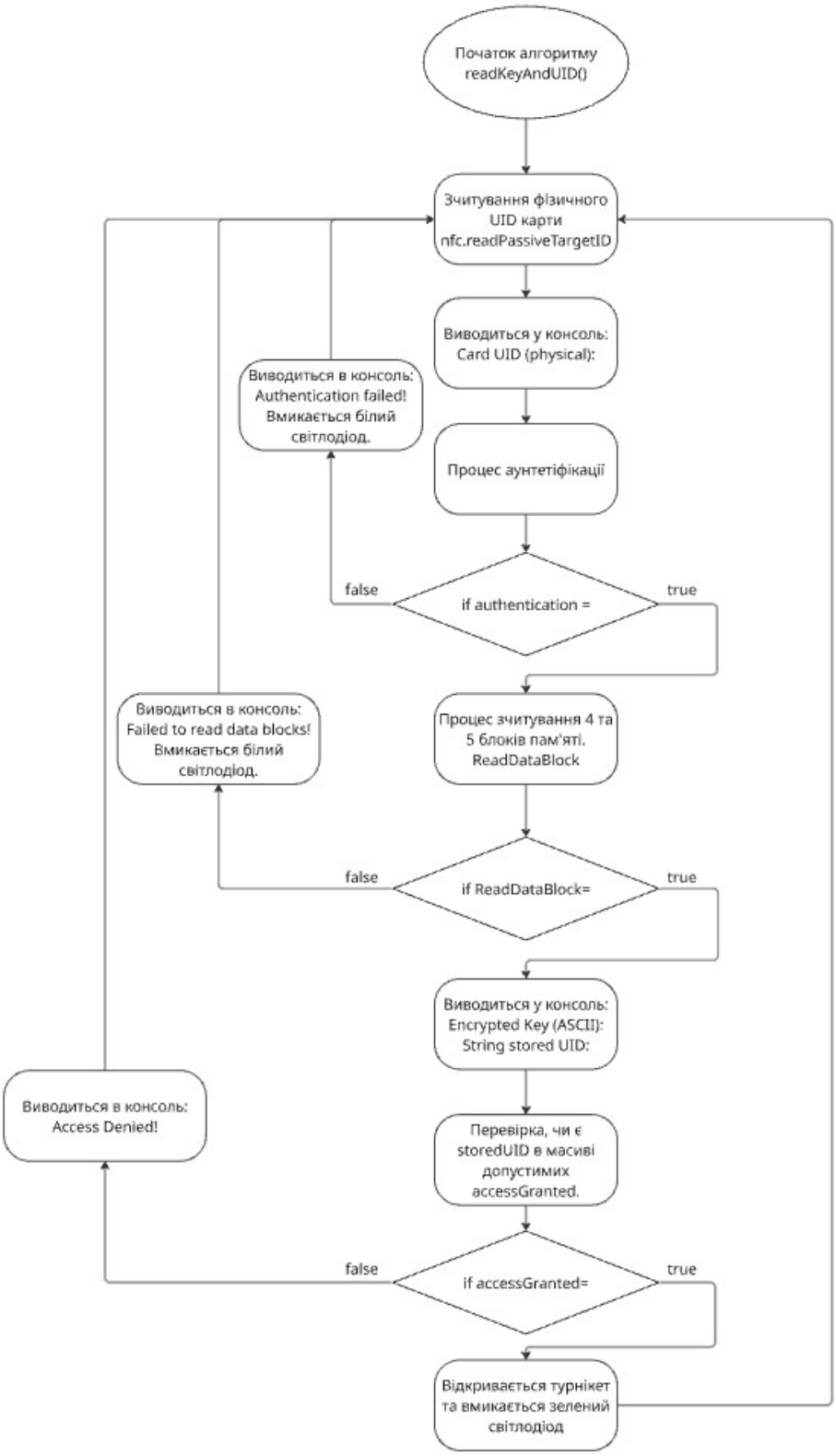

Рис.5.2. Блок-схема readKeyAndUID().

