RFID-Based Access Control System with RTC, Google Sheet Integration, and Stepper Motor Interface

This document outlines a comprehensive ESP32-based RFID access control system that includes:

- UID verification
- Real-time clock (RTC) for timestamping
- Data logging to Google Sheets
- Control of gate opening via a servo motor
- Visual/auditory feedback (LED + buzzer)
- Integration of a stepper motor

Hardware Components Used:

- ESP32 microcontroller
- MFRC522 RFID Reader
- DS1302 RTC Module
- Servo Motor (for gate mechanism)
- Buzzer and LED (visual and audio indicators)
- 16x2 LCD with I2C for UID and status display

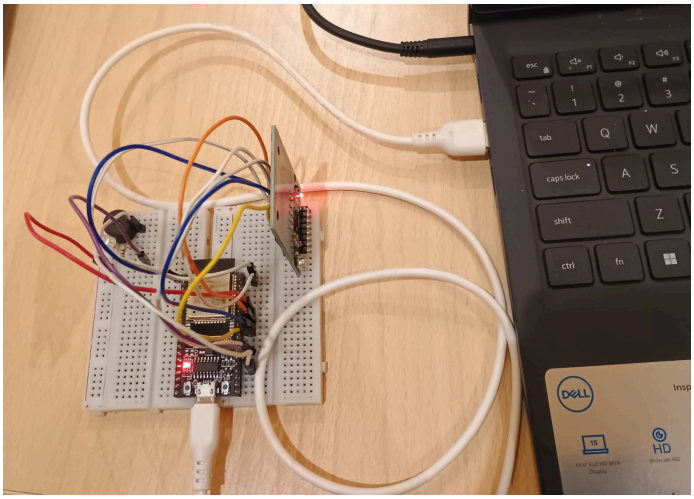
Functionality Breakdown:

1. RFID Detection
    - The RFID module reads the UID of scanned cards.
    - It compares the UID with a predefined value.
    - Matching UID results in access being granted.

2. Gate Control
    - A servo motor opens the gate upon successful UID verification.
    - The buzzer and LED provide feedback.
3. Time Logging
    - The RTC module logs the exact date and time of the event.
4. Data Logging via Internet
    - Wi-Fi connection is established.
    - UID, access status, and timestamp are sent via HTTP POST to a Google Apps Script.
    - Data is appended to a Google Sheet.

RC522 RFID:

RC522 Pin	ESP32 Pin
SDA	GPIO5
SCK	GPIO18
MOSI	GPIO23
MISO	GPIO19
RST	GPIO4
VCC	3.3V
GND	GND

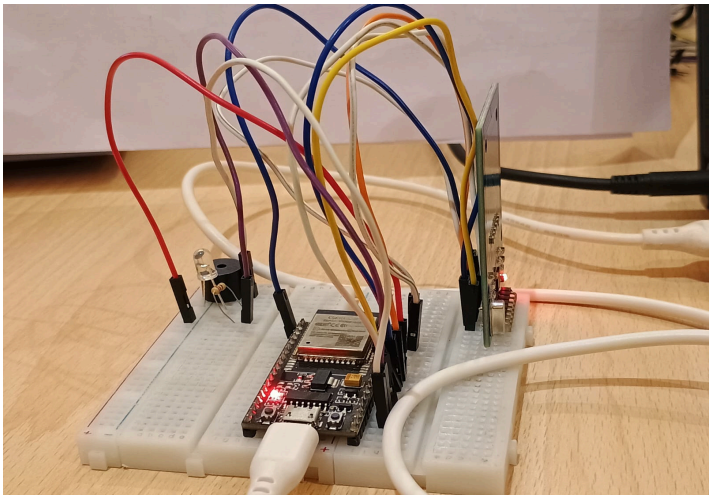


LED:

- Anode (long leg) → 220Ω resistor → **GPIO2**
- Cathode → GND

Buzzer:

