LONDON
METROPOLITAN
UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title
CS5068NI– Cloud Computing & IoT

RFID-Based IoT Attendance System

Assessment Type
10% Proposal Report / 50% Group Report

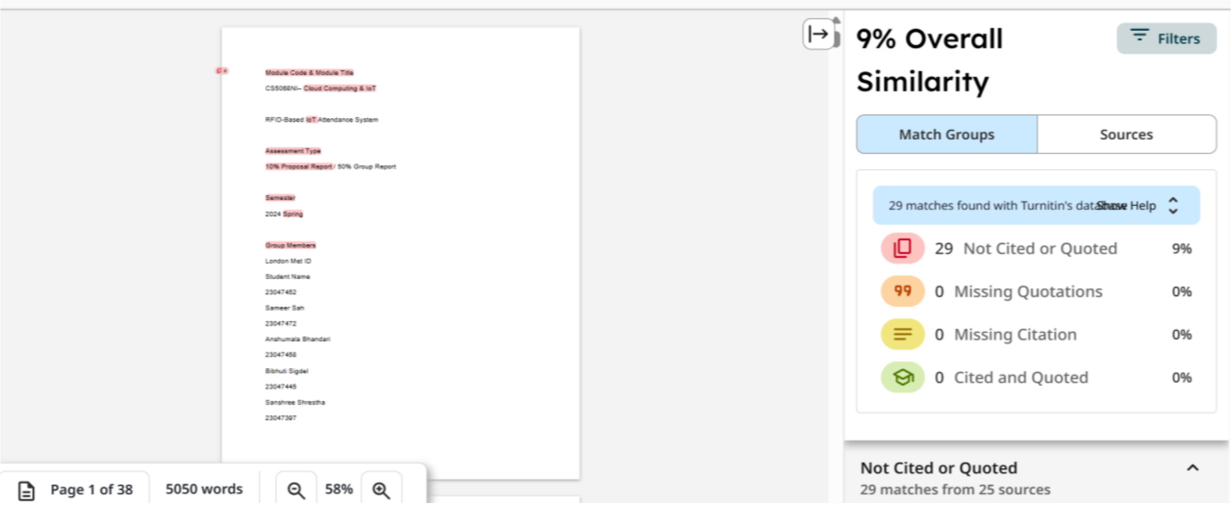
Semester
2024 Spring

Group Members

London Met ID	Student Name
23047452	Sameer Sah
23047472	Anshumala Bhandari
23047458	Bibhuti Sigdel
23047445	Sanshree Shrestha
23047397	Shraddha Budhathoki
23047501	Sagun Budhathoki

Assignment Due Date: 15th May, 2025
Assignment Submission Date: 15th May, 2025
Submitted to: Mr. Sugat Man Shakya
Word Count: 4338

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.



Module Code & Module Title

CS5068NI– Cloud Computing & IoT

RFID-Based IoT Attendance System

Assessment Type

10% Proposal Report / 50% Group Report

Semester

2024 Spring

Group Members

London Met ID

Student Name

23047452

Sameer Sah

23047472

Anshumala Bhandari

23047458

Bibhuti Sigdel

23047445

Sanshree Shrestha

23047397

Page 1 of 38

5050 words

58%

9% Overall Similarity

29 matches found with Turnitin's database

29	Not Cited or Quoted	9%
0	Missing Quotations	0%
0	Missing Citation	0%
0	Cited and Quoted	0%

Not Cited or Quoted

29 matches from 25 sources

Table of Contents

Table of Figures

Table of Tables

1. Introduction

The Internet of Things (IoT) is all about connecting everyday devices to the internet, allowing them to communicate and share data with each other and the cloud. It includes things like home appliances, vehicles, and devices equipped with sensors, software, and network connectivity. In simple terms, IoT enables ordinary objects to become "smart" by allowing them to interact with the internet (Schulze, 2025).

RFID (Radio Frequency Identification) is one of the technologies that plays a key role in IoT. RFID uses electromagnetic fields to automatically identify and track tags attached to objects. Back in the 1990s, RFID was mainly used for asset tracking and inventory control. Over time, as sensors and computing devices became smaller and more affordable, RFID found wider applications, including attendance tracking, access control, and automation in various industries (Almansor, 2021).

For our project, we're applying IoT concepts by creating a useful solution which is **ESP32-based RFID Attendance System.** This system simplifies and automates the process of taking attendance by using RFID cards and a scanner. When a user scans their card, the system records their presence and displays it on an LCD screen. The information can also be stored for future use. This not only saves time but also ensures accuracy and reduces the chances of false attendance, improving overall efficiency in schools, colleges, and workplaces (Kumar, 2024).

1.1 Current Scenario

Taking attendance manually is still a common practice in many institutions, especially in schools, colleges, and offices in Nepal. Teachers or administrators call out names and mark attendance one by one, which is time-consuming and can cause human error. In some cases, students or employees mark attendance for others, leading to false entries or proxy attendance.

To improve this, a few institutions in Nepal have started adopting automated systems like biometric scanners and RFID-based attendance tracking. For example, several private schools in Kathmandu have introduced smart attendance systems using RFID cards. These systems allow students to tap their card on a reader to mark their presence, and the data is recorded instantly.

However, these technologies are not yet widely implemented across the country. Many schools and offices still rely on outdated attendance methods. Limited technical knowledge, cost concerns, and lack of awareness are some of the reasons for this slow adoption (katmatic, 2025).

1.2 Problem Statement and Project as a Solution

In this current world, due to increasing numbers of students and employees, and the use of traditional manual attendance systems, many institutions are facing difficulties in maintaining accurate and efficient attendance records. Manual processes take time, are often inaccurate, and are easily misused. This causes inconvenience, increased workload, and unreliable data (katmatic, 2025).

The **RFID Attendance System** addresses these types of problems by using IoT-based devices like RFID readers, ESP32 boards, and cloud computing to automate the attendance-taking process. This project allows each individual to carry an RFID card, which can be scanned within seconds to mark attendance. The system shows the data on an LCD display and stores it for future reference. It reduces human error, prevents proxy attendance, and saves time for both teachers and administrators.

1.3 Aims and Objectives

Aim

The main aim of this project is to automate attendance management and provide a faster, more accurate, and reliable way of recording presence using IoT-based RFID technology.

Objectives

The objectives of this project are:

- To automate the attendance process using RFID and ESP32 components.
- To reduce time spent on manual attendance taking.
- To prevent false or proxy attendance.
- To provide real-time feedback using LCD display.
- To minimize human involvement and administrative workload.
- To use IoT devices like RFID readers and tags for real-time identification.

2. Background

In today’s digital world, automation is transforming how routine tasks are performed, particularly in educational institutions. One such area is attendance management, where traditional manual systems are often slow, unreliable, and vulnerable to misuse. To solve this, our project introduces a smart **RFID-based Attendance System**, which uses IoT technology to ensure fast, accurate, and secure attendance tracking. Students simply tap their RFID enabled ID cards near a reader, and the system automatically logs their presence in an online database. This approach saves time, prevents proxy attendance, and provides real-time records for staff and administrators. Below is a detailed explanation of the system, including its design and resource needs.

2.1 System overview

The RFID Attendance System is designed to make tracking attendance easier, faster, and more accurate. The system uses a few essential components, including the ESP32 Devkit V1, RFID RC522 Module, RFID Cards and Tags, LCD 20x4 I2C, Breadboard, Jumper Wires, and a USB Cable. These components work together to automatically detect when a student or employee scans their RFID card or tag, logging their attendance in real-time.

At the heart of the system is the ESP32 Devkit V1, which processes the data from the RFID RC522 Reader and sends it to a Google Sheet, where the attendance is stored. The RFID cards or tags each have a unique ID, so the system knows exactly who is present. The LCD display shows immediate feedback, like "Attendance Recorded," so everyone knows their presence has been registered.

The breadboard is used to test and connect all the components before putting everything together, and the jumper wires make sure everything is wired correctly. The USB cable connects the ESP32 to a computer for uploading the necessary code and powers the system during development.

This system replaces the traditional manual process, making it quicker, more reliable, and much less prone to mistakes, which helps save time and ensures accurate attendance records.

2.2 Diagram Design

2.2.1 Block diagram

Below is the image of block diagram of the RFID attendance system.

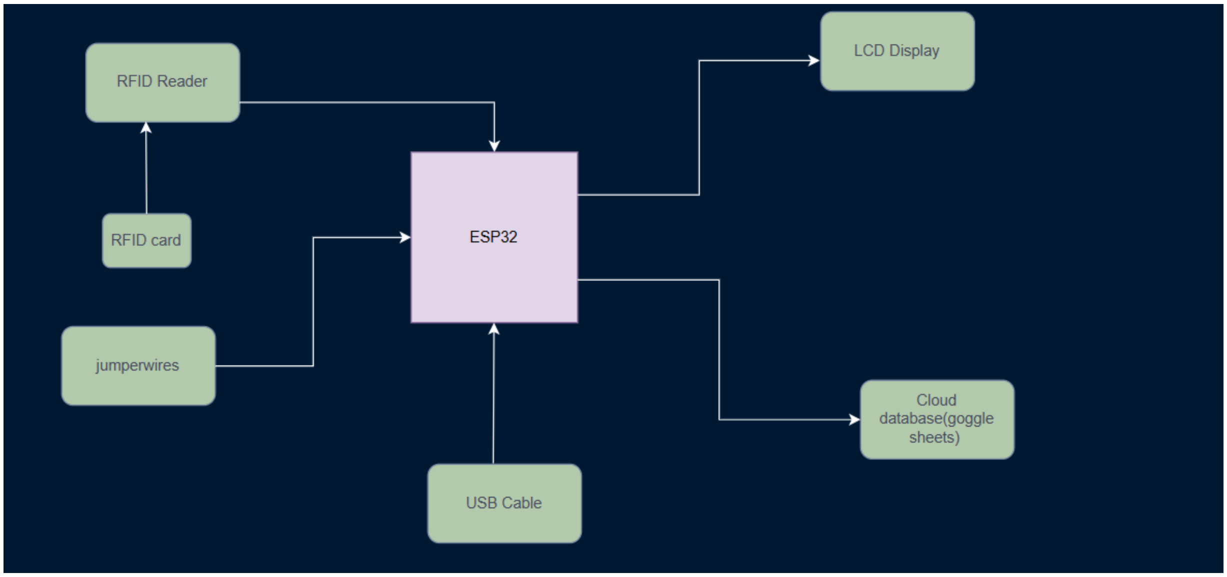


Figure 1: Block diagram of RFID Attendance system

2.2.2 System Architecture

Below is the system architecture of RFID attendance system

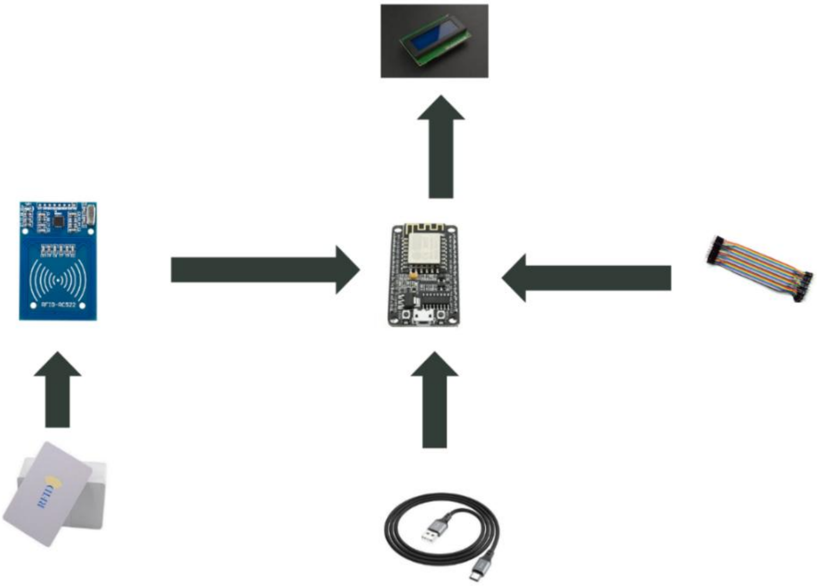


Figure 2: System Architecture of RFID Attendance system

2.2.3 Flowchart

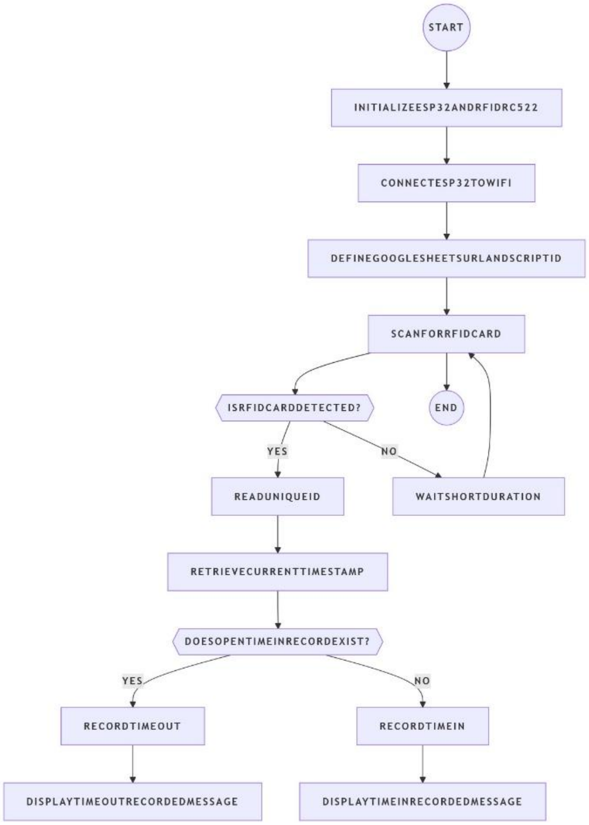


Figure 3: Flowchart I

2.2.4 Circuit Diagram

The designers used Fritzing to produce the circuit layout for the RFID-Based IoT Attendance System. It shows the connection pathway for the components incorporated in the project. The main function of the ESP32 microcontroller entails overseeing data exchange between the RFID module and cloud platform. Both the RFID module and the display system serve distinct functions in the system. The RC522 RFID module enables the ESP32 to read tags from RFID cards and the display shows live messages about card information and attendance status. The ESP32 receives power from USB and all wire connections operate through jumper wires. The diagram shows how the system hardware components need to be connected for normal system functioning.