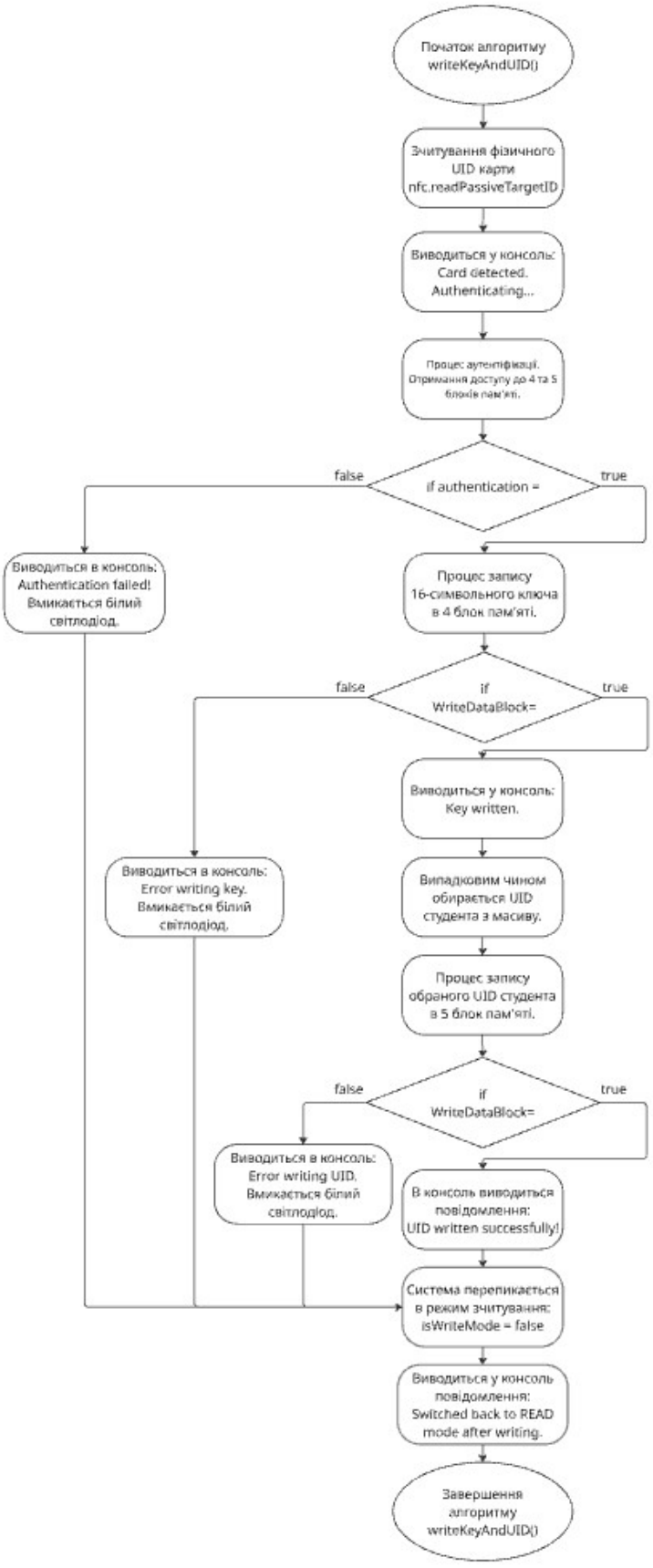

Рис.5.3. Блок-схема writeKeyAndUID().

Як бачимо, після запису в консоль виводиться повідомлення об успішном записі даних:

Key written.

Вибирається випадковий UID студента з масива allowedCount[i] та записується в 5 блок пам’яті:

```
if (nfc.mifareclassic_WriteDataBlock(5, uidData)) {
    Serial.println("UID written successfully!");
    digitalWrite(ledPin, LOW);
    isKeyReady = false;
    incomingKey = "";
}
```

Як тільки запис пройшов успішно, жовтий світлодіод вимикається та в консоль виводиться повідомлення:

UID written successfully!

Якщо один з процесів не процесів не дійшов до кінця, або виникла помилка, то в консоль виводиться відповідне повідомлення, де вказується який саме процес не дійшов до кінця, та вмикається білий світлодіод, котрий вказує на помилку. Після завершення запису на карту ключа змінна isWriteMode = false. Це означає, що система автоматично перейшла знову у режим зчитування даних з картки та в консоль виводиться відповідне повідомлення:

Switched back to READ mode after writing.

Повна програма з Arduino міститься у додатку А.

3.2. База даних Microsoft Access

База даних розроблена у програмі Microsoft Access і є ключовим компонентом системи, який забезпечує зберігання, обробку та візуалізацію даних про студентів та їх відвідування. Microsoft Access – дуже потужний інструмент для швидкой розробки бази даних, в якому є весь необхідний функціонал для досягнення нашої мети. Архітектура бази даних була

зроблена досить просто для того, щоб система працювала стабільно та достатньо швидко для великого потоку студентів.

З чого саме складається база даних:

1. Дві таблиці (Students та AttendanceLog);
2. Форма Add_Student;
3. Форма AutoStart.

3.2.1. Структура таблиць

В таблиці “Students” (рис.6.1) міститься основна інформація про студентів і включає до себе наступні поля:

1. ID (AutoNumber) – первинний ключ для кожного студента, також рахує кількість студентів у базі даних.
2. FirstName (Text, 50 символів) – ім’я студента.
3. LastName (Text, 50 символів) – прізвище студента.
4. Course (Number) – номер курсу навчання.
5. Specialty (Text, 100 символів) – назва спеціальності.
6. StudentKey (Text, 16 символів) – унікальний ключ, записаний на RFID-картку. Для цього поля встановлено обмеження унікальності та перевірку на довжину рядка (ровно 16 символів).

Students

ID	FirstName	LastName	Specialty	Course	StudentKey
2	Євген	Ємельянов	АКІТ	4	V7J5ROgkM3V3Q6xF
3	Михайло	Михайлов	АКІТ	3	sm9DEUwLVtllzO6c
4	Михаило	Віталійович	АКІТ	5	1wogLTsmT6KeM9Gt
5	Михайло	Михайлов	АКІТ 151	1	coCaVQluhbKC6qQh
(New)				0	

Рис.6.1. Таблиця Students. (Ім’я студентів як приклад заповнення)

Головна особливість програми Microsoft Access полягає в тому, що можна дуже швидко та легко додати ще столбці для додакових даних, наприклад, RegistrationDate (Date/Time – дата реєстрації студента) та Photo (OLE object – поле для зберігання фотографії).

AttendanceLog | Students

ID	StudentKey	DetectedTime	FirstName	LastName	Course	Specialty
33	V7J5ROgkM3'	03.05.2025 11:09:15	Євген	Ємельянов	4	АКІТ
34	coCaVQluhbK	03.05.2025 11:09:35	Михайло	Ємельянов	1	АКІТ 151
35	sm9DEUwLVtl	03.05.2025 11:09:51	Михайло	Михайлов	3	АКІТ
36	sm9DEUwLVtl	03.05.2025 11:10:05	Михайло	Михайлов	3	АКІТ
37	coCaVQluhbK	03.05.2025 11:10:20	Михайло	Ємельянов	1	АКІТ 151
38	sm9DEUwLVtl	03.05.2025 11:10:30	Михайло	Михайлов	3	АКІТ
39	V7J5ROgkM3'	03.05.2025 11:10:51	Євген	Ємельянов	4	АКІТ
40	V7J5ROgkM3'	03.05.2025 11:11:24	Євген	Ємельянов	4	АКІТ
41	V7J5ROgkM3'	03.05.2025 11:39:52	Євген	Ємельянов	4	АКІТ
42	V7J5ROgkM3'	03.05.2025 11:41:53	Євген	Ємельянов	4	АКІТ
43	coCaVQluhbK	03.05.2025 11:42:02	Михайло	Михайлов	1	АКІТ 151
44	sm9DEUwLVtl	03.05.2025 11:42:14	Михайло	Михайлов	3	АКІТ
45	V7J5ROgkM3'	03.05.2025 11:42:20	Євген	Ємельянов	4	АКІТ
46	sm9DEUwLVtl	03.05.2025 11:42:56	Михайло	Михайлов	3	АКІТ
(New)		02.06.2025 9:21:10			0	

Рис. 6.2. Таблиця AttendanceLog. (Ім'я студентів як приклад заповнення).

Таблиця «AttendanceLog» (рис. 6.2) зберігає в собі кожне сканування карток, містить в собі такі столбці:

1. ID (AutoNumber) – первинний ключ для кожного студента, також рахує кількість студентів.
2. StudentKey (Text, 16 символів) – унікальний ключ, записаний на RFID-картку.
3. DetectedTime (Date/Time) – фіксує точний час і дату сканування картки.
4. FirstName (Text, 50 символів) – ім'я студента.
5. LastName (Text, 50 символів) – прізвище студента.
6. Course (Number) – номер курсу навчання.
7. Specialty (Text, 100 символів) – назва спеціальності.