- Positive (+) → **GPIO15**
- Negative (–) → GND

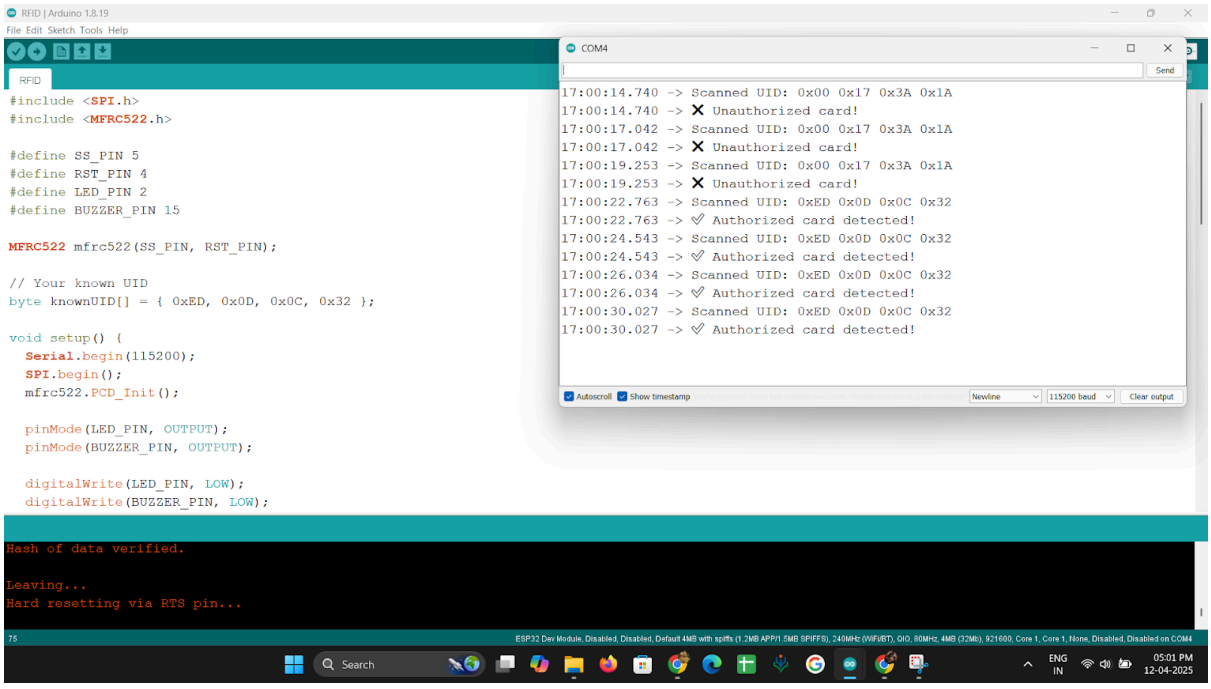


Full Code with LED & Buzzer for both Authorized and Unauthorized Cards:

```
#include <SPI.h>
#include <MFRC522.h>
#define SS_PIN 5
#define RST_PIN 4
#define LED_PIN 2
#define BUZZER_PIN 15

MFRC522 mfrc522(SS_PIN, RST_PIN);
// Your known UID
byte knownUID[] = { 0xED, 0x0D, 0x0C, 0x32 };
void setup() {
  Serial.begin(115200);
  SPI.begin();
  mfrc522.PCD_Init();
  pinMode(LED_PIN, OUTPUT);
  pinMode(BUZZER_PIN, OUTPUT);
  digitalWrite(LED_PIN, LOW);
  digitalWrite(BUZZER_PIN, LOW);
  Serial.println("Scan your card...");
}
void loop() {
  if (!mfrc522.PICC_IsNewCardPresent() ||
!mfrc522.PICC_ReadCardSerial()) {
    return;
  }
  Serial.print("Scanned UID: ");
  bool match = true;
  for (byte i = 0; i < mfrc522.uid.size; i++) {
    Serial.print("0x");
    if (mfrc522.uid.uidByte[i] < 0x10) Serial.print("0");
    Serial.print(mfrc522.uid.uidByte[i], HEX);
    Serial.print(" ");
    if (mfrc522.uid.uidByte[i] != knownUID[i]) {
      match = false;
    }
  }
  Serial.println();
```

```
  if (match) {
    Serial.println("✅ Authorized card detected!");
    digitalWrite(LED_PIN, HIGH);
    digitalWrite(BUZZER_PIN, HIGH);
    delay(500); // 0.5 sec ON
    digitalWrite(LED_PIN, LOW);
    digitalWrite(BUZZER_PIN, LOW);
  } else {
    Serial.println("❌ Unauthorized card!");
    for (int i = 0; i < 3; i++) {
      digitalWrite(LED_PIN, HIGH);
      digitalWrite(BUZZER_PIN, HIGH);
      delay(200);
      digitalWrite(LED_PIN, LOW);
      digitalWrite(BUZZER_PIN, LOW);
      delay(200);
    }
  }
  // Halt PICC and stop encryption
  mfrc522.PICC_HaltA();
  mfrc522.PCD_StopCrypto1();
  delay(1000); // Pause before next scan
}
```