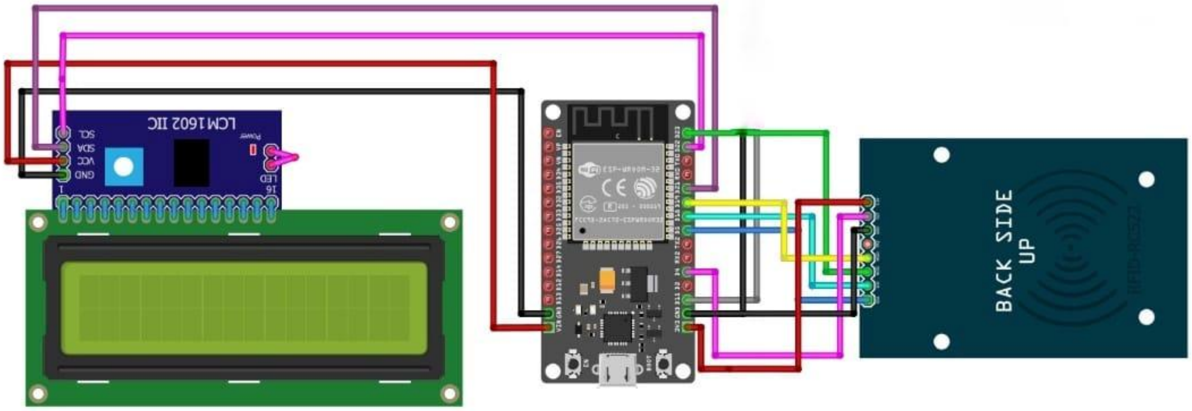


Figure 4: Circuit Diagram

Wiring:

Table 1 Wiring LCD to ESP32

LCD 20x4 L2C	**ESP32 Pin**
VCC	VIN
GND	GND
SDA	D21
SCL	D22

Table 2 Wiring ESP32 to RFID-RC 522

ESP32 Pin	RFID-RC522
D5	SDA/SS
D18	SCK
D23	MOSI
D19	MISO
D4	RST
GND	GND
3V3	3.3V

2.2.5 Schematic Diagram

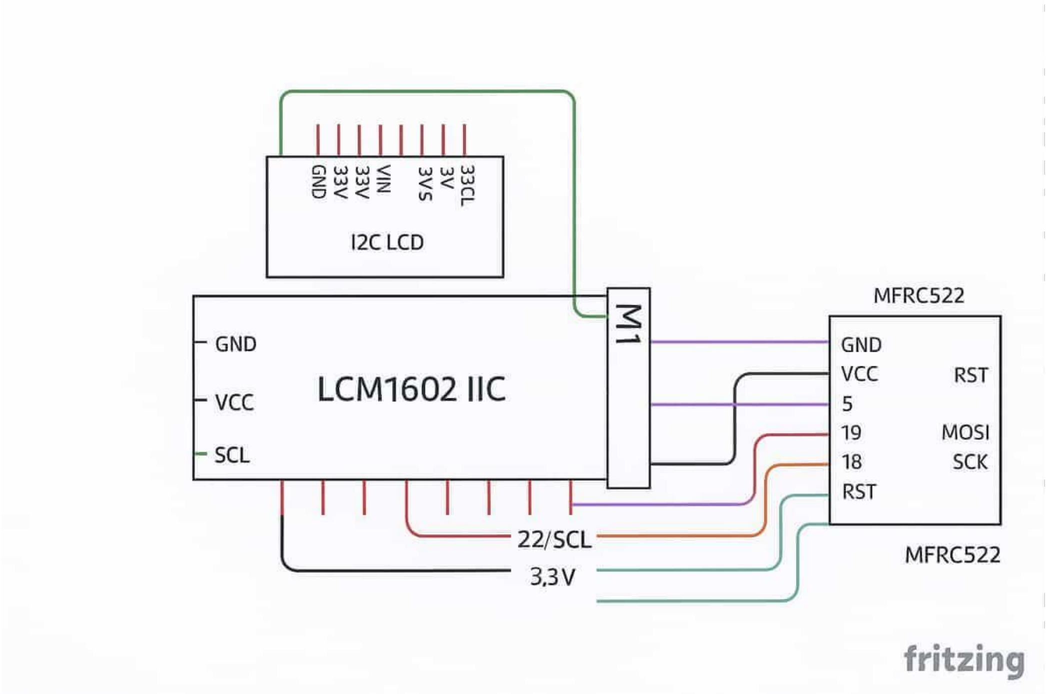


Figure 5: Schematic Diagram

2.3 Data Flow Diagram (DFD)

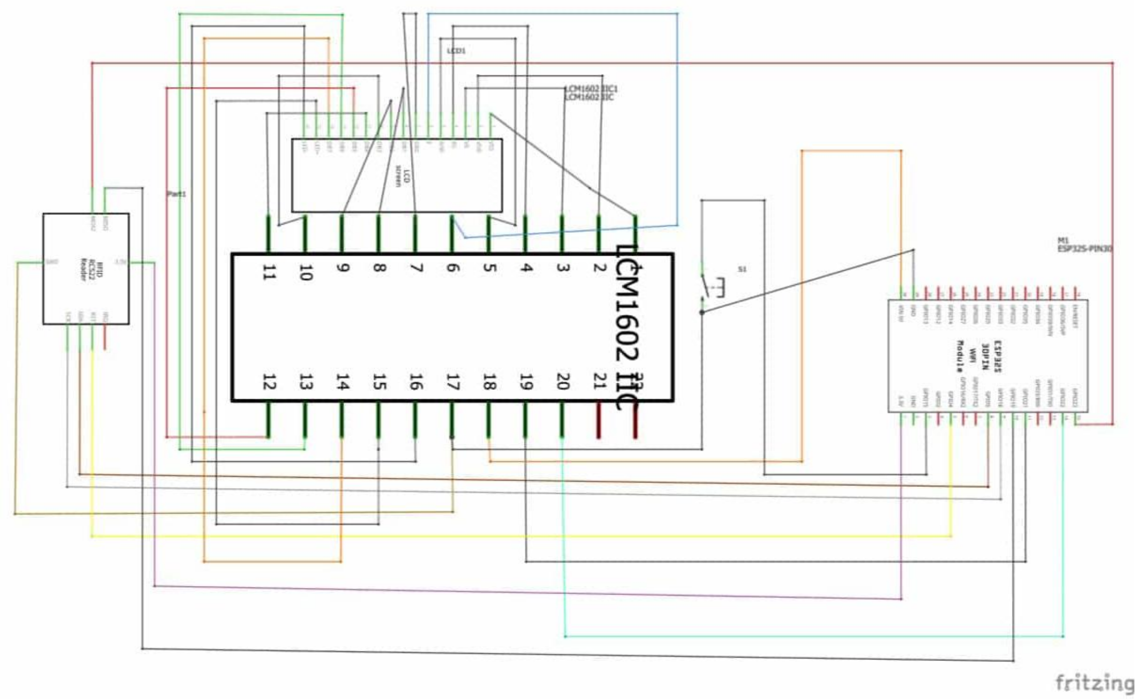


Figure 6: Data Flow Diagram (DFD)

2.4 Requirement Analysis

2.4.1 Hardware Components

1. ESP32 Devkit V1.

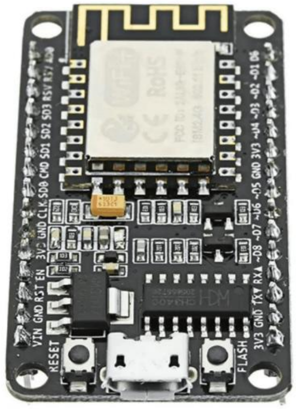


Figure 7: ESP32 Devkit V1 (inventkart, 2025)

The ESP32 DevKit V1 is a beginner-friendly board by Espressif Systems, featuring the ESP32-WROOM-32 module. It's designed for developing IoT and embedded applications, allowing easy connection to sensors and other peripherals. It comes in two versions 30-pin and 38-pin offering different numbers of GPIO pins but the same core functionality (azadtechhub, 2024).

2. RFID

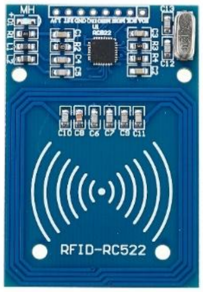


Figure 8: picture demonstrating RFID RC522 Module (zemfiraaksenova, 2019)

The RFID RC522 is a 13.56 MHz reader module designed to communicate with RFID tags adhering to the ISO 14443A standard. It creates an electromagnetic field to interact with RFID tags and can communicate with microcontrollers via SPI, I²C, or UART protocols allowing for efficient data exchange in various applications, including access control and attendance systems. (Sanketh, 2020).

3. RFID Tags

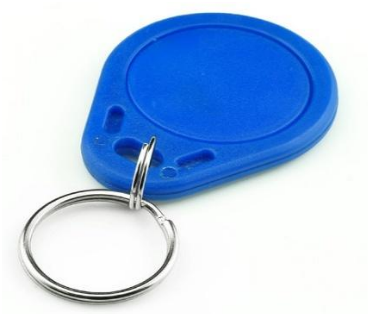


Figure 9: picture demonstrating RFID tag (Anon., 2025)

An RFID tag is a compact device that contains a microchip and antenna, enabling it to store and transmit data wirelessly. When brought near an RFID reader, the tag's unique identifier is detected, facilitating processes like inventory tracking or attendance logging without physical contact (kaczor, 2025).

4. RFID Card

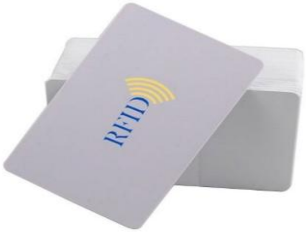


Figure 10:picture demonstrating RFID Card (Anon., 2025)

Similar to RFID tags, RFID cards are embedded with a microchip and antenna but are typically in the form of a standard-sized card. They are commonly used for identification purposes, allowing users to gain access or record attendance by simply tapping the card near an RFID reader (zheng, 2025).

5. Breadboard

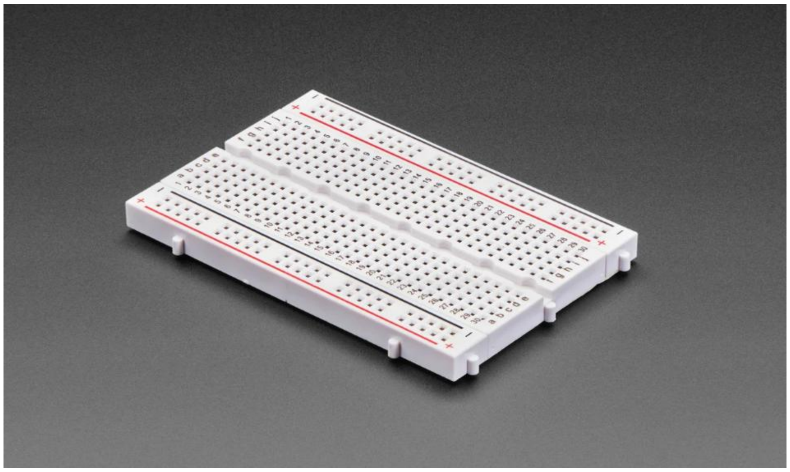


Figure 11:picture demonstrating breadboard (petergong, 2025)

A breadboard is a reusable platform that allows developers and students to build and test electronic circuits without the need for soldering. It consists of a grid of small sockets into which electronic components and jumper wires can be inserted to create temporary circuits. This makes it especially useful in the prototyping phase, where changes and adjustments are frequent. The internal metal strips inside the breadboard connect rows of holes, enabling smooth power and signal flow between components (WatElectronic, 2021).

6. Jumper Wires

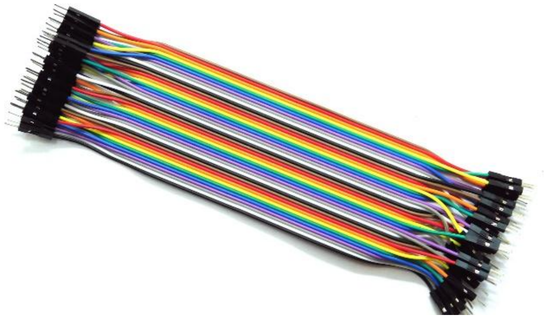


Figure 12: picture demonstrating jumper wires (nettigo, 2025)

Jumper wires are insulated conductors used to establish electrical connections between components on a breadboard or other prototyping platforms. They come in various lengths and connector types, facilitating flexible and temporary circuit configurations during development (wiltronics, 2022).

7. LCD 16x4 I2C

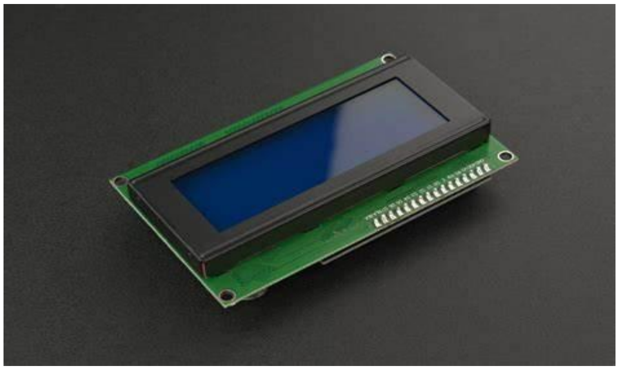


Figure 13: picture demonstrating LCD 20X4 12C (elecena, 2025)

The Liquid Crystal Display (LCD) is commonly utilized in embedded systems to showcase text and numerical data for users as an output device. The 16×4 alphanumeric display relies on the Hitachi HD44780 driver IC, a small black circular chip located at the back of the LCD module. This controller governs the LCD's operation, and through Arduino, we can interact with it to transmit commands and display text messages (magdey, 2025).

8. USB Cable

Figure 14: picture demonstrating USB cable (Anon., 2025)

USB (Universal Serial Bus) cables are a type of connection used to transfer data and provide power between devices such as computers, smartphones, printers, cameras, and more. These cables have a standardized connector that ensures compatibility across a wide range of devices (Anker, 2024).

2.4.2 Software Requirements

1. Arduino IDE

The **Arduino IDE** is a free and open-source software used to write, compile, and upload code to Arduino boards like the Arduino Uno. The IDE allows integration of libraries and modules, making it ideal for building embedded system projects.

2. MS word

Figure 15: MS word (logosworld, 2024)

Microsoft Word is a component of the Microsoft Office suite that allows users to create, edit, and format text-based documents. It offers features such as spell check, grammar correction, and a range of text formatting options. Users can also insert images, tables, and charts to enhance their documents (Adams, 2025).

3. Draw.io

Figure 16: Draw.io (fittyclub, 2025)

Draw.io is a proprietary tool designed for creating diagrams and charts. It offers options for both automatic and custom layouts, allowing for flexible design. With a wide variety of shapes and visual elements, users can craft unique diagrams and charts. The drag-and-drop functionality makes it easy to design professional-looking visuals (hope, 2024).