В цій таблиці зберігається кожний студент, котрий відсканував картку, та був присутній на занятті.

3.2.2. Форми бази даних.

Форма "Add_Student" (рис.6.3) призначена для додавання нових студентів до системи. Містить в собі поля для введення особистої інформації студентів (ім'я, прізвище, курс, спеціальність), також містить поле для 16-

символьного ключа, де виводиться сгенерований ключ, але ключ також можна вносити власноруч.

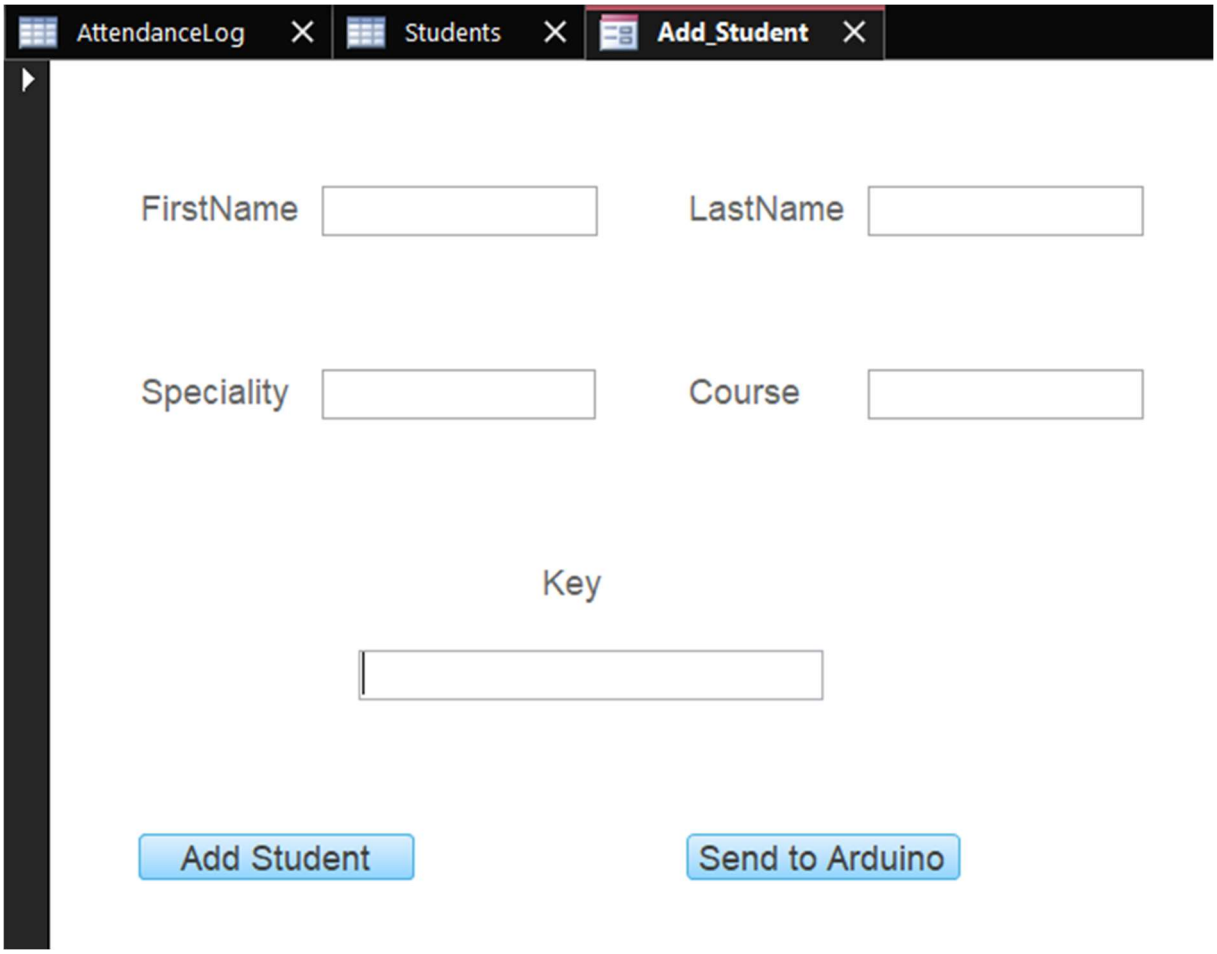

Рис.6.3. Форма Add_Student.

Також ця форма має в собі дві кнопки, а саме:

1. Add Student – натискаючи на цю кнопку, після заповнення форми особистими даними студентів, форма генерує випадковий 16 символьний ключ та зберігає всі дані у таблиці “Students”.
2. Send to Arduino – ця кнопка визиває додатковий виконуючий файл SendToArduino.ps1, відправляючи ключ на Arduino для запису на нову RFID-картку.

Форма “AutoStart” не містить в собі ніяких елементів для користувача, працює в фоновому режиму. Але виконує дуже важливий алгоритм, а саме:

1. З моменту запуску запускає виконуючий файл “MonitorArduino.ps1” для моніторингу активності Arduino.

2. Кожні 5 секунд аналізує вміст файлу Incoming_keys.txt. Якщо файл пустий, то скрипт чекає першу строку, коли в ней з’явиться ключ. Якщо ключ з’явився в першій строчці, то алгоритм чекає слідуючий ключ на наступній строчці, а отриманий ключ записується у стовбець StudentKey.
3. При заповненні стовбця StudentKey, форма автоматично перевіряє чи є такий ключ в таблиці “Students” та заповнює всі інші столбці з персональними даними студентів. Столбець DetectedTime заповнюється автоматично.

   Весь VBA програмний код вказан у додатках Б, В.