Result:



INTERFACE WITH LCD DISPLAY

Components Needed

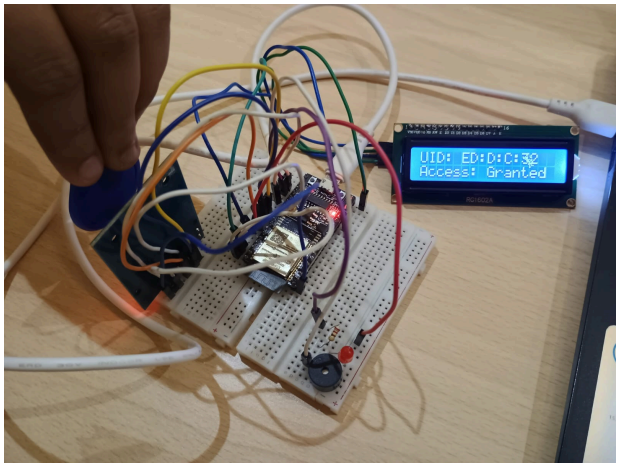
- 16x2 LCD (with or without I2C module)
- RFID-RC522 module
- ESP32
- LED (GPIO 2), Buzzer (GPIO 15)
- Jumper wires

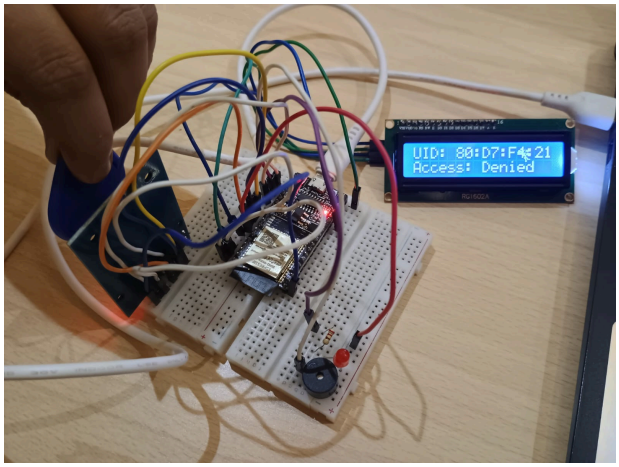
CODE

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <SPI.h>
#include <MFRC522.h>

// --- Pin Definitions ---
#define SS_PIN 5           // RFID SS (SDA)
#define RST_PIN 4          // RFID RST
#define LED_PIN 2          // On-board LED
#define BUZZER_PIN 15      // Buzzer pin

// --- Initialize RFID and LCD ---
LiquidCrystal_I2C lcd(0x27, 16, 2);     // LCD at I2C address 0x27
MFRC522 mfrc522(SS_PIN, RST_PIN);       // RFID module

// --- Your Card UID ---
byte knownUID[] = { 0xED, 0x0D, 0x0C, 0x32 };  // Your UID (4 bytes)

void setup() {
  Serial.begin(115200);         // Open serial communication
  SPI.begin();                  // Init SPI for RFID
  mfrc522.PCD_Init();           // Init MFRC522

  pinMode(LED_PIN, OUTPUT);
  pinMode(BUZZER_PIN, OUTPUT);