4. Google sheets

Figure 17: Google sheets (stickpng, 2025)

Google Sheets is an online tool that allows users to create, edit, and manage spreadsheets while sharing the data in real time. This product from Google provides standard spreadsheet functionalities, such as adding, deleting, and sorting rows and columns (chai, 2025).

5. Fritzing

It is an open-source design which provides users with a tool for developing electronic schematics. The project utilized Fritzing for developing the schematic design of the RFID-based IoT attendance system (soldered, 2023). Through Fritzing users obtained visual representation of how the ESP32 and RFID module (RC522) and LCD elements linked to other hardware components.

3. Development

3.1. Planning and design

At the beginning of the project, our team sat together to discuss different ideas for automation. After some brainstorming, we decided to create a **smart attendance system** using **RFID technology**. We wanted something that could make the attendance process faster, easier, and more organized, especially compared to traditional paper-based methods.

Our main goal was to build a system where students or staff could just tap a card and have their attendance recorded automatically in a Google Sheet no writing, no manual checking. After some research, we selected the **ESP32 board** because it supports **Wi-Fi** and works well with the **RFID RC522 reader**. These two components together allowed us to read RFID cards and send that data to Google Sheets over the internet.

We also needed to figure out how to connect everything and send data online. So, in the early stage, we tested if the RFID reader could detect cards properly and checked if we could display that information in the **Serial Monitor**. We also practiced sending small amounts of data to **Google Sheets** through a special web link (called a Web App URL) to see if our idea was even possible. After these tests went well, we were confident to move forward with full development.

3.2. Resource collection

Required Components

- **ESP32 Dev Module**: Microcontroller with Wi-Fi for data transmission.
- **RFID RC522 Module**: Reads 13.56 MHz RFID tags/cards via SPI.
- **LCD 16x4**: To Display the required Information as it read RFID.
- **USB Type-C Cable**: For programming ESP32 and supplying 5V power.
- **Breadboard**: Prototyping platform for circuit connections.
- **Jumper Wires**: Male-to-male and male-to-female for connections.
- **RFID Tags/Cards**: MIFARE 1K cards for testing (at least 2 for multiple users).
- **Computer with Arduino IDE**: For programming the ESP32.

- **Wi-Fi Network**: Stable internet for ESP32 to connect to Google Sheets.
- **Google Account**: To create and manage Google Sheet and Apps Script.

Sourcing

- Source components from electronics suppliers (e.g., Amazon, eBay, local stores). Ensure the ESP32 has a Type-C port (e.g., ESP32-WROOM-32).
- Use a data-capable USB Type-C cable (not charge-only).
- RFID tags/cards are often included with the RC522 module.
- Download Arduino IDE from arduino.cc.
- Access Google Sheets and Apps Script via sheets.google.com.

We collected most of the hardware from our college's IoT lab. We submitted a request to **Shishir Subedi sir**, and after approval, we got access to the required tools. A few items like RFID Reader, jumper wires, Type-C cable and small decorative materials (cardboard, glue, etc.) were bought by the team members themselves.

3.3. System development

Phase 1: Connecting ESP32 and Breadboard

Objective: Verify ESP32 programming and Serial Monitor output using the USB Type-C cable.

Wiring:

- Connect the ESP32 to the computer via the USB Type-C cable.

- No additional components are needed for this phase.

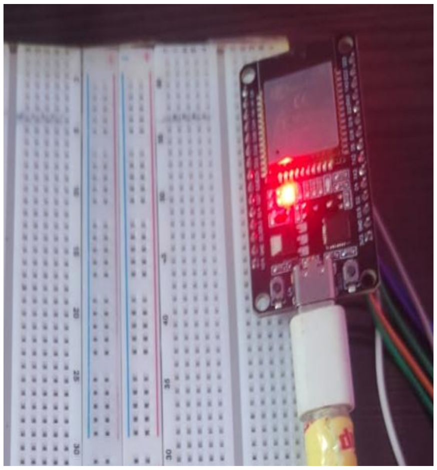


Figure 18 Connecting ESP32 with breadboard

Outcome: The Serial Monitor displays "ESP32 RFID Attendance System" and "System Ready". If not, check the USB Type-C cable, board selection (e.g., "ESP32 Dev Module"), and COM port.

Phase 2: RFID RC522 Connection with ESP32

Objective: Interface the RFID RC522 with ESP32 to read card UIDs using raw SPI communication.

Wiring:

Table 3 System Development

RC522 Pin	**ESP32 Pin**	**Description**
3V3	3.3V	Power supply (RC522 uses 3.3V)
GND	GND	Ground via breadboard
RST	D4	Reset pin
SDA (SS)	D5	Slave select for SPI
SCK	D18	SPI clock
MOSI	D23	SPI master out, slave in

MISO	D19	SPI master in, slave out
IRQ	Not connected	Interrupt (not used)

Steps:

1. Connect the RC522 to the ESP32 as per the table. Use 3.3V power to avoid damaging the RC522.
2. Upload a sketch that implements basic SPI communication to initialize the RC522 and read card UIDs.
3. Use the Serial Monitor to display the UID.

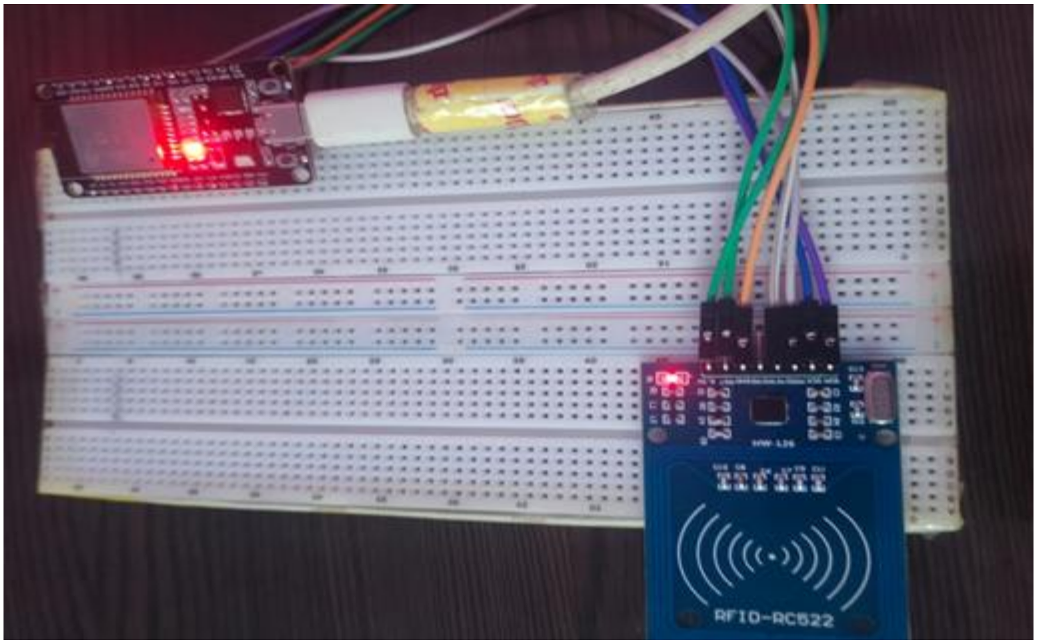
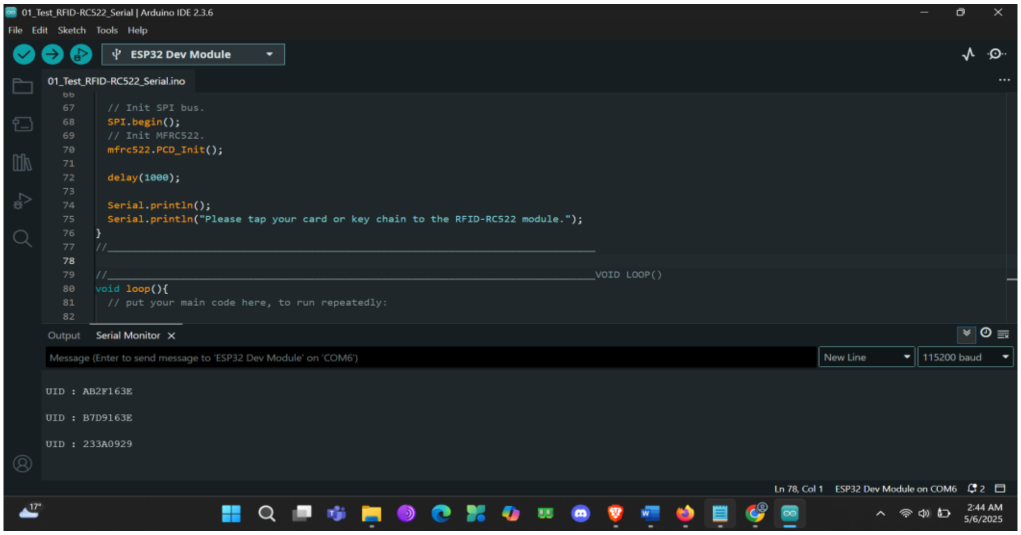


Figure 19 Connecting Esp32 with RFID module



01_Test_RFID-RC522_Serial | Arduino IDE 2.3.6
File Edit Sketch Tools Help
ESP32 Dev Module
01_Test_RFID-RC522_Serial.ino

```
  // Init SPI bus.
  SPI.begin();
  // Init MFRC522.
  mfrc522.PCD_Init();

  delay(1000);

  Serial.println();
  Serial.println("Please tap your card or key chain to the RFID-RC522 module.");
}
//________________________________________________________________________________

//________________________________________________________________________________VOID LOOP()
void loop(){
  // put your main code here, to run repeatedly:

```

Output
Serial Monitor
Message (Enter to send message to 'ESP32 Dev Module' on 'COM6')
New Line
115200 baud

```
UID : AB2F163E

UID : B7D9163E

UID : 233A0929
```

Ln 78, Col 1 ESP32 Dev Module on COM6
2:44 AM
5/6/2025

Figure 20 RFID RC522 Connection and UID Reading

Outcome: Scanning an RFID card displays its UID (e.g., "1A2B3C4D") on the Serial Monitor. If no UID is detected, verify 3.3V power, wiring, and SPI pin assignments. This code uses minimal RC522 commands for card detection and UID reading.

Phase 3: LCD Connection with ESP32

Objective: Interface the LCD with ESP32 to display the UID and name as the information using raw SPI communication.

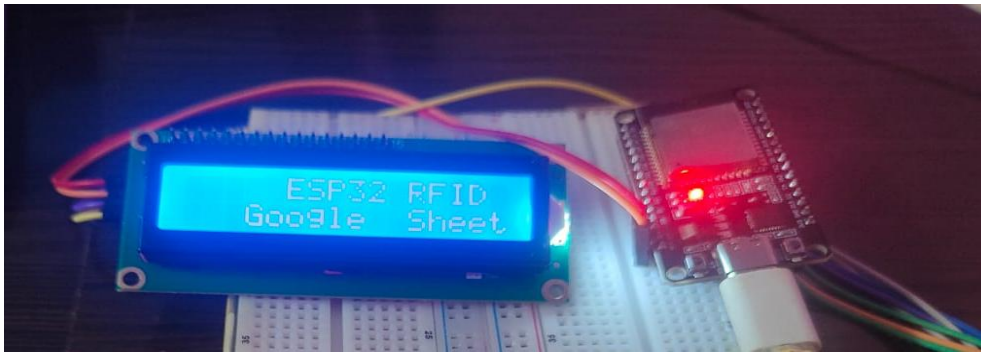


Figure 21 LCD Connection with ESP32

LCD 16x4 I2C	ESP32
GND	GND
VCC	VIN
SDA	D21

SCL	D22

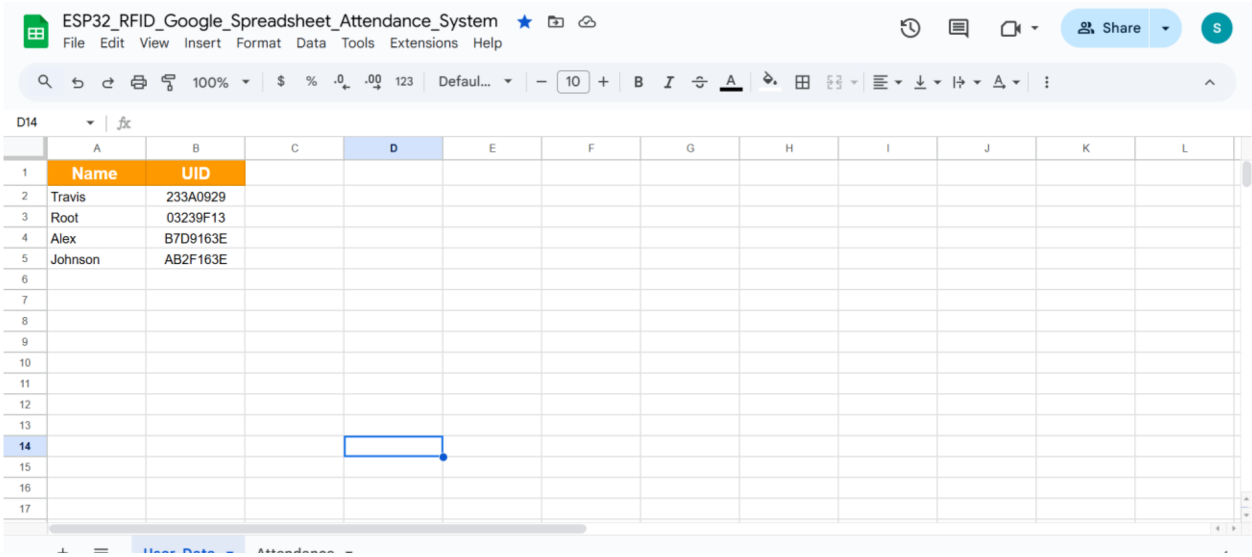
Table 4 LCD Connection with ESP32

Google Sheets Setup:

1. Create a new Google Sheet named "RFID Attendance".
2. Add headers in the first row: UID, Timestamp.

Steps:

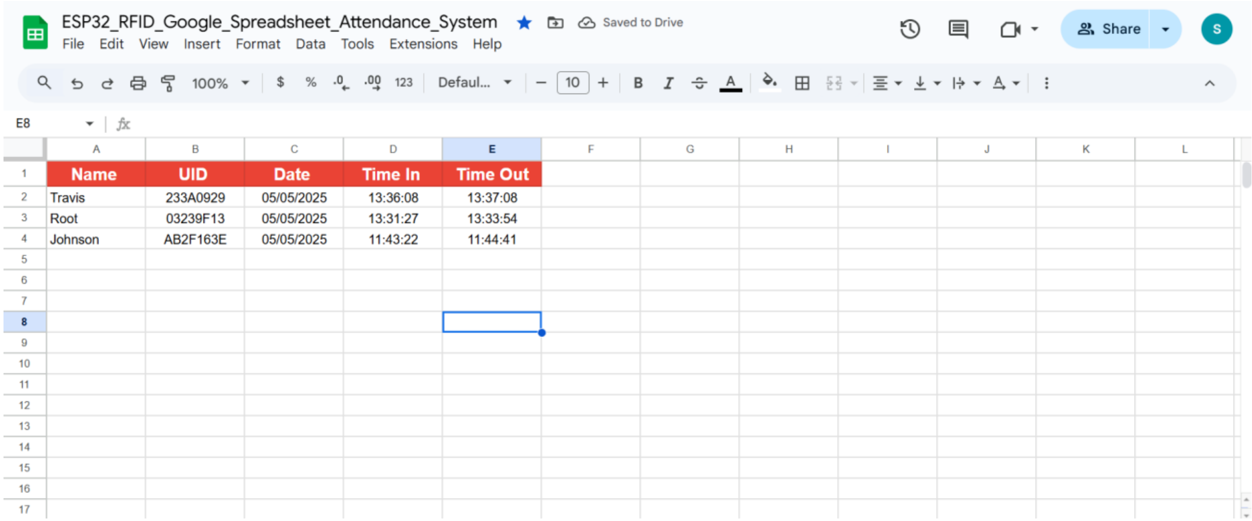
1. Ensure the ESP32 board package is installed (see Phase 1).
2. Upload the code using the USB Type-C cable.
3. Scan an RFID card and check the Google Sheet for a new row with the UID and timestamp.