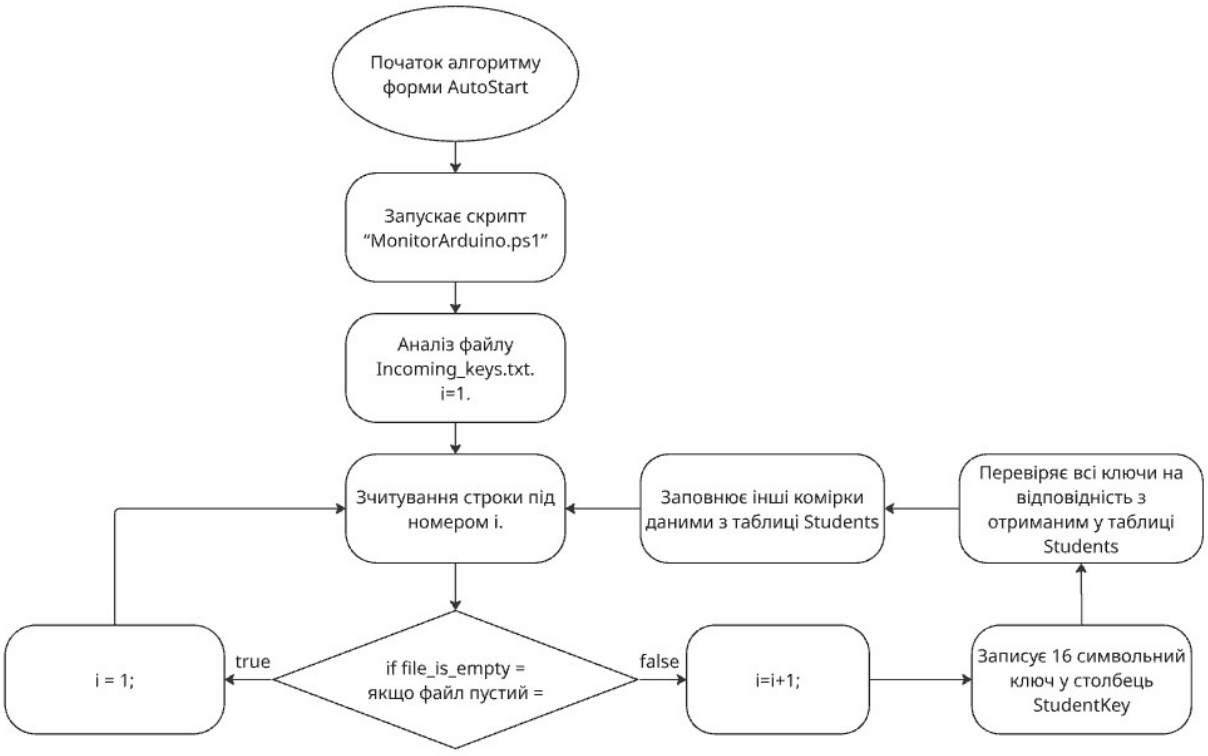

Рис. 6.4. Блок-схема форми AutoStart.

3.3. Інтеграція через PowerShell

Для забезпечення зв’язку між базою даних та апаратною частиною Arduino використовується PowerShell – потужний інструмент автоматизації від Microsoft. Ця технологія поєднує в собі функціонал командного рядка з

можливостями повноцінної мови програмування, що робить її ідеальним рішенням для наших задач. У нашій системи було реалізовано два ключових PowerShell-скрипта, які забезпечують двосторонній обмін даними: SendToArduino.ps1 та MonitorArduino.ps1.

Як зазначалося раніше, після натискання кнопки SendToArduino запускається саме скрипт SendToArduino.ps1 (рис7.1). Цей скрипт відповідає за передачу 16-символьного ключа, згенерованого базою даних. Він отримує ідентифікатор, відправлений з бази даних, через вхідний параметр, після чого встановлює з’єднання з COM-портом, через який підключен Arduino. Скрипт спочатку надсилає команду «1» в консоль Arduino, яка перемикає систему у режим запису, після чого надсилає сам ідентифікатор. Весь процес супроводжується логуванням у спеціальний файл для подальшого аналізу та налагодження. Повний код представлен у додатку Г.

Другий скрипт, MonitorArduino.ps1 (рис.7.2), виконує зворотну функцію – він постійно відстежує COM-порт на предмет надходження індентифікатора від мікроконтролера. Коли студент прикладає свою RFID-картку до зчитувача, Arduino формує відповідний сигнал і передає через послідновний порт у консоль. PowerShell-скрипт зчитує ідентифікатор, здійнює його попередню обробку, фільтрацію, перевірку коректності і зберігає у тимчасовому файлі Incoming_keys.txt. Для запобігання переповнення дискового простору та помилок реалізовано атоматичну очистку цього файлу через певний інтервал часу. Цей скрипт запускається автоматично після запуску форми “AutoStart”. Недоліком цього алгоритму є постійний цикл, з якого можна вийти тільки зупинив програму на апаратному рівні. Треба ще додати блок розгалуження для закриття COM-порту, для цього потрібно додати у форму AutoStart відповідний вимикач, котрий буде зупиняти цей процес. Повний код цього скрипта знаходиться у додатку Д.

Рис. 7.1. Блок-схема алгоритму SendToArduino.ps1.