  // Initialize I2C for ESP32 and LCD
  Wire.begin(21, 22);           // SDA = 21, SCL = 22
  lcd.begin();                  // Initialize LCD
```

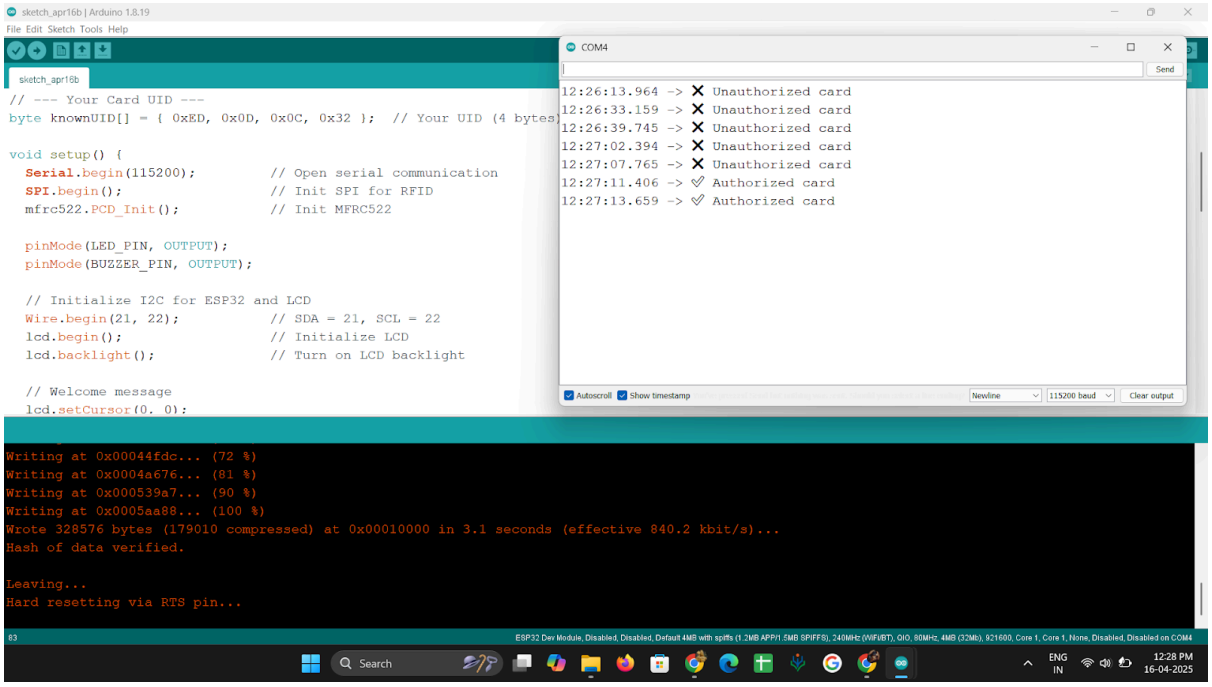
```
  lcd.backlight();              // Turn on LCD backlight

  // Welcome message
  lcd.setCursor(0, 0);
  lcd.print("Scan your card");
}

void loop() {
  // Wait for a new card
  if (!mfrc522.PICC_IsNewCardPresent() ||
!mfrc522.PICC_ReadCardSerial()) {
    return;
  }

  bool match = true;
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("UID: ");

  // Display UID & compare
  for (byte i = 0; i < mfrc522.uid.size; i++) {
    lcd.print(mfrc522.uid.uidByte[i], HEX);
    if (i < mfrc522.uid.size - 1) lcd.print(":");

    if (mfrc522.uid.uidByte[i] != knownUID[i]) {
      match = false;
    }
  }

  lcd.setCursor(0, 1);

  if (match) {
    Serial.println("✅ Authorized card");
    lcd.print("Access: Granted");

    digitalWrite(LED_PIN, HIGH);
    digitalWrite(BUZZER_PIN, HIGH);
    delay(500);
```