ESP32_RFID_Google_Spreadsheet_Attendance_System

Name	UID
Travis	233A0929
Root	03239F13
Alex	B7D9163E
Johnson	AB2F163E

Figure 22 Google Sheets Integration - Data Registration


ESP32_RFID_Google_Spreadsheet_Attendance_System
Saved to Drive
File Edit View Insert Format Data Tools Extensions Help
100%
Defaul...
10
Share
E8

Name	UID	Date	Time In	Time Out
Travis	233A0929	05/05/2025	13:36:08	13:37:08
Root	03239F13	05/05/2025	13:31:27	13:33:54
Johnson	AB2F163E	05/05/2025	11:43:22	11:44:41

Figure 23 Google Sheets Integration - Attendance Validation

Outcome: The ESP32 reads the RFID UID and logs it to the Google Sheet with a timestamp. The Serial Monitor shows "Data sent to Google Sheet" on success or an error code on failure.

Phase 4: Full System Integration

Objective: Combine RFID reading and Google Sheets logging into a cohesive system.

Final Wiring:

- Use the RC522 connections from Phase 2.
- Ensure the USB Type-C cable powers the ESP32 and RC522 via the breadboard’s 3.3V rail.
- Verify all connections are secure, as the RC522 is sensitive to power fluctuations.

Final Code: The code from Phase 3 is the complete implementation, using only built-in SPI, WiFi, and HTTPClient.

Testing:

1. Power the system via the USB Type-C cable connected to a computer or 5V USB adapter.
2. Scan multiple RFID cards and verify:
    - The Serial Monitor displays the UID and "Data sent to Google Sheet" or an error.
    - The Google Sheet updates with the UID and timestamp for each scan.

3. Check Serial Monitor for debugging (e.g., Wi-Fi status, HTTP codes).

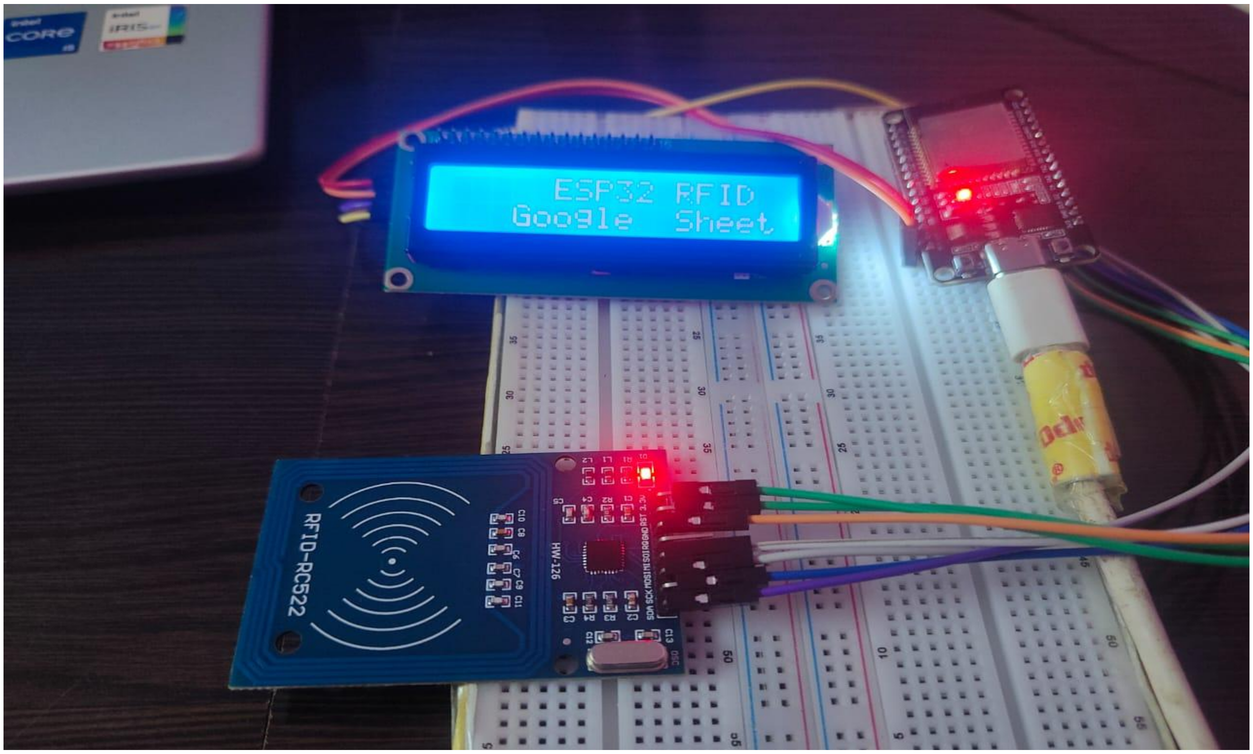
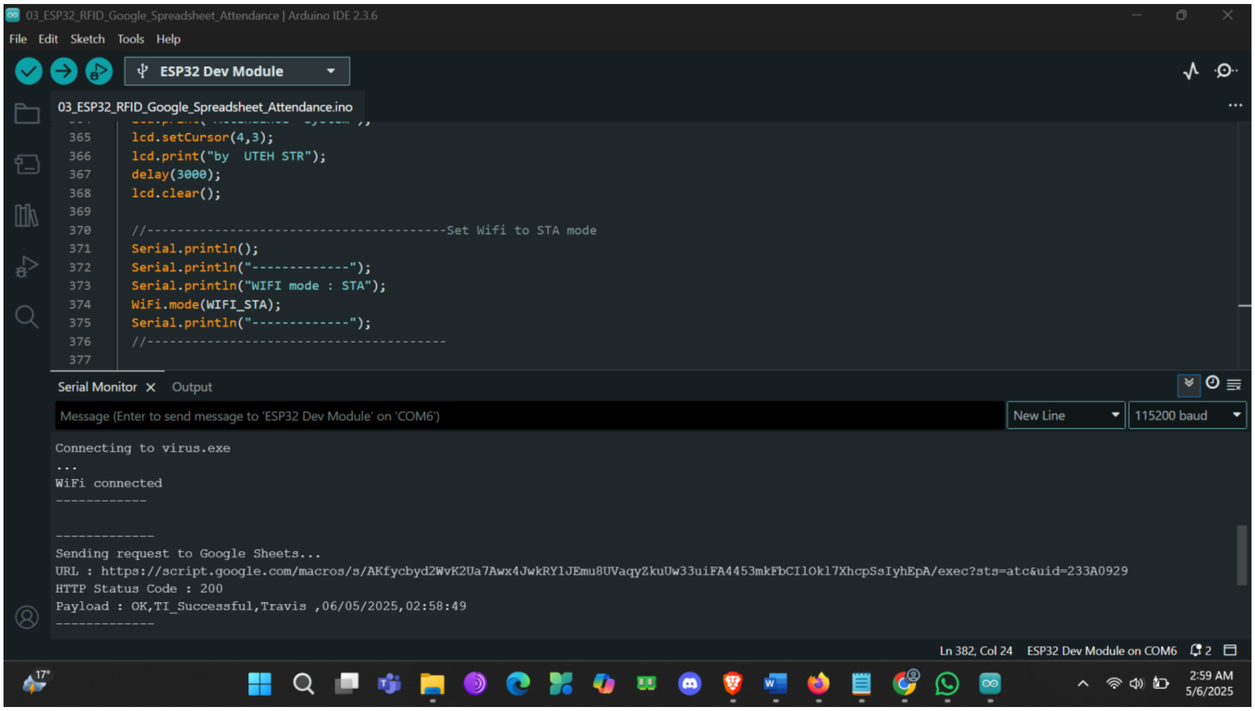


Figure 24 Final System Integration


03_ESP32_RFID_Google_Spreadsheet_Attendance | Arduino IDE 2.3.6
File Edit Sketch Tools Help
ESP32 Dev Module
03_ESP32_RFID_Google_Spreadsheet_Attendance.ino
Serial Monitor
Output
Message (Enter to send message to 'ESP32 Dev Module' on 'COM6')
New Line
115200 baud
Ln 382, Col 24
ESP32 Dev Module on COM6
2
17°
2:59 AM
5/6/2025

```
  lcd.setCursor(4,3);
  lcd.print("by  UTEH STR");
  delay(3000);
  lcd.clear();

  //----------------------------------------Set Wifi to STA mode
  Serial.println();
  Serial.println("-------------");
  Serial.println("WIFI mode : STA");
  WiFi.mode(WIFI_STA);
  Serial.println("-------------");
  //----------------------------------------
```

```
Connecting to virus.exe
...
WiFi connected
------------

-------------
Sending request to Google Sheets...
URL : https://script.google.com/macros/s/AKfycbyd2WvK2Ua7Awx4JwkRY1JEmu8UVaqyZkuUw33uiFA4453mkFbCIlOkl7XhcpSsIyhEpA/exec?sts=atc&uid=233A0929
HTTP Status Code : 200
Payload : OK,TI_Successful,Travis ,06/05/2025,02:58:49
-------------
```

Figure 25 Final System Demonstration IDE

Outcome: A functional RFID attendance system that logs data to Google Sheets in real-time, using no external libraries. The USB Type-C cable ensures reliable programming and power.

3.4 Notes and Troubleshooting

- **RC522 Power**: Use 3.3V for the RC522 to avoid damage. If card detection fails, check wiring and power stability.
- **Wi-Fi Issues**: Verify SSID and password. Restart the ESP32 if the connection fails and monitor Serial output.
- **Google Sheets**: Ensure the Apps Script Web App is deployed with "Anyone" access. Test the URL independently.
- **USB Type-C**: Use a data-capable Type-C cable. Charge-only cables won’t program the ESP32.
- **SPI Communication**: The raw SPI code is simplified and may not handle all RC522 features. If unreliable, consider using the MFRC522 library (against your request) for robust performance.

3.5 Potential Improvements

- **Feedback Mechanism**: Add an LED or buzzer (using GPIO pins) for scan confirmation, avoiding the need for an LCD and its library.
- **User Database**: Map UIDs to usernames in the Google Sheet manually or via a separate script.
- **Security**: Ensure the Google Apps Script URL is secure (HTTPS) and limit access if possible.
- **Portability**: Use a battery pack for standalone operation, with the USB Type-C cable as a backup.

4. Result and Findings

The RFID-based attendance system implementation's findings and results show that the project's main goals have been achieved effectively. The system accurately recorded attendance in real time by effectively detecting and scanning RFID cards. The time needed to take attendance was greatly decreased by this technology, which also removed frequent problems like proxy entries and human error. It was a viable substitute for traditional attendance methods in educational institutions since it was faster and more reliable.

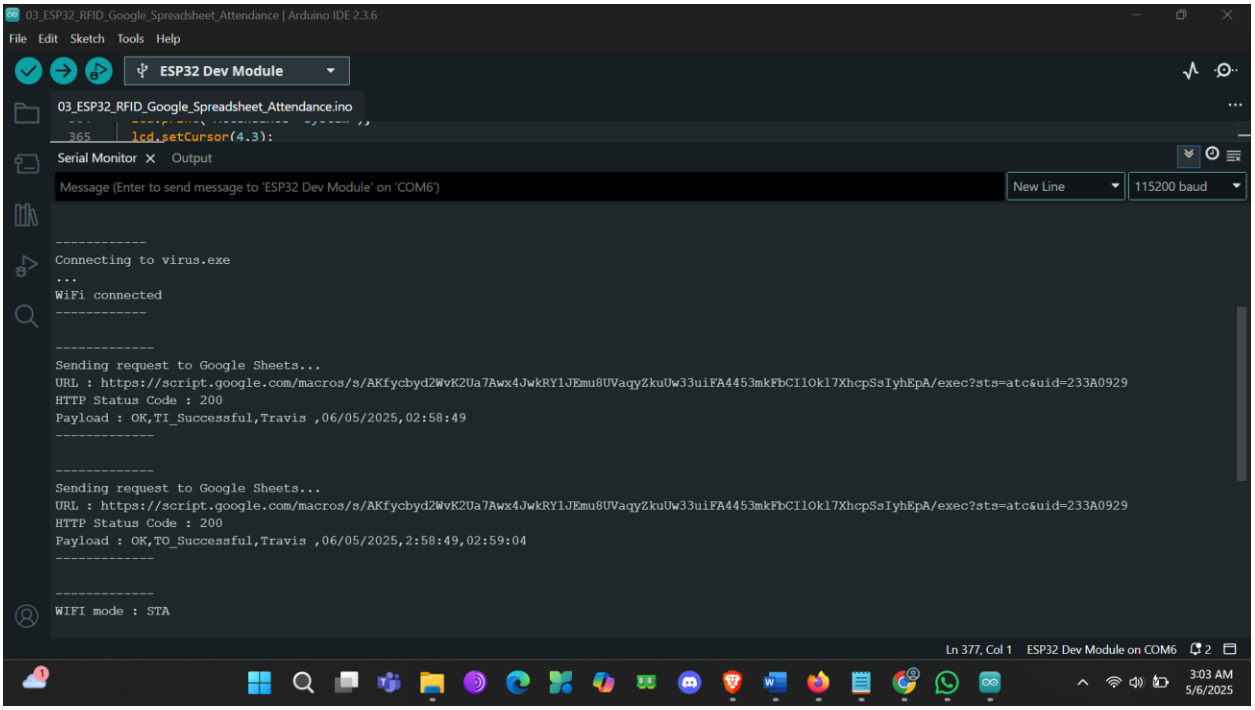
Regarding data management, the system enabled efficient storage and easy access to attendance records. Logging data electronically made it simpler to monitor and analyse attendance trends, which can support academic evaluations and administrative tasks. Additionally, the overall project was budget-friendly, as it relied on low-cost components like RFID modules and microcontrollers.