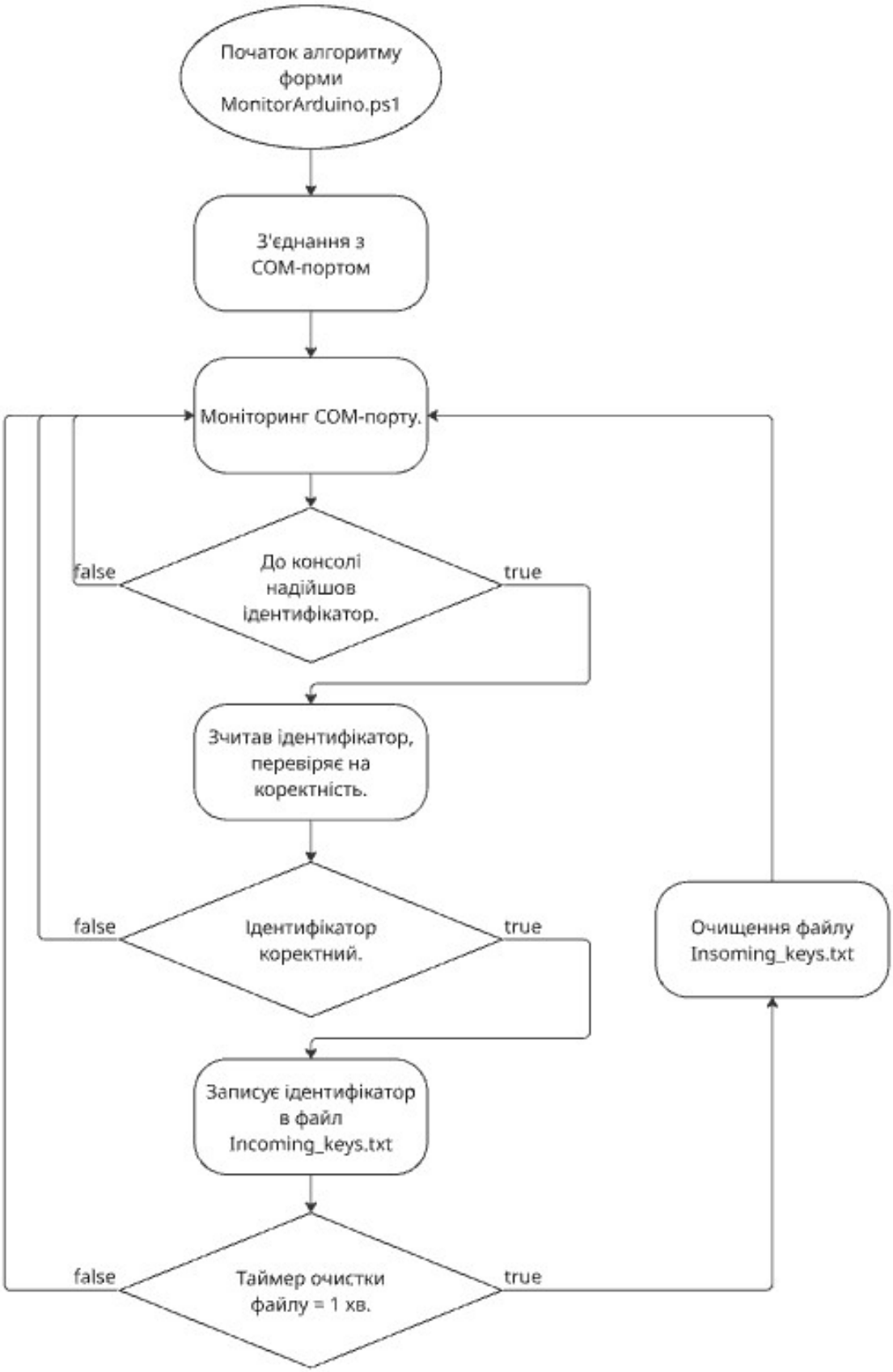

Рис. 7.2. Блок-схема алгоритму MonitorArduino.ps1.

Основною перевагою використання PowerShell у цій системі є його інтеграція з операційною системою Windows, зокрема з Microsoft Access. Це дає нам можливість здійснювати безперервний обмін даними між різними компонентами системи уникаючи розробки додаткового проміжного програмного забезпечення.

Крім того, PowerShell має дуже багато інструментів для обробки помилок і логування подій, що значно підвищує надійність всієї системи. Вищеописані скрипти працюють у фоновому режимі, не завантажуючи систему і не заважаючи основним процесам. Одна з важливих особоливостей даного рішення є його гнучкість – при необхідності можна досить легко модифікувати існуючи скрипти або додати нові, не вносячи багато змін в основну структуру системи, бази даних або апаратну частину. Через велику кількість інформації в інтернеті, інтегрувати новий функціонал достатньо зручно та просто. Таким чином, PowerShell – проміжна ланка між апаратною та програмною частинами. Таке рішення нам дозволило створити ефективний, надійний і масштабований механізм обміну даними, який цілком відповідає поставленим вимогам і завданням проекту.

3.4 Результати та тестування

Таким чином було побудовано робочу модель системи автоматизованого обліку студентів, яка поєднує в собі електроний блок на базі Arduino та базу даних Access. Обмін даними між апаратною частиною та базою даних реалізовано через PowerShell-скрипти. Переваги такої системи полягає у тому, що все навантаження рівномірно розподілено між компонентами цієї системи. Мікроконтролер виконує досить прості але дуже важливі алгоритми, не втручаючись у роботу бази даних та комп’ютера. В той самий час база даних, котра знаходиться на комп’ютері, виконує більш складні задачі, але через більшу потужність центрального процесору всі обчислення та алгоритми проходять достатньо швидко.

Під час експлуатаційних випробувань розробленої системи автоматизації обліку відвідувань було проведено комплексне тестування всіх її компонентів та функцій. Тестування охопило як апаратну частину (Arduino з модулем PN532), так і програмні модулі (базу даних Access та PowerShell-скрипти).

При тестуванні налаштовувалися час відгуку та відправлення даних на Arduino. Основна помилка відзаначалася у тому, що база даних та PowerShell-скрипти дуже швидко надсилали команди на мікроконтролер, коли ще не відкрився COM-порт. Для відкриття COM-порту потрібно певний час (в середньому 1 секунда). Тому PowerShell-скрипти налаштовувалися з врахуванням цієї затримки. Також між надсиланням команд налаштовувалися проміжок часу (0.5 – 1 секунда), для того щоб мікронтролер встигав відреагувати на запит та коректно відпрацювати. Також помилка відбувалася при зчитуванні даних з файла Incoming_keys.txt. Потрібно було також налаштувати проміжки часу видалення всієї інформації з файла та його зчитування. Виникала проблема, що ідентифікатор записався в файл, але дуже швидко видалився, VBA форма AutoStart не встигала записати дані у відповідний столбець таблиці AtendanceLog. Всі ці помилки були виправленні під час тестування. Розрахована точність системи становить близько 90-95%. При точному налаштуванні можна добитися й 100% результату, якщо проводити тестування на велику кількість студентів та тривалий час.

Також виникали помилки при доступі до COM-порту. При роботі з COM-портом потрібно враховувати, що тільки одна програма може працювати з COM-портом, після виконання програми потрібно закрити COM-порт, та знову відкрити іншою програмою моніторингу або передавання даних. Кожен компонент (апаратна частина, PowerShell-інтеграція та база даних) оптимально доповнює один одного, забезпечуючи високу продуктивність при мінімальних ресурсних витратах.