```
    digitalWrite(LED_PIN, LOW);
    digitalWrite(BUZZER_PIN, LOW);
  } else {
    Serial.println("❌ Unauthorized card");
    lcd.print("Access: Denied");

    for (int i = 0; i < 3; i++) {
      digitalWrite(LED_PIN, HIGH);
      digitalWrite(BUZZER_PIN, HIGH);
      delay(200);
      digitalWrite(LED_PIN, LOW);
      digitalWrite(BUZZER_PIN, LOW);
      delay(200);
    }
  }

  // End communication with current card
  mfrc522.PICC_HaltA();
  mfrc522.PCD_StopCrypto1();

  delay(1000);  // Pause before next scan
}
```





OUTPUT:



STEPPER MOTOR INTERFACE:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <SPI.h>
#include <MFRC522.h>
#include <ESP32Servo.h>

#define SS_PIN 5
#define RST_PIN 4
#define LED_PIN 2
#define BUZZER_PIN 15
#define SERVO_PIN 13

byte knownUID[] = { 0xED, 0x0D, 0x0C, 0x32 };

LiquidCrystal_I2C lcd(0x27, 16, 2);
MFRC522 mfrc522(SS_PIN, RST_PIN);
Servo gateServo;
```

```
void setup() {
  Serial.begin(115200);
  SPI.begin();
  mfrc522.PCD_Init();

  pinMode(LED_PIN, OUTPUT);
  pinMode(BUZZER_PIN, OUTPUT);

  lcd.begin();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Scan your card");

  gateServo.setPeriodHertz(50);
  gateServo.attach(SERVO_PIN, 500, 2400);  // Signal pin for servo
  gateServo.write(0); // Start at 0°
}

void loop() {
  if (!mfrc522.PICC_IsNewCardPresent() || !mfrc522.PICC_ReadCardSerial()) {
    return;
  }

  bool match = true;
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("UID: ");

  for (byte i = 0; i < mfrc522.uid.size; i++) {
    lcd.print(mfrc522.uid.uidByte[i], HEX);
    if (i < mfrc522.uid.size - 1) lcd.print(":");
    if (mfrc522.uid.uidByte[i] != knownUID[i]) match = false;
  }
```

```
  if (match) {
    Serial.println("✅ Access Granted");
    lcd.setCursor(0, 1);
    lcd.print("Access: Granted");
    digitalWrite(LED_PIN, HIGH);
    digitalWrite(BUZZER_PIN, HIGH);
    gateServo.write(180);   // Open gate
    delay(500);
    digitalWrite(LED_PIN, LOW);
    digitalWrite(BUZZER_PIN, LOW);
    delay(3000);
    gateServo.write(0);     // Close gate
  } else {
    Serial.println("❌ Access Denied");
    lcd.setCursor(0, 1);
    lcd.print("Access: Denied");
    for (int i = 0; i < 3; i++) {
      digitalWrite(LED_PIN, HIGH);
      digitalWrite(BUZZER_PIN, HIGH);
      delay(200);
      digitalWrite(LED_PIN, LOW);
      digitalWrite(BUZZER_PIN, LOW);
      delay(200);
    }
    gateServo.write(0);  // Keep gate closed
  }

  mfrc522.PICC_HaltA();
  mfrc522.PCD_StopCrypto1();
  delay(1000);
}
```

WhatsApp Video 2025-04-17 at 4.22.46 PM.mp4

RTC Wiring Configuration:

DS1302 Pin	ESP32 Pin	Description
VCC	3.3V	Power supply
GND	GND	Ground
CLK	GPIO14	Clock
DAT	GPIO19	Data I/O
RST	GPIO5	Chip Enable (CE)

RFID UID Matching Logic

```C/C++
bool match = true;
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("UID: ");

for (byte i = 0; i < mfrc522.uid.size; i++) {
  lcd.print(mfrc522.uid.uidByte[i], HEX);
```

```
  if (i < mfrc522.uid.size - 1) lcd.print(":");
  if (mfrc522.uid.uidByte[i] != knownUID[i]) match = false;
}

if (match) {
  Serial.println("✅ Access Granted");
  lcd.setCursor(0, 1);
  lcd.print("Access: Granted");
  digitalWrite(LED_PIN