4.1 Testing

Test 1: To detect and validate a registered RFID tag

Table 5: To detect and validate a registered RFID tag

Test	**1**
Objective	To verify that the system correctly detects and logs a valid RFID card.
Action	• Compile the program. • Swipe a registered RFID card over the reader. • Observe LCD and database.
Expected Result	Systems displays “Access Granted” and logs student ID with actual time.
Actual Result	Displayed “Access Granted” and entry successfully logged.
Conclusion	The test was successful.



03_ESP32_RFID_Google_Spreadsheet_Attendance | Arduino IDE 2.3.6
File Edit Sketch Tools Help
ESP32 Dev Module
03_ESP32_RFID_Google_Spreadsheet_Attendance.ino
Serial Monitor
Output
Message (Enter to send message to 'ESP32 Dev Module' on 'COM6')
New Line
115200 baud
Ln 377, Col 1
ESP32 Dev Module on COM6
3:03 AM
5/6/2025

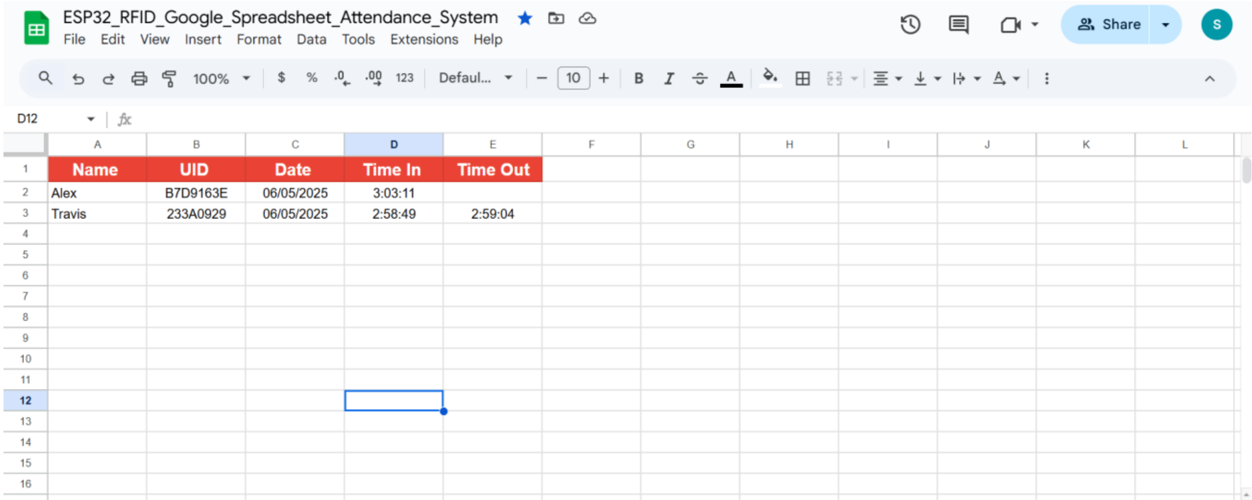
```
------------
Connecting to virus.exe
...
WiFi connected
------------

-------------
Sending request to Google Sheets...
URL : https://script.google.com/macros/s/AKfycbyd2WvK2Ua7Awx4JwkRY1JEmu8UVaqyZkuUw33uiFA4453mkFbCIlOkl7XhcpSsIyhEpA/exec?sts=atc&uid=233A0929
HTTP Status Code : 200
Payload : OK,TI_Successful,Travis ,06/05/2025,02:58:49
-------------

-------------
Sending request to Google Sheets...
URL : https://script.google.com/macros/s/AKfycbyd2WvK2Ua7Awx4JwkRY1JEmu8UVaqyZkuUw33uiFA4453mkFbCIlOkl7XhcpSsIyhEpA/exec?sts=atc&uid=233A0929
HTTP Status Code : 200
Payload : OK,TO_Successful,Travis ,06/05/2025,2:58:49,02:59:04
-------------

-------------
WIFI mode : STA
```

Figure 26 To detect and validate a registered RFID tag

Test 2: To display attendance logs

Table 6: To display attendance logs

Test	**2**
Objective	To confirm that the logged attendance data is visible in database or sheet.
Action	• Open the database or attendance sheet. • Navigate to the attendance records section. • View the most recent scan entry.
Expected Result	Attendance record is displayed with correct name, ID, date and time.
Actual Result	Attendance details correctly shown in database or sheet.
Conclusion	The test was successful.



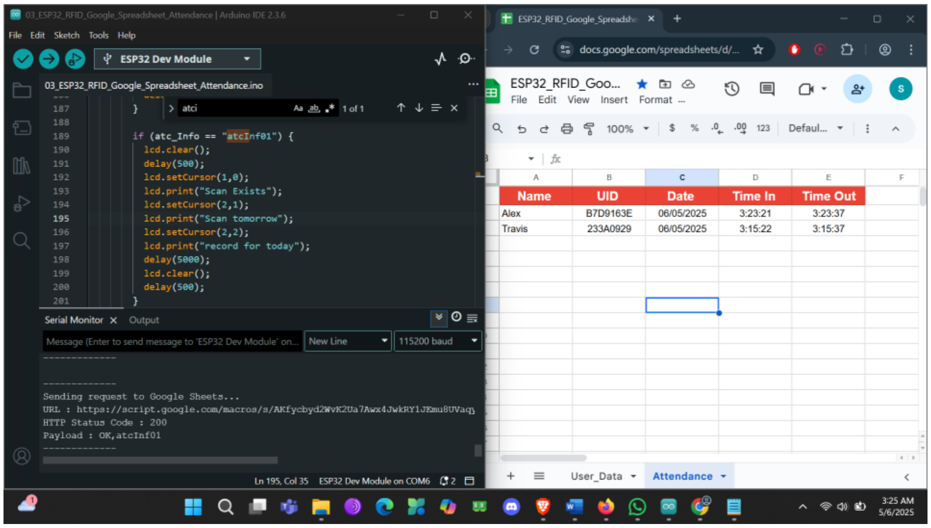
Name	UID	Date	Time In	Time Out
Alex	B7D9163E	06/05/2025	3:03:11	
Travis	233A0929	06/05/2025	2:58:49	2:59:04

Figure 27 Displaying the attendance logs

Test 3: To ensure the system records only two entries per user per day

Table 7: To ensure the system records only two entries per user per day

Test	**3**
Objective	To ensure the system records only two entries per user per day
Action	Scanned the same RFID card twice: once in the morning, once in the evening
Expected Result	First scan logs "Time In", second logs "Time Out", no extra entries allowed
Actual Result	System recorded Time In and Time Out correctly, ignored further scans
Conclusion	The test was successful.


03_ESP32_RFID_Google_Spreadsheet_Attendance | Arduino IDE 2.3.6
File Edit Sketch Tools Help
ESP32 Dev Module
03_ESP32_RFID_Google_Spreadsheet_Attendance.ino
atci
1 of 1
Serial Monitor
Output
Message (Enter to send message to 'ESP32 Dev Module' on...
New Line
115200 baud
Ln 195, Col 35
ESP32 Dev Module on COM6
ESP32_RFID_Goo...
File Edit View Insert Format
User_Data
Attendance

```
    }

    if (atc_Info == "atcInf01") {
      lcd.clear();
      delay(500);
      lcd.setCursor(1,0);
      lcd.print("Scan Exists");
      lcd.setCursor(2,1);
      lcd.print("Scan tomorrow");
      lcd.setCursor(2,2);
      lcd.print("record for today");
      delay(5000);
      lcd.clear();
      delay(500);
    }
```

```
-------------

-------------
Sending request to Google Sheets...
URL : https://script.google.com/macros/s/AKfycbyd2WvK2Ua7Awx4JwkRY1JEmu8UVaq
HTTP Status Code : 200
Payload : OK,atcInf01
-------------
```

Name	UID	Date	Time In	Time Out
Alex	B7D9163E	06/05/2025	3:23:21	3:23:37
Travis	233A0929	06/05/2025	3:15:22	3:15:37

Figure 28 Ensuring the system records only two entries per user per day

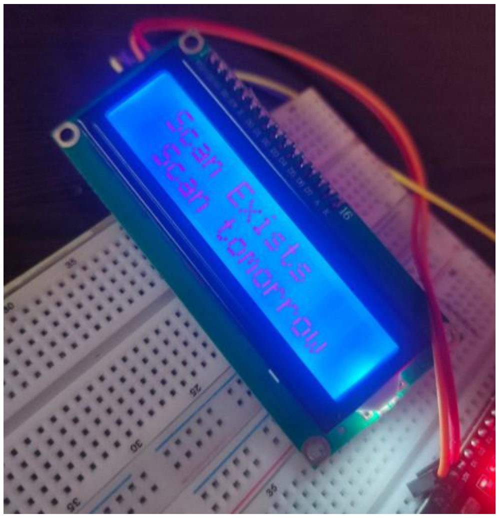


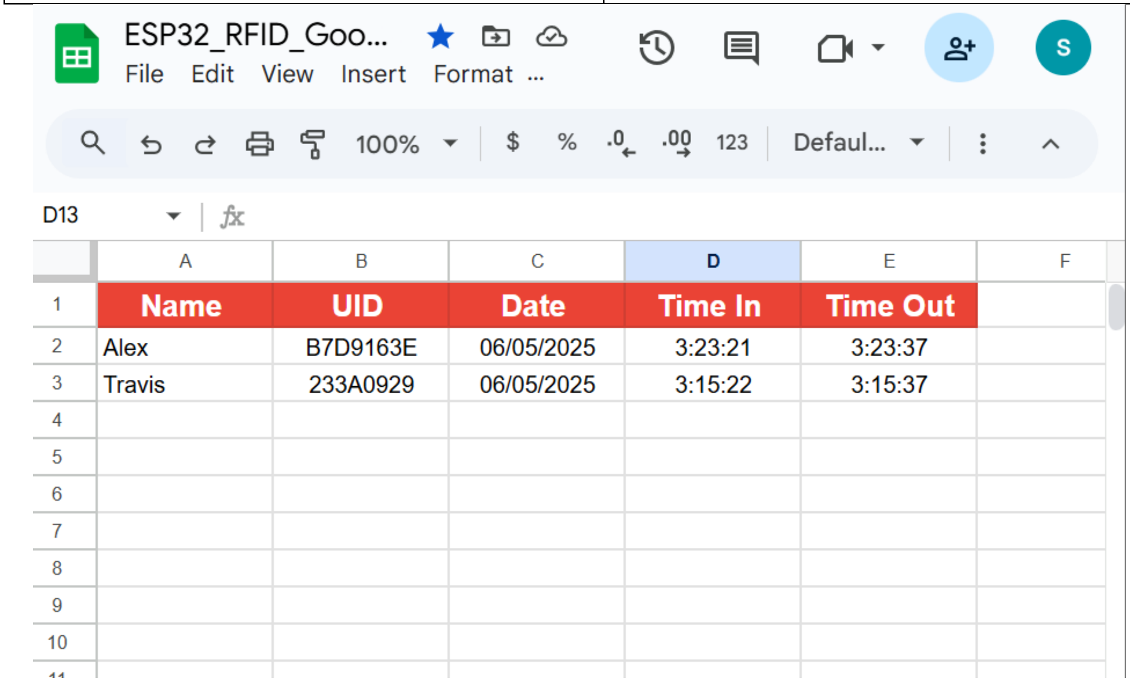
Figure 29 Ensuring the system records only two entries per user per day - Output

Test 4: To record both Time In and Time Out entries for the same user.

Table 8: To record both Time In and Time Out entries for the same user.

Test	**4**
Objective	To verify that the system accurately records both “Time In” and “Time Out” for user using the same RFID Cards.

Action	• Swipe a valid card to record “Time In” entry. • Swipe the same tag again after brief interval to record “Time Out” entry.
Expected Result	Two distinct entries should appear in the database or attendance sheet, one marked as “Time In” and second as “Time Out”.
Actual Result	The system successfully captured the initial swipe as “Time In” and the second swipe as “Time Out”.
Conclusion	The test was successful.



Name	UID	Date	Time In	Time Out
Alex	B7D9163E	06/05/2025	3:23:21	3:23:37
Travis	233A0929	06/05/2025	3:15:22	3:15:37

Figure 30 Recording both Time In and Time Out entries for the same user

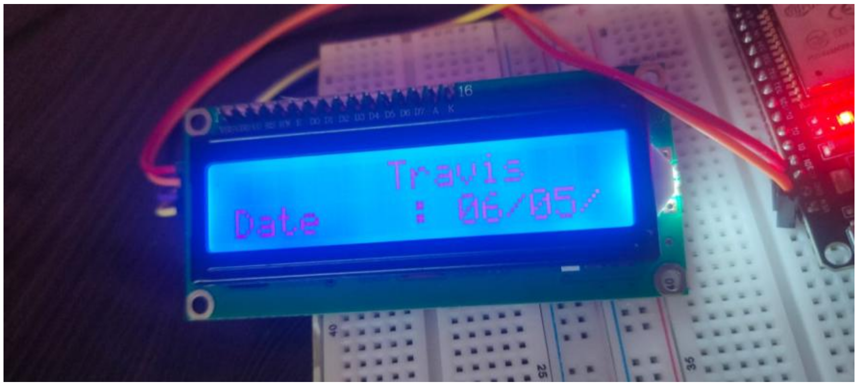


Figure 31 Recording both Time In and Time Out entries for the same user - Displayed

Test 5: To verify RFID detection output on the monitor.

Table 9: To verify RFID detection output on the monitor.

Test	**5**
Objective	To ensure that RFID tag detection is properly reflected on the output monitor during system operation.
Action	• Execute the system code. • Swipe a valid RFID card. • Observe output monitor for confirmation. • Verify the corresponding entry in the attendance sheet.
Expected Result	The RFID detection message should be displayed on the monitor and the entry should be logged in the database.
Actual Result	The RFID tag was successfully recorded in the database, the confirmation message was not seen on the output monitor.
Conclusion	The test was unsuccessful.