3.5. Шляхи подальшого розвитку системи

Звісно, що розроблена система не вдосконалена, але через достатньо вдалу структуру системи вона має великий потенціал для подальшого розвитку:

1. Якщо потрібно фіксувати не тільки вхід студента в аудіторію а також й вихід з аудіторії то в систему можна додати другий зчитувач, котрий буде зчитувати ідентифікатор на виході з аудіторії та передавати на базу даних. В той час база даних буде фіксувати час виходу з аудіторії.
2. Звя’зок з базою даних. Зараз апаратна частина пов’язана з системою через фізичний провідник. Але є декілька способів зв’язку, а саме: радіозв’язок, по мережі WiFi та через технологію зв’зяку Bluetooth.
3. Також для закріплення апаратної частини системи на деякій поверхні, а також, щоб захистити компоненти, рекомендується розробити корпус, наприклад, роздрукувати на 3Д принтері.
4. NFC технологія. Контролер зчитування PN532 підтримує роботу з NFC-мітками, котрі встановлюються у смартфони. Це дає нам змогу при розробці власного телефоного застосунка додати функцію пропуску в аудіторії за допомогою звичайного смартфону. Якщо у студента немає у смартфоні NFC-мітки то можна видати звичайну RFID-картку.

РЕЗУЛЬТАТИ РОБОТИ ТА ВИСНОВКИ

В роботі було розроблено модель комплексного апаратно-програмного рішення обліку відвідувань занять студентами. На даний момент система здатна суттєво прискорити цей процес без участі викладача або інших робітників навчального закладу.

В результаті проведенної роботи були отримані такі результати:

1. Проведено аналіз існуючих технологічних рішень автоматизії та була обрана доступна та надійна RFID-технологія.
2. Система була реалізована з універсальних апаратних засобів та модулів, що дає можливості для масштабування та вдосконалення.
3. Програмне забезпечення було розроблено таким чином, що все навантаження рівномірно розподіляється між компонентами системи.
4. Оцінка роботи моделі вказує на можливість економити до 10-15 хвилин навчального процесу.

З наведених результатів можна зробити висновок, що система знаходиться тільки на початку розробки і має перспективи для подальшого розвитку.

_____ Ємельянов Є.М.
(Підпис автора роботи)

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Петро Бідюк, Володимир Бондарчук. Сучасні методи біометричної ідентифікації Інститут прикладного системного аналізу. *Національного технічного університету «Київський політехнічний інститут»,* 2009. С. 138
   URL: https://ela.kpi.ua/server/api/core/bitstreams/7f1251ba-7156-4730-8a08-3ae82ddbc1f3/content
2. Onecollab: Ollie Rayburn, Understanding the Security Risks of QR Codes. 2023.
   URL: https://onecollab.co.uk/news-and-insights/from-convenience-to-vulnerability-understanding-the-security-risks-of-qr-codes/#:~:text=Data%20Privacy%20Risks,and%20safety%20of%20QR%20codes.
3. RFID Journal: Дослідження тестувань RFID систем, їх точність. URL: https://www.rfidjournal.com/news/gs1-netherlands-rfid-benchmark-results-show-sales-boost-inventory-accuracy/187214/
4. EDUCASE: Jason O. Hallstrom. IoT and the Campus of Things. 2016.
   URL: https://er.educause.edu/articles/2016/8/iot-and-the-campus-of-things
5. Arduino LLC: Official Arduino UNO Technical Specifications. 2023.
   URL: https://docs.arduino.cc/hardware/uno-rev3/#tech-specs
6. PN532/C1 Near Field Communication (NFC) controller. 2017.
   URL: https://www.nxp.com/docs/en/nxp/data-sheets/PN532_C1.pdf
7. Finkenzeller, K. RFID Handbook: Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication. 2010.
8. Microsoft Access офіційний сайт. URL:
   https://support.microsoft.com/uk-UA/access
9. Офіційна документація PowerShell.
   URL: https://learn.microsoft.com/uk-ua/powershell/

10. Офійний сайт Microsoft: приклади роботи з COM-портом. URL:https://learn.microsoft.com/uk-ua/dotnet/api/system.io.ports.serialport?view=net-9.0-pp