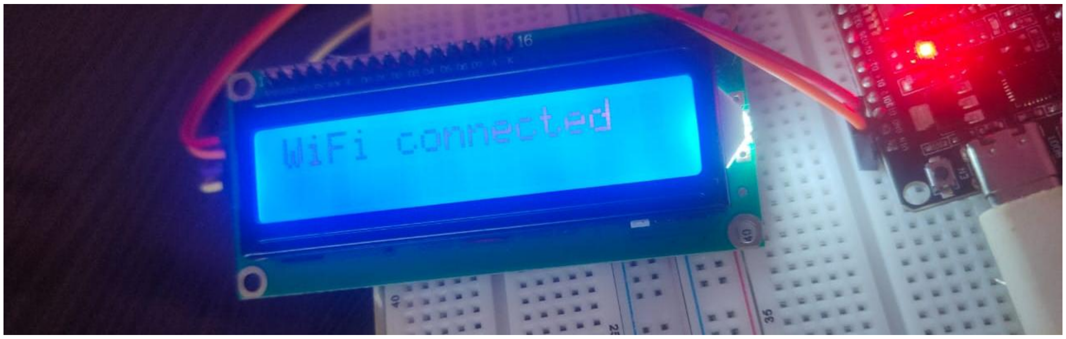


Figure 32 Verify RFID detection output on the monitor - I

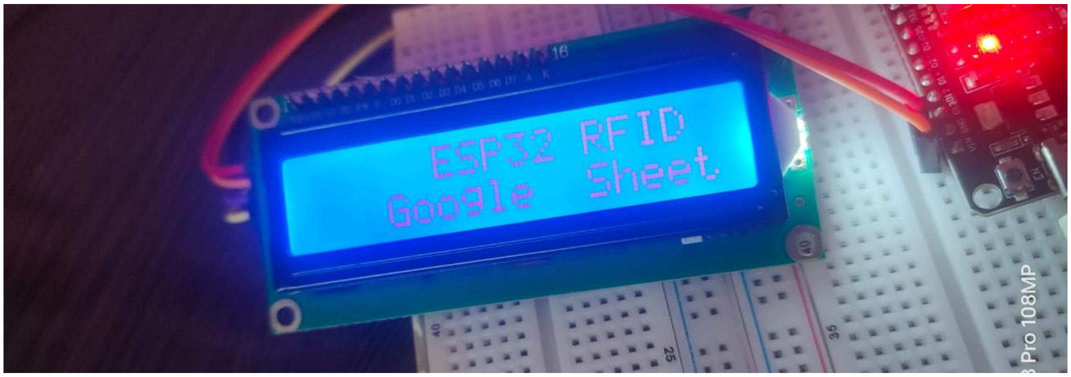


Figure 33 Verify RFID detection output on the monitor - II

5. Future Work

Improved Time and Resources Utilities: By utilizing this system, classrooms, staff, and time can be managed more efficiently in organizations. Automate attendance in hours per week with minimal manual effort.

No More Manual Marking: There will be no teachers calling names. Students simply tap their card and attendance is recorded immediately. This saves time, reduces mistakes, and simplifies reporting.

Smart Analysis Using AI: AI can be used to track patterns of attendance, identify students who are starting to skip a lot of class and even make proactive recommendations for how they might improve. AI can also automatically compile attendance reports and notify teachers or parents.

Mobile App Integration: Dedicated apps can also allow students to check their attendance status and be alerted and reminded. Teachers can access live data with the app, as can parents keeping up with their child’s attendance too.

Better Security with Biometrics: In order to prevent misuse (proxy attendance), RFID could be used in combination with fingerprint or face recognition. This prevents the wrong student from marking his or her attendance.

Smart Campus System: Such system can be communicated to the library, labs, hostels and other reaches. It is able to control access and monitor usage, which makes the campus more interrelated and smarter.

Flexible Rules for Attendance: The system will accommodate late entries, half days and additional activities such as seminars and sports depending on the needs for each institution.

Live Monitoring and Reports: Real-time dashboards can help admin staff track attendance. Reports will be produced and sent automatically through email or app.

Support for Learning Activities: Attendance can be associated with assignments, additional classes, or study groups. Students that attend more and participate can be rewarded.

6. Conclusion

Upgrading the attendance has been undergoing a manual checking process to implementing an RFID based attendance marking system, which represents modernization and automation for marking student attendance. When contrasted to manual tracking being inefficient prone to errors and consuming time RFID attendance technology offers being more accurate, faster seamlessly delivered solution. Seamless offering recording through the use of RFID readers where attendance is instantly registered without any oversight requiring manual input automatic attendance interfaces and significant proven participator.

The system provides users and administrators with an uncomplicated and secure experience while improving operational efficiency. Institutions can quickly access attendance records resulting from centralized data management, enabling informed decision-making along with detailed performance reporting. In addition, the adaptability of the RFID-based systems makes them long-term economic solutions in numerous environments, including educational, office, and industrial settings.

Looking toward the future, there is significant potential for enhancing the system's capabilities. Upgrades such as mobile application support for up-to-the-minute tracking and remote data access through the cloud, as well as biometric verification to strengthen security, could be implemented. Used with comprehensive "smart" campus or "smart" workplace infrastructures, RFID attendance systems can augment the development of automated and intelligent environments.

In summary, the use of RFID technology in attendance management is progressive as it strives to meet the reliability standards while replacing older systems, hence devices based on this technology will lead to further innovations and advancements in digitization.

7. References

Adams, S., 2025. *the knowledge academy*. [Online]
Available at: https://www.theknowledgeacademy.com/blog/what-is-microsoft-word/

Almansor, M. a. A. F., 2021. Student Attendance Using RFID System. *Journal of University of Shanghai for Science and Technology*, p. 264.

Anker, 2024. *Anker.* [Online]
Available at: https://www.anker.com/blogs/cables/how-to-identify-different-types-of-usb-cables-a-brief-guide

Anon., 2025. [Online]
Available at: https://www.tinyosshop.com/index.php?route=product/product&product_id=827

Anon., 2025. [Online]
Available at: https://www.indiamart.com/proddetail/rfid-cards-23324782755.html

Anon., 2025. [Online]
Available at: https://www.made-in-china.com/price/cellphone-charger-cable-price.html

azadtechhub, 2024. *azadtechhub.* [Online]
Available at: https://azadtechhub.com/esp32-devkit-v1-a-comprehensive-guide/

chai, W., 2025. *techtarget.* [Online]
Available at: https://www.techtarget.com/whatis/definition/Google-Spreadsheets

elecena, 2025. [Online]
Available at: https://elecena.pl/product/22785820/i2c-20x4-arduino-lcd-display-module

fittyclub, 2025. *fittyclub.* [Online]
Available at: https://www.fity.club/lists/suggestions/draw-io-web/

hope, C., 2024. *computer hope.* [Online]
Available at: https://www.computerhope.com/jargon/d/drawio.htm

inventkart, 2025. [Online]
Available at: https://inventkart.com/product/nodemcu-esp8266-wifi-development-iot-board/

kaczor, c., 2025. *Camcode.* [Online]
Available at: https://www.camcode.com/blog/what-are-rfid-tags/

katmatic, 2025. *Smart School Attendance System | Transforming Education in Nepal.* [Online]
Available at: https://www.katmatic.com/smart-attendance-system-katmatic/

Kumar, P. K. R. S. H. &. B. S., 2024. IoT Based RFID Attendance System. *International Journal of Scientific Research & Engineering Trends,* p. 304.

logosworld, 2024. *logosworld.* [Online]
Available at: https://logos-world.net/microsoft-word-logo/

magdey, k., 2025. *Deepbluembedded.* [Online]
Available at: https://deepbluembedded.com/arduino-lcd-20x4-i2c-code-example-tutorial/

nettigo, 2025. [Online]
Available at: https://nettigo.eu/products/jumper-wires-m-m-40-pcs-20-cm

petergong, 2025. [Online]
Available at: https://sfxpcb.com/how-to-use-a-breadboard-and-how-it-works/

Sanketh, R. S., 2020. *Medium.* [Online]
Available at: https://medium.com/autonomous-robotics/an-introduction-to-rfid-dc6228767691

Schulze, J., 2025. *What Is the Internet of Things (IoT)? With Examples.* [Online]
Available at: https://www.coursera.org/articles/internet-of-things

soldered, 2023. *WHAT IS FRITZING, HOW DOES IT WORK AND HOW TO USE IT?.* [Online]
Available at: https://soldered.com/learn/what-is-fritzing-how-does-it-work-and-how-to-use-it/
[Accessed 06 05 2025].

stickpng, 2025. *stickpng.* [Online]
Available at: https://www.stickpng.com/img/icons-logos-emojis/iconic-brands/google-sheets-logo-and-symbol

WatElectronic, 2021. *Watelectronics.* [Online]
Available at: https://www.watelectronics.com/breadboard-construction-types-working/

wiltronics, 2022. *wiltronics.* [Online]
Available at: https://www.wiltronics.com.au/wiltronics-knowledge-base/what-are-jumper-wires/

zemfiraaksenova, 2019. *blogspot.* [Online]
Available at: https://zemfiraaksenova.blogspot.com/2019/11/rfid-rc522-rf-ic-card-reader-sensor.html

zheng, 2025. *rfidcard.* [Online]
Available at: https://www.rfidcard.com/what-is-rfid-card/

Uteh Str. (2023, November 7). Arduino IDE + ESP32 + RFID RC522 + Google Sheets | RFID Simple Attendance System ESP32 Google Sheets [Video]. YouTube. https://www.youtube.com/watch?v=r0Ndu2VYR2Y

8. Appendix

8.1 Working Code

Test_RFID-RC522_Serial

```
#include <SPI.h>

#include <MFRC522.h>

#define SS_PIN  5

#define RST_PIN 4


// Variable to read data from RFID-RC522.

int readsuccess;

char str[32] = "";

String UID_Result = "";


// Create MFRC522 object as "mfrc522" and set SS/SDA PIN and Reset PIN.

MFRC522 mfrc522(SS_PIN, RST_PIN);


//________________________________________________________________________________
____getUID()

// Subroutine to obtain UID/ID when RFID card or RFID keychain is tapped to RFID-RC522
module.

int getUID() {

  if(!mfrc522.PICC_IsNewCardPresent()) {

    return 0;

  }

  if(!mfrc522.PICC_ReadCardSerial()) {

    return 0;

  }
  byteArray_to_string(mfrc522.uid.uidByte, mfrc522.uid.size, str);
  UID_Result = str;

  mfrc522.PICC_HaltA();
  mfrc522.PCD_StopCrypto1();

  return 1;
}
//________________________________________________________________________________

//________________________________________________________________________________byteArray_to_string()
void byteArray_to_string(byte array[], unsigned int len, char buffer[]) {
  for (unsigned int i = 0; i < len; i++) {
    byte nib1 = (array[i] >> 4) & 0x0F;
    byte nib2 = (array[i] >> 0) & 0x0F;
    buffer[i*2+0] = nib1  < 0xA ? '0' + nib1  : 'A' + nib1  - 0xA;
    buffer[i*2+1] = nib2  < 0xA ? '0' + nib2  : 'A' + nib2  - 0xA;
  }
  buffer[len*2] = '\0';
}
//________________________________________________________________________________

//________________________________________________________________________________VOID SETUP()
void setup(){
  // put your setup code here, to run once:
```

```
  Serial.begin(115200);
  Serial.println();
  delay(1000);

  // Init SPI bus.
  SPI.begin();
  // Init MFRC522.
  mfrc522.PCD_Init();

  delay(1000);

  Serial.println();
  Serial.println("Please tap your card or key chain to the RFID-RC522 module.");
}
//_____________________________________________________________________________
_____

//_____________________________________________________________________________
_____VOID LOOP()
void loop(){
  // put your main code here, to run repeatedly:

  readsuccess = getUID();

  if(readsuccess){
    Serial.println();
    Serial.print("UID : ");
    Serial.println(UID_Result);
    delay(2000);
  }
  delay(10);
}
```

Test_RFID-RC522_LCD-20x4_Button

```
#include <LiquidCrystal_I2C.h>

#include <SPI.h>

#include <MFRC522.h>

//------------------------------------


// Defines SS/SDA PIN and Reset PIN for RFID-RC522.

#define SS_PIN  5

#define RST_PIN 4


// Defines the button PIN.

#define BTN_PIN 15


// LCD configuration

int lcdColumns = 20;

int lcdRows = 4;


// Variables

int readsuccess;

char str[32] = "";

String UID_Result = "--------";


// LCD object (address: 0x27, 20 cols, 4 rows)

LiquidCrystal_I2C lcd(0x27, lcdColumns, lcdRows);
```

```
// RFID object
MFRC522 mfrc522(SS_PIN, RST_PIN);

//____________________________________________________________________________
____byteArray_to_string()
void byteArray_to_string(byte array[], unsigned int len, char buffer[]) {
  for (unsigned int i = 0; i < len; i++) {
    byte nib1 = (array[i] >> 4) & 0x0F;
    byte nib2 = (array[i] >> 0) & 0x0F;
    buffer[i*2+0] = nib1  < 0xA ? '0' + nib1  : 'A' + nib1  - 0xA;
    buffer[i*2+1] = nib2  < 0xA ? '0' + nib2  : 'A' + nib2  - 0xA;
  }
  buffer[len*2] = '\0';
}
//____________________________________________________________________________
____

//____________________________________________________________________________
____getUID()
int getUID() {
  if(!mfrc522.PICC_IsNewCardPresent()) {
    return 0;
  }
  if(!mfrc522.PICC_ReadCardSerial()) {
    return 0;
  }

  byteArray_to_string(mfrc522.uid.uidByte, mfrc522.uid.size, str);
  UID_Result = str;
  Serial.println();
  Serial.print("UID : ");
  Serial.println(UID_Result);

  mfrc522.PICC_HaltA();
  mfrc522.PCD_StopCrypto1();

  return 1;
}
//______________________________________________________________________________
_____

//______________________________________________________________________________
_____setup()
void setup(){
  // Start serial
  Serial.begin(115200);
  delay(1000);
  Serial.println();
  Serial.println("Please tap your card or keychain...");

  // Setup button
  pinMode(BTN_PIN, INPUT_PULLUP);

  // Init LCD
  lcd.init();
  lcd.backlight();
  lcd.clear();
  // Init SPI & RFID
  SPI.begin();
  mfrc522.PCD_Init();
  delay(1000);
}
//______________________________________________________________________________
____

//______________________________________________________________________________
____loop()
void loop(){
  lcd.setCursor(0, 0);
  lcd.print("Tap your card/key  ");

  lcd.setCursor(0, 1);
  lcd.print("or keychain        ");

  lcd.setCursor(0, 2);
  lcd.print("UID: ");
  lcd.print(UID_Result);
  lcd.print("        "); // Clear extra space

  lcd.setCursor(0, 3);
  lcd.print("BTN: ");
  lcd.print(digitalRead(BTN_PIN));
  lcd.print("          "); // Clear old text

  readsuccess = getUID();
  if(readsuccess){
    delay(2000);
  }

  delay(100);
}
```

ESP32_RFID_Google_Spreadsheet_Attendance

```
#include <LiquidCrystal_I2C.h>
#include <SPI.h>
#include <MFRC522.h>
#include "WiFi.h"
#include <HTTPClient.h>
//----------------------------------------

// Defines SS/SDA PIN and Reset PIN for RFID-RC522.
#define SS_PIN  5
#define RST_PIN 4

// Defines the button PIN.
#define BTN_PIN 15

//----------------------------------------SSID and PASSWORD of your WiFi network.
const char* ssid = "virus.exe";  //--> Your wifi name
const char* password = "virus.exe"; //--> Your wifi password
//----------------------------------------

// Google script Web_App_URL.
```

```
String Web_App_URL =
"https://script.google.com/macros/s/AKfycbyd2WvK2Ua7Awx4JwkRY1JEmu8UVaqyZkuUw33uiFA4453mkFbCIlOkl7XhcpSsIyhEpA/exec";

String reg_Info = "";

String atc_Info = "";
String atc_Name = "";
String atc_Date = "";
String atc_Time_In = "";
String atc_Time_Out = "";

// Variables for the number of columns and rows on the LCD.
int lcdColumns = 20;
int lcdRows = 4;

// Variable to read data from RFID-RC522.
int readsuccess;
char str[32] = "";
String UID_Result = "--------";

String modes = "atc";

// Create LiquidCrystal_I2C object as "lcd" and set the LCD I2C address to 0x27 and set the LCD configuration to 20x4.
// In general, the address of a 20x4 I2C LCD is "0x27".
// However, if the address "0x27" doesn't work, you can find out the address with "i2c_scanner". Look here : https://playground.arduino.cc/Main/I2cScanner/
LiquidCrystal_I2C lcd(0x27, lcdColumns, lcdRows);  // (lcd_address, lcd_Columns, lcd_Rows)
```

```
// Create MFRC522 object as "mfrc522" and set SS/SDA PIN and Reset PIN.
MFRC522 mfrc522(SS_PIN, RST_PIN);  //--> Create MFRC522 instance.

//____________________________________________________________________________
____http_Req()
// Subroutine for sending HTTP requests to Google Sheets.
void http_Req(String str_modes, String str_uid) {
  if (WiFi.status() == WL_CONNECTED) {
    String http_req_url = "";

    //----------------------------------------Create links to make HTTP requests to Google Sheets.
    if (str_modes == "atc") {
      http_req_url  = Web_App_URL + "?sts=atc";
      http_req_url += "&uid=" + str_uid;
    }
    if (str_modes == "reg") {
      http_req_url = Web_App_URL + "?sts=reg";
      http_req_url += "&uid=" + str_uid;
    }
    //----------------------------------------

    //----------------------------------------Sending HTTP requests to Google Sheets.
    Serial.println();
    Serial.println("-------------");
    Serial.println("Sending request to Google Sheets...");
    Serial.print("URL : ");
    Serial.println(http_req_url);

    // Create an HTTPClient object as "http".
```

```
HTTPClient http;

// HTTP GET Request.
http.begin(http_req_url.c_str());
http.setFollowRedirects(HTTPC_STRICT_FOLLOW_REDIRECTS);

// Gets the HTTP status code.
int httpCode = http.GET();
Serial.print("HTTP Status Code : ");
Serial.println(httpCode);

// Getting response from google sheet.
String payload;
if (httpCode > 0) {
  payload = http.getString();
  Serial.println("Payload : " + payload);
}

Serial.println("-------------");
http.end();
//----------------------------------------

////////////////////////////////////////////////////////////////////////////////////////////////////
// Example :                                                                  //
// Sending an http request to fill in "Time In" attendance.                  //
// User data :                                                                //
// - Name : Adam                                                              //
// - UID  : A01                                                               //
```

```
    // So the payload received if the http request is successful and the parameters are correct is as below : //
    // OK,Adam,29/10/2023,08:30:00 ---> Status,Name,Date,Time_In                                    //
    //                                                                                              //
    // So, if you want to retrieve "Status", then getValue(payload, ',', 0);                        //
    // String sts_Res = getValue(payload, ',', 0);                                                  //
    // So the value of sts_Res is "OK".                                                             //
    ////////////////////////////////////////////////////////////////////////////////////////////////////////////

    String sts_Res = getValue(payload, ',', 0);

    //----------------------------------------Conditions that are executed are based on the payload response from Google Sheets (the payload response is set in Google Apps Script).
    if (sts_Res == "OK") {
      //..................
      if (str_modes == "atc") {
        atc_Info = getValue(payload, ',', 1);

        if (atc_Info == "TI_Successful") {
          atc_Name = getValue(payload, ',', 2);
          atc_Date = getValue(payload, ',', 3);
          atc_Time_In = getValue(payload, ',', 4);

          //::::::::::::::::::Create a position value for displaying "Name" on the LCD so that it is centered.
          int name_Lenght = atc_Name.length();
          int pos = 0;
          if (name_Lenght > 0 && name_Lenght <= lcdColumns) {
            pos = map(name_Lenght, 1, lcdColumns, 0, (lcdColumns / 2) - 1);
    pos = ((lcdColumns / 2) - 1) - pos;
  } else if (name_Lenght > lcdColumns) {
    atc_Name = atc_Name.substring(0, lcdColumns);
  }
  //::::::::::::::::::

  lcd.clear();
  delay(500);
  lcd.setCursor(pos,0);
  lcd.print(atc_Name);
  lcd.setCursor(0,1);
  lcd.print("Date    : ");
  lcd.print(atc_Date);
  lcd.setCursor(0,2);
  lcd.print("Time IN :   ");
  lcd.print(atc_Time_In);
  lcd.setCursor(0,3);
  lcd.print("Time Out:   ");
  delay(5000);
  lcd.clear();
  delay(500);
}

if (atc_Info == "TO_Successful") {
  atc_Name = getValue(payload, ',', 2);
  atc_Date = getValue(payload, ',', 3);
  atc_Time_In = getValue(payload, ',', 4);
  atc_Time_Out = getValue(payload, ',', 5);
```

```
//::::::::::::::::::Create a position value for displaying "Name" on the LCD so that it is centered.
int name_Lenght = atc_Name.length();
int pos = 0;
if (name_Lenght > 0 && name_Lenght <= lcdColumns) {
  pos = map(name_Lenght, 1, lcdColumns, 0, (lcdColumns / 2) - 1);
  pos = ((lcdColumns / 2) - 1) - pos;
} else if (name_Lenght > lcdColumns) {
  atc_Name = atc_Name.substring(0, lcdColumns);
}
//::::::::::::::::::

lcd.clear();
delay(500);
lcd.setCursor(pos,0);
lcd.print(atc_Name);
lcd.setCursor(0,1);
lcd.print("Date    : ");
lcd.print(atc_Date);
lcd.setCursor(0,2);
lcd.print("Time IN :   ");
lcd.print(atc_Time_In);
lcd.setCursor(0,3);
lcd.print("Time Out:   ");
lcd.print(atc_Time_Out);
delay(5000);
lcd.clear();
delay(500);
```

```
}

if (atc_Info == "atcInf01") {
  lcd.clear();
  delay(500);
  lcd.setCursor(1,0);
  lcd.print("You have completed");
  lcd.setCursor(2,1);
  lcd.print("your  attendance");
  lcd.setCursor(2,2);
  lcd.print("record for today");
  delay(5000);
  lcd.clear();
  delay(500);
}

if (atc_Info == "atcErr01") {
  lcd.clear();
  delay(500);
  lcd.setCursor(6,0);
  lcd.print("Error !");
  lcd.setCursor(4,1);
  lcd.print("Your card or");
  lcd.setCursor(4,2);
  lcd.print("key chain is");
  lcd.setCursor(3,3);
  lcd.print("not registered");
  delay(5000);
```

```
    lcd.clear();
    delay(500);
  }

  atc_Info = "";
  atc_Name = "";
  atc_Date = "";
  atc_Time_In = "";
  atc_Time_Out = "";
}
//..................

//..................
if (str_modes == "reg") {
  reg_Info = getValue(payload, ',', 1);

  if (reg_Info == "R_Successful") {
    lcd.clear();
    delay(500);
    lcd.setCursor(2,0);
    lcd.print("The UID of your");
    lcd.setCursor(0,1);
    lcd.print("card or keychain has");
    lcd.setCursor(1,2);
    lcd.print("been successfully");
    lcd.setCursor(6,3);
    lcd.print("uploaded");
    delay(5000);
```

```
        lcd.clear();
        delay(500);
      }

      if (reg_Info == "regErr01") {
        lcd.clear();
        delay(500);
        lcd.setCursor(6,0);
        lcd.print("Error !");
        lcd.setCursor(0,1);
        lcd.print("The UID of your card");
        lcd.setCursor(0,2);
        lcd.print("or keychain has been");
        lcd.setCursor(5,3);
        lcd.print("registered");
        delay(5000);
        lcd.clear();
        delay(500);
      }

      reg_Info = "";
    }
    //..................
  }
  //--------------------------------------
} else {
  lcd.clear();
  delay(500);
```

```
    lcd.setCursor(6,0);
    lcd.print("Error !");
    lcd.setCursor(1,1);
    lcd.print("WiFi disconnected");
    delay(3000);
    lcd.clear();
    delay(500);
  }
}
//____________________________________________________________________________
____

//____________________________________________________________________________
____getValue()
// String function to process the data (Split String).
// I got this from : https://www.electroniclinic.com/reyax-lora-based-multiple-sensors-monitoring-using-arduino/
String getValue(String data, char separator, int index) {
  int found = 0;
  int strIndex[] = { 0, -1 };
  int maxIndex = data.length() - 1;

  for (int i = 0; i <= maxIndex && found <= index; i++) {
    if (data.charAt(i) == separator || i == maxIndex) {
      found++;
      strIndex[0] = strIndex[1] + 1;
      strIndex[1] = (i == maxIndex) ? i+1 : i;
    }
  }
  return found > index ? data.substring(strIndex[0], strIndex[1]) : "";
}
//_________________________________________________________________________________________


//_________________________________________________________________________________________getUID()
// Subroutine to obtain UID/ID when RFID card or RFID keychain is tapped to RFID-RC522 module.
int getUID() {
  if(!mfrc522.PICC_IsNewCardPresent()) {
    return 0;
  }
  if(!mfrc522.PICC_ReadCardSerial()) {
    return 0;
  }

  byteArray_to_string(mfrc522.uid.uidByte, mfrc522.uid.size, str);
  UID_Result = str;

  mfrc522.PICC_HaltA();
  mfrc522.PCD_StopCrypto1();

  return 1;
}

  Serial.begin(115200);
  Serial.println();
  delay(1000);
```

```
pinMode(BTN_PIN, INPUT_PULLUP);

// Initialize LCD.
lcd.init();
// turn on LCD backlight.
lcd.backlight();

lcd.clear();

delay(500);

// Init SPI bus.
SPI.begin();
// Init MFRC522.
mfrc522.PCD_Init();

delay(500);

lcd.setCursor(5,0);
lcd.print("ESP32 RFID");
lcd.setCursor(3,1);
lcd.print("Google  Sheets");
lcd.setCursor(1,2);
lcd.print("Attendance  System");
lcd.setCursor(4,3);
lcd.print("by  UTEH STR");
delay(3000);
```

```
lcd.clear();

//----------------------------------------Set Wifi to STA mode
Serial.println();
Serial.println("-------------");
Serial.println("WIFI mode : STA");
WiFi.mode(WIFI_STA);
Serial.println("-------------");
//----------------------------------------

//----------------------------------------Connect to Wi-Fi (STA).
Serial.println();
Serial.println("------------");
Serial.print("Connecting to ");
Serial.println(ssid);d
WiFi.begin(ssid, password);

int connecting_process_timed_out = 20; //--> 20 = 20 seconds.
connecting_process_timed_out = connecting_process_timed_out * 2;
while (WiFi.status() != WL_CONNECTED) {
  Serial.print(".");

  lcd.setCursor(0,0);
  lcd.print("Connecting to SSID");
  delay(250);

  lcd.clear();
  delay(250);
```

```
    if (connecting_process_timed_out > 0) connecting_process_timed_out--;
    if (connecting_process_timed_out == 0) {
      delay(1000);
      ESP.restart();
    }
  }

  Serial.println();
  Serial.println("WiFi connected");
  Serial.println("------------");

  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("WiFi connected");
  delay(2000);
  //::::::::::::::::::
  //----------------------------------------

  lcd.clear();
  delay(500);
}
//__________________________________________________________________________________________
_____

//__________________________________________________________________________________________
_____VOID LOOP()
void loop(){
  // put your main code here, to run repeatedly:
```

```
//----------------------------------------Switches modes when the button is pressed.
// modes = "reg" means the mode for new user registration.
// modes = "atc" means the mode for filling in attendance (Time In and Time Out).

int BTN_State = digitalRead(BTN_PIN);

if (BTN_State == LOW) {
  lcd.clear();

  if (modes == "atc") {
    modes = "reg";
  } else if (modes == "reg") {
    modes = "atc";
  }

  delay(500);
}
//----------------------------------------

// Detect if reading the UID from the card or keychain was successful.
readsuccess = getUID();

//----------------------------------------Conditions that are executed if modes == "atc".
if (modes == "atc") {
  lcd.setCursor(5,0);
  lcd.print("ATTENDANCE");
  lcd.setCursor(0,1);
  lcd.print("");
  lcd.setCursor(0,2);
  lcd.print("Please tap your card");
  lcd.setCursor(4,3);
  lcd.print("or key chain");

  if (readsuccess){
    lcd.clear();
    delay(500);
    lcd.setCursor(4,0);
    lcd.print("Getting  UID");
    lcd.setCursor(4,1);
    lcd.print("Successfully");
    lcd.setCursor(0,2);
    lcd.print("");
    lcd.setCursor(3,3);
    lcd.print("Please wait...");
    delay(1000);

    http_Req(modes, UID_Result);
  }
}
//--------------------------------------Conditions that are executed if modes == "reg".
if (modes == "reg") {
  lcd.setCursor(4,0);
  lcd.print("REGISTRATION");
  lcd.setCursor(0,1);
  lcd.print("");
```

```
    lcd.setCursor(0,2);
    lcd.print("Please tap your card");
    lcd.setCursor(4,3);
    lcd.print("or key chain");

    if (readsuccess){
      lcd.clear();
      delay(500);
      lcd.setCursor(0,0);
      lcd.print("Getting  UID");
      lcd.setCursor(0,1);
      lcd.print("Successfully");
      lcd.setCursor(0,2);
      lcd.print("UID : ");
      lcd.print(UID_Result);
      lcd.setCursor(0,3);
      lcd.print("Please wait...");
      delay(1000);

      http_Req(modes, UID_Result);
    }
  }
  //---------------------------------------

  delay(10);
}
```