Додаток А. Програмний код Arduino.

```
#include <Wire.h>
#include <Adafruit_PN532.h>
#define SDA_PIN A4
#define SCL_PIN A5
#define RELAY_PIN 7
Adafruit_PN532 nfc(SDA_PIN, SCL_PIN);
int ledPin = 2;
int ledPinErr = 8;
bool isWriteMode = false;
bool isKeyReady = false;
String incomingKey = "";
String tagID = "";
uint8_t defaultKeyA[6] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
String allowedUIDs[] = {
  "A1B2C3D4", "1A2B3C4D", "1234ABCD", "DEADBEEF"
};
const int allowedCount = sizeof(allowedUIDs) / sizeof(allowedUIDs[0]);
void setup() {
  Serial.begin(9600);
  Serial.println("Initializing NFC module...");

  pinMode(ledPin, OUTPUT);
  pinMode(ledPinErr, OUTPUT);
  pinMode(RELAY_PIN, OUTPUT);
  digitalWrite(RELAY_PIN, LOW);
  digitalWrite(ledPin, LOW);
  digitalWrite(ledPinErr, LOW);
  isWriteMode = false;
  nfc.begin();
```

```
  uint32_t versiondata = nfc.getFirmwareVersion();
  if (!versiondata) {
    Serial.println("Didn't find PN532 board");
    while (1);
  }
  nfc.SAMConfig();
  Serial.println("Ready. Send '1' for WRITE mode, '2' for READ mode.");
}
void loop() {
  if (Serial.available()) {
    String command = Serial.readStringUntil('\n');
    command.trim();
    if (command == "1") {
      isWriteMode = true;
      Serial.println("Switched to WRITE mode.");
    } else if (command == "2") {
      isWriteMode = false;
      Serial.println("Switched to READ mode.");
    } else if (isWriteMode) {
      if (command.length() == 16) {
        incomingKey = command;
        isKeyReady = true;
        Serial.println("Received new key for writing. Please tap the card.");
        digitalWrite(ledPin, HIGH); // Ожидание карты
      } else {
        Serial.println("Invalid key length! Must be 16 characters.");
        digitalWrite(ledPin, LOW);
        digitalWrite(ledPinErr, HIGH);
        delay(3000);
        digitalWrite(ledPinErr, LOW);
```

```
      }
    }
  }
  if (isWriteMode && isKeyReady) {
    writeKeyAndUID();
  } else if (!isWriteMode) {
    readKeyAndUID();
  }
}
void writeKeyAndUID() {
  uint8_t uid[7];
  uint8_t uidLength;
  bool success = nfc.readPassiveTargetID(PN532_MIFARE_ISO14443A, uid,
&uidLength, 50);
  if (success) {
    Serial.println("Card detected. Authenticating...");
    if (nfc.mifareclassic_AuthenticateBlock(uid, uidLength, 4, 0, defaultKeyA) &&
        nfc.mifareclassic_AuthenticateBlock(uid, uidLength, 5, 0, defaultKeyA)) {
      Serial.println("Authentication successful. Writing key and UID...")
      uint8_t keyData[16] = {0};
      incomingKey.getBytes(keyData, 17);
      if (nfc.mifareclassic_WriteDataBlock(4, keyData)) {
        Serial.println("Key written.");
        int index = random(allowedCount);
        String randomUID = allowedUIDs[index];
        Serial.print("Selected UID to write: ");
        Serial.println(randomUID);
        uint8_t uidData[16] = {0};
        randomUID.getBytes(uidData, 17);
```

```
      if (nfc.mifareclassic_WriteDataBlock(5, uidData)) {
        Serial.println("UID written successfully!");
        digitalWrite(ledPin, LOW);
        isKeyReady = false;
        incomingKey = "";
      } else {
        Serial.println("Error writing UID.");
        digitalWrite(ledPinErr, HIGH);
        delay(2000);
        digitalWrite(ledPinErr, LOW);
      }
    } else {
      Serial.println("Error writing key.");
      digitalWrite(ledPinErr, HIGH);
      delay(2000);
      digitalWrite(ledPinErr, LOW);
    }
  } else {
    Serial.println("Authentication failed!");
    digitalWrite(ledPinErr, HIGH);
    delay(2000);
    digitalWrite(ledPinErr, LOW);
  }
  isWriteMode = false;
  Serial.println("Switched back to READ mode after writing.");
  }
}
void readKeyAndUID() {
  uint8_t uid[7];
  uint8_t uidLength;
```

```
  bool success = nfc.readPassiveTargetID(PN532_MIFARE_ISO14443A, uid,
&uidLength, 50);
  if (success) {
    Serial.print("Card UID (physical): ");
    for (uint8_t i = 0; i < uidLength; i++) {
      if (uid[i] < 16) Serial.print("0");
      Serial.print(uid[i], HEX);
    }
    Serial.println();
    if (nfc.mifareclassic_AuthenticateBlock(uid, uidLength, 4, 0, defaultKeyA) &&
        nfc.mifareclassic_AuthenticateBlock(uid, uidLength, 5, 0, defaultKeyA)) {
      Serial.println("Authentication successful. Reading data...");
      uint8_t keyData[16];
      uint8_t uidData[16];
      if (nfc.mifareclassic_ReadDataBlock(4, keyData) &&
nfc.mifareclassic_ReadDataBlock(5, uidData)) {
        // Вывод ключа
        Serial.print("Encrypted Key (ASCII): ");
        for (int i = 0; i < 16; i++) {
          Serial.print((char)keyData[i]);
        }
        Serial.println();
        String storedUID = "";
        for (int i = 0; i < 16; i++) {
          if (uidData[i] != 0) storedUID += (char)uidData[i];
        }
        Serial.print("Stored UID: ");
        Serial.println(storedUID);
        bool accessGranted = false;
        for (int i = 0; i < allowedCount; i++) {
```

```
      if (storedUID == allowedUIDs[i]) {
        accessGranted = true;
        break;
      }
    }
    if (accessGranted) {
      Serial.println("✅ Access granted!");
      digitalWrite(RELAY_PIN, HIGH);
      delay(3000);
      digitalWrite(RELAY_PIN, LOW);
    } else {
      Serial.println("❌ Access Denied!");
      digitalWrite(RELAY_PIN, LOW);
    }
  } else {
    Serial.println("❌ Failed to read data blocks!");
    digitalWrite(ledPinErr, HIGH);
    delay(2000);
    digitalWrite(ledPinErr, LOW);
  }
 } else {
  Serial.println("❌ Authentication failed!");
  digitalWrite(ledPinErr, HIGH);
  delay(2000);
  digitalWrite(ledPinErr, LOW);
 }
 delay(3000);
 Serial.println();
} }
```

Додаток Б. Форма AddStudent.

```
#If VBA7 Then
    Private Declare PtrSafe Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As
LongPtr)
#Else
    Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
#End If
Private Sub btnAddStudent_Click()
    Dim Key As String
    Dim symbols As String
    Dim i As Integer
    ' Генерація 16-символьного ключа
    symbols =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456
789"
    Randomize
    Key = ""
    For i = 1 To 16
        Key = Key & Mid(symbols, Int((Len(symbols) * Rnd) + 1), 1)
    Next i
    ' Додаємо запис до таблиці Students
    CurrentDb.Execute "INSERT INTO Students (FirstName, LastName, Course,
Specialty, StudentKey) " & _
                      "VALUES ('" & Me.FirstName & "', '" & Me.txtLastName & "', '" &
Me.txtCourse & "', '" & Me.txtSpecialty & "', '" & Key & "')"
    MsgBox "Student added successfully!" & vbCrLf & "Key: " & Key,
vbInformation
    ' Очищаємо поля введення
    Me.FirstName = ""
    Me.txtLastName = ""
```

```
    Me.txtCourse = ""
    Me.txtSpecialty = ""
    ' Записуємо ключ у текстове поле для відправлення на Arduino
Me.txtStudentKey = Key
End Sub
Private Sub btnSendToArduino_Click()
    Dim Key As String
    Dim killCommand As String
    Dim psCommand As String
    Dim result As Double
    Key = Me.txtStudentKey.Value
    If Len(Key) <> 16 Then
        MsgBox " Ключ має бути 16 символів!", vbExclamation
        Exit Sub
    End If
    ' Команда запуска PowerShell с передачей ключа
    psCommand = "powershell -ExecutionPolicy Bypass -File
""E:\Students_Arduino_Temp\SendToArduino.ps1"" -Key """ & Key & """"
    result = shell(psCommand, vbHide)
    MsgBox " Ключ відправлено на Arduino!", vbInformation
End Sub
```