Google APP Script

```
//______________________________________________________________________________
____doGet()
function doGet(e) {
  Logger.log(JSON.stringify(e));
  var result = 'OK';
  if (e.parameter == 'undefined') {
    result = 'No_Parameters';
  }
  else {
    var sheet_id = '178VEs65UjaArbcUAdiMjSgp-2Cv8aEg92v2ZdVQuVV4';       // Spreadsheet ID.
    var sheet_UD = 'User_Data';  // Sheet name for user data.
    var sheet_AT = 'Attendance';  // Sheet name for attendance.

    var sheet_open = SpreadsheetApp.openById(sheet_id);
    var sheet_user_data = sheet_open.getSheetByName(sheet_UD);
    var sheet_attendence = sheet_open.getSheetByName(sheet_AT);

    // sts_val is a variable to hold the status sent by ESP32.
    // sts_val will contain "reg" or "atc".
    // "reg" = new user registration.
    // "atc" = attendance (time in and time out).
    var sts_val = "";
```

```
// uid_val is a variable to hold the UID of the RFID card or keychain sent by the ESP32.
var uid_val = "";

// UID storage column.
var uid_column = "B";

// Variable to retrieve the "Time In" value from the sheet.
var TI_val = "";
// Variable to retrieve the "Date" value from the sheet.
var Date_val = "";

//---------------------------------------Retrieves the value of the parameter sent by the ESP32.
for (var param in e.parameter) {
  Logger.log('In for loop, param=' + param);
  var value = stripQuotes(e.parameter[param]);
  Logger.log(param + ':' + e.parameter[param]);
  switch (param) {
    case 'sts':
      sts_val = value;
      break;

    case 'uid':
      uid_val = value;
      break;

    default:
      // result += ",unsupported_parameter";
  }
}
//----------------------------------------
//----------------------------------------Conditions for registering new users.
if (sts_val == 'reg') {
  var check_new_UID = checkUID(sheet_id, sheet_UD, 2, uid_val);

  // Conditions when the UID has been registered. Then registration was cancelled.
  if (check_new_UID == true) {
    result += ",regErr01"; // Err_01 = UID is already registered.

    // Sends response payload to ESP32.
    return ContentService.createTextOutput(result);
  }

  // Writes the new user's UID to the "user data" sheet.
  var getLastRowUIDCol = findLastRow(sheet_id, sheet_UD, uid_column);  // Look for a row to write the new user's UID.
```

```
      var newUID = sheet_open.getRange(uid_column + (getLastRowUIDCol + 1));
      newUID.setValue(uid_val);
      result += ",R_Successful";

      // Sends response payload to ESP32.
      return ContentService.createTextOutput(result);
    }
    //----------------------------------------

    //----------------------------------------Conditions for filling attendance (Time In and Time Out).
    if (sts_val == 'atc') {
      // Checks whether the UID is already registered in the "user data" sheet.
      // findUID(Spreadsheet ID, sheet name, index column, UID value)
      // index column : 1 = column A, 2 = column B and so on.
      var FUID = findUID(sheet_id, sheet_UD, 2, uid_val);

      // "(FUID == -1)" means that the UID has not been registered in the "user data" sheet, so
attendance filling is rejected.
      if (FUID == -1) {
        result += ",atcErr01"; // atcErr01 = UID not registered.
        return ContentService.createTextOutput(result);
      } else {
        // After the UID has been checked and the result is that the UID has been registered,
        // then take the "name" of the UID owner from the "user data" sheet.
        // The name of the UID owner is in column "A" on the "user data" sheet.
        var get_Range = sheet_user_data.getRange("A" + (FUID+2));
        var user_name_by_UID = get_Range.getValue();

        // Variables to determine attendance filling, whether to fill in "Time In", "Time Out" or
attendance has been completed for today.
        var enter_data = "time_in";

        // Variable to get row position. This is used to fill in "Time Out".
        var num_row;

        // Variables to get the current Date and Time.
        var Curr_Date = Utilities.formatDate(new Date(), "Asia/Kathmandu", 'dd/MM/yyyy');
        var Curr_Time = Utilities.formatDate(new Date(), "Asia/Kathmandu", 'HH:mm:ss');

        // Variable to get all the data from the "attendance" sheet.
        var data = sheet_attendence.getDataRange().getDisplayValues();

        //..................Check whether "Time In" or "Time Out" is filled in.
        if (data.length > 1) {
          for (var i = 0; i < data.length; i++) {
            if (data[i][1] == uid_val) {
```

```
 if (data[i][2] == Curr_Date) {
  if (data[i][4] === undefined || data[i][4] === "") {  // If Time Out is empty or undefined, it's time to record Time Out.
   Date_val = data[i][2];
   TI_val = data[i][3];
   enter_data = "time_out";
   num_row = i + 1;
   break;
  } else {
   enter_data = "finish";
  }
 }
}
     }
    }
    //..................

    //..................Conditions for filling in "Time In" attendance.
    if (enter_data == "time_in") {
     sheet_attendence.insertRows(2);
     sheet_attendence.getRange("A2").setValue(user_name_by_UID);
     sheet_attendence.getRange("B2").setValue(uid_val);
     sheet_attendence.getRange("C2").setValue(Curr_Date);
```

```
      sheet_attendence.getRange("D2").setValue(Curr_Time);
      SpreadsheetApp.flush();

      // Sends response payload to ESP32.
      result += ",TI_Successful" + "," + user_name_by_UID + "," + Curr_Date + "," +
Curr_Time;
      return ContentService.createTextOutput(result);
    }
    //..................

    //..................Conditions for filling in "Time Out" attendance.
    if (enter_data == "time_out") {
      sheet_attendence.getRange("E" + num_row).setValue(Curr_Time);
      result += ",TO_Successful" + "," + user_name_by_UID + "," + Date_val + "," + TI_val +
"," + Curr_Time;

      // Sends response payload to ESP32.
      return ContentService.createTextOutput(result);
    }
    //..................

    //..................Condition when "Time In" and "Time Out" are filled.
    if (enter_data == "finish") {
```

```
  result += ",atcInf01"; // atcInf01 = You have completed your attendance record for today.


  // Check if the user scanned again for a new session on the same day
  if (data[i][4] !== "") {  // If Time Out is already filled, allow a new entry for the next scan.
    result += ",New_Session_Allowed";
    // Optionally, log this to indicate that a new session is allowed.
    // Sends response payload to ESP32.
    return ContentService.createTextOutput(result);
  }
}


        //.................
      }
    }
    //---------------------------------------
  }
}
//__________________________________________________________________________________


//__________________________________________________________________________________stripQuotes()
function stripQuotes( value ) {
  return value.replace(/^["']|["']$/g, "");
}
//______________________________________________________________________________
____


//______________________________________________________________________________
____findLastRow()
// Function to find the last row in a certain column.
// Reference : https://www.jsowl.com/find-the-last-row-of-a-single-column-in-google-sheets-in-
apps-script/
function findLastRow(id_sheet, name_sheet, name_column) {
  var spreadsheet = SpreadsheetApp.openById(id_sheet);
  var sheet = spreadsheet.getSheetByName(name_sheet);
  var lastRow = sheet.getLastRow();

  var range = sheet.getRange(name_column + lastRow);

  if (range.getValue() !== "") {
    return lastRow;
  } else {
    return range.getNextDataCell(SpreadsheetApp.Direction.UP).getRow();
  }
}
```

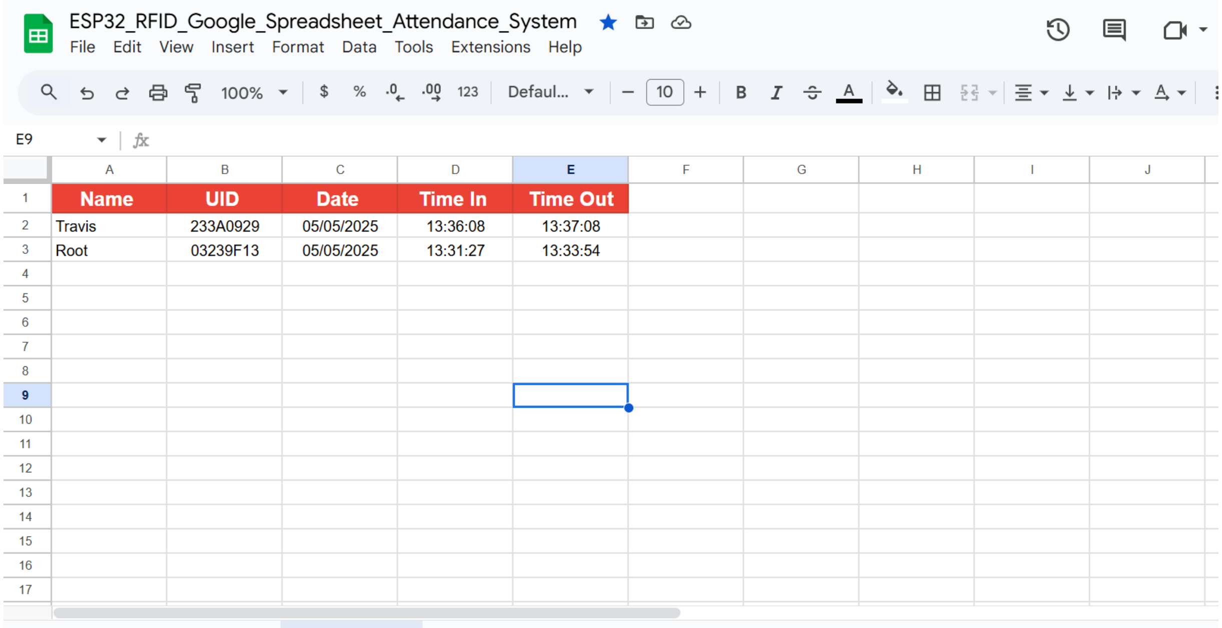
```
//________________________________________________________________________________

//________________________________________________________________________________findUID()
// Reference : https://stackoverflow.com/a/29546373
function findUID(id_sheet, name_sheet, column_index, searchString) {
  var open_sheet = SpreadsheetApp.openById(id_sheet);
  var sheet = open_sheet.getSheetByName(name_sheet);
  var columnValues = sheet.getRange(2, column_index, sheet.getLastRow()).getValues();  // 1st is header row.
  var searchResult = columnValues.findIndex(searchString);  // Row Index - 2.

  return searchResult;
}
//________________________________________________________________________________

//________________________________________________________________________________checkUID()
// Reference : https://stackoverflow.com/a/29546373
function checkUID(id_sheet, name_sheet, column_index, searchString) {
  var open_sheet = SpreadsheetApp.openById(id_sheet);
  var sheet = open_sheet.getSheetByName(name_sheet);
  var columnValues = sheet.getRange(2, column_index, sheet.getLastRow()).getValues();  // 1st is header row.
  var searchResult = columnValues.findIndex(searchString);  // Row Index - 2.

  if(searchResult != -1) {
    // searchResult + 2 is row index.
    sheet.setActiveRange(sheet.getRange(searchResult + 2, 3)).setValue("UID has been registered in this row.");
    return true;
  } else {
    return false;
  }
}
//____________________________________________________________________________________________

//____________________________________________________________________________________________findIndex()
Array.prototype.findIndex = function(search){
  if(search == "") return false;
  for (var i=0; i<this.length; i++)
    if (this[i].toString().indexOf(search) > -1 ) return i;

  return -1;
```

```
}
```

8.2 Algorithm

```
Step 1: START
Step 2: Initialize the ESP32 and RFID RC522 reader.
Step 3: Connect the ESP32 to the Wi-Fi network.
Step 4: Define the Google Sheets URL and the script deployment ID for data posting.
Step 5: Continuously scan for RFID card presence.
Step 6: IF an RFID card is detected:
    Step 6.1: Read the unique ID of the scanned RFID card.
    Step 6.2: Retrieve the current timestamp.
    Step 6.3: Check if this RFID card ID has an existing "time-in" record in the Google Sheet that does NOT have a corresponding "time-out".
    Step 6.4: IF an open "time-in" record exists for this card:
        Step 6.4.1: Record the current timestamp as the "time-out" for that RFID card ID in the Google Sheet.
        Step 6.4.2: Optionally, display a "Time-out recorded" message (if using a display).
    Step 6.5: ELSE (if no open "time-in" record exists):
        Step 6.5.1: Record the RFID card ID and the current timestamp as the "time-in" in a new row of the Google Sheet.
        Step 6.5.2: Optionally, display a "Time-in recorded" message (if using a display).
Step 7: Wait for a short duration before the next scan to avoid multiple reads of the same card.
Step 8: Go back to Step 5.
Step 9: END (This step is conceptual, as the system will run continuously).
```

8.3 Google Sheet Data Format Snapshot


ESP32_RFID_Google_Spreadsheet_Attendance_System
File Edit View Insert Format Data Tools Extensions Help
100%
Defaul...
10
E9

Name	UID	Date	Time In	Time Out
Travis	233A0929	05/05/2025	13:36:08	13:37:08
Root	03239F13	05/05/2025	13:31:27	13:33:54

Figure 34 Google Sheet Data Format Snapshot

8.4 Individual contribution Plan

Student Name	Role	Contribution
Sanshree Shrestha	**Proposal**: Abstract, Introduction, and initial project idea. **System Development Report**: Introduction (1), Current Scenario (1.1). **Application Implementation**: Setting up the Arduino IDE for ESP32 and basic ESP32 environment.	17%
Anshumala Bhandari	**Proposal**: Aims and Objectives. **System Development Report**: Problem Statement and Project as a Solution (1.2), Aims and Objectives (1.3). **Application Implementation**: Interfacing the RFID RC522 module with the ESP32.	17%
Bibhuti Sigdel	**Proposal**: Initial System overview for the proposal. **System Development Report**: System overview (2.1), Block diagram (2.2.1).	17%

	Application Implementation: Writing code to read the unique ID (UID) from the RFID tags.	
Sagun Budhathoki	**Proposal**: Initial ideas for Diagram Design. **System Development Report**: System Architecture (2.2.2), Flowchart (2.2.3). **Application Implementation**: Interfacing the LCD with the ESP32 to display information.	17%
Shraddha Budhathoki	**Proposal**: Contributing to the initial Requirement Analysis. **System Development Report**: Circuit Diagram (2.2.4), Schematic Diagram (2.2.5). **Application Implementation**: Integrating the RFID reading and LCD display functionalities.	16%
Sameer Sah	**Proposal**: Contributing to the initial Requirement Analysis. **System Development Report**: Data Flow Diagram (DFD) (2.3), Hardware Components (2.4.1), Software Requirements (2.4.2). **Application Implementation**: Working on data logging (to a serial monitor or preparing for cloud integration).	16%

Table 10 Individual contribution Plan