Додаток В. Форма AutoStart.

```
Option Compare Database
Option Explicit
Dim lastLineIndex As Long
Private Sub Form_Load()
    lastLineIndex = 1
    ' Запуск PowerShell скрипта в скрытом режиме
    Dim cmd As String
    cmd = "powershell.exe -ExecutionPolicy Bypass -WindowStyle Hidden -File
""E:\Students_Arduino_Temp\MonitorArduino.ps1"""
    shell cmd, vbHide
End Sub
Private Sub Form_Timer()
    Dim filePath As String
    Dim fileLines() As String
    Dim fso As Object, ts As Object
    Dim i As Long
    Dim Key As String
    filePath = "E:\Students_Arduino_Temp\Incoming_keys.txt"
    On Error Resume Next
    Set fso = CreateObject("Scripting.FileSystemObject")
    If fso.FileExists(filePath) Then
        Set ts = fso.OpenTextFile(filePath, 1) ' ForReading
        fileLines = Split(ts.ReadAll, vbCrLf)
        ts.Close
        Debug.Print "? Индекс чтения (до): " & lastLineIndex
        ' Якщо файл повністю порожній - скинути індекс
        If UBound(fileLines) = 0 And Trim(fileLines(0)) = "" Then
            lastLineIndex = 0
            Debug.Print "? Файл пуст. Индекс сброшен до 0."
```

```
            Exit Sub
        End If
        If UBound(fileLines) >= lastLineIndex Then
            For i = lastLineIndex To UBound(fileLines)
                Key = Trim(fileLines(i))
                If Len(Key) = 16 Then
                    InsertAttendance Key
                    Debug.Print "? Ключ записан: " & Key
                    lastLineIndex = lastLineIndex + 1
                ElseIf Len(Key) > 0 Then
                    Debug.Print "? Невалідний ключ: " & Key & " (длина: " & Len(Key)
& ")"
                    lastLineIndex = lastLineIndex + 1
                Else
                    Debug.Print "? Пропущено порожній рядок на позиції: " & i
                    ' Не збільшуємо індекс
                End If
            Next i
        End If
        Debug.Print "? Індекс читання (після): " & lastLineIndex
    Else
        Debug.Print "? Файл не знайдено: " & filePath
    End If
End Sub
Private Sub InsertAttendance(StudentKey As String)
    Dim sql As String
    Dim rs As DAO.Recordset
    On Error Resume Next
    StudentKey = Trim(StudentKey)
```

```
    Set rs = CurrentDb.OpenRecordset("SELECT * FROM Students WHERE
StudentKey = '" & StudentKey & "'")
    If Not rs.EOF Then
        sql = "INSERT INTO AttendanceLog (StudentKey, DetectedTime, FirstName,
LastName, Specialty, Course) VALUES (" & _
              "'" & StudentKey & "', Now(), " & _
              "'" & rs!FirstName & "', " & _
              "'" & rs!LastName & "', " & _
              "'" & rs!Specialty & "', " & _
              "'" & rs!Course & "')"

        CurrentDb.Execute sql
        Debug.Print "? Відвідування додано: " & StudentKey
    Else
        Debug.Print "? Студент не знайдено: " & StudentKey
    End If

    rs.Close
    Set rs = Nothing
End Sub
```

Додаток Г. Виконавчий файл SendToArduino.ps1.

```
param (
    [string]$Key
)
$logFile = "E:\Students_Arduino_Temp\log.txt"
$logStamp = "==== $(Get-Date -Format 'MM/dd/yyyy HH:mm:ss') ===="
Add-Content $logFile "`r`n$logStamp"
Add-Content $logFile "Key entered: $Key"
if (-not $Key -or $Key.Length -ne 16) {
    Add-Content $logFile "❌ Invalid key length!"
    Write-Host "The key must contain exactly 16 characters!"
    pause
    exit
}
try {
    $port = New-Object System.IO.Ports.SerialPort "COM9", 9600, 'None', 8, 'One'
    $port.ReadTimeout = 500  # 0.5 second timeout for faster reaction
    $port.Open()
    Start-Sleep -Milliseconds 10000
    $port.WriteLine("1")
    Add-Content $logFile "✅ Sent WRITE command (1)."
    Start-Sleep -Milliseconds 1000

    $port.WriteLine($Key)
    Add-Content $logFile "✅ Key sent: $Key"
    Write-Host "✅ Key sent to Arduino. Listening for responses..."
    $timeoutSeconds = 10
    $startTime = Get-Date
    while ((Get-Date) -lt $startTime.AddSeconds($timeoutSeconds)) {
```

```
        try {
            $line = $port.ReadLine()
            if ($line) {
                $line = $line.Trim()
                Write-Host "Arduino: $line"
                Add-Content $logFile "Arduino: $line"

                # If Arduino says "Ready" or similar — can break early
                if ($line -match "Key successfully written|Error writing key|Invalid key
length") {
                    break
                }
            }
        } catch {
            # Ignore timeout errors
        }
    }
    $port.Close()
}
catch {
    Write-Host "❌ Error: $_"
    Add-Content $logFile "❌ Error: $_"
}
pause
```

Додаток Д. Виконавчий файл MonitorArduino.ps1.

```
# Ім'я порту Arduino
$portName = "COM9"
$baudRate = 9600
# Шлях до файлу для запису ключів
$outFile = "E:\Students_Arduino_Temp\Incoming_keys.txt"
# Час останнього очищення
$lastFlush = Get-Date
$flushIntervalMinutes = 5
# Відкрити COM-порт
$port = New-Object System.IO.Ports.SerialPort $portName, $baudRate, 'None', 8,
'One'
try {
    $port.Open()
    Write-Output "Monitoring Arduino on $portName..."
    # Очистить файл перед началом
    Clear-Content $outFile -ErrorAction SilentlyContinue
    # Нескінченний цикл для читання даних
    while ($true) {
        # Перевірка, чи потрібно очистити файл за часом
        $now = Get-Date
        if (($now - $lastFlush).TotalMinutes -ge $flushIntervalMinutes) {
            Clear-Content $outFile -ErrorAction SilentlyContinue
            $lastFlush = $now
            Write-Output " 🧹 File cleared at $now"
        }
        # Читання даних із порту
        if ($port.BytesToRead -gt 0) {
            $line = $port.ReadLine().Trim()
```

```
            if ($line -like "Encrypted Key (ASCII)*") {
                $key = $line.Replace("Encrypted Key (ASCII):", "").Trim()
                Add-Content -Path $outFile -Value $key
                Write-Output "✅ Received key: $key"
            }
        }
        Start-Sleep -Milliseconds 200
    }
}
finally {
    # Очищення та закриття при завершенні скрипту
    if ($port.IsOpen) { $port.Close() }
    Clear-Content $outFile -ErrorAction SilentlyContinue
    Write-Output "🚪 Port closed and file cleared on exit."
}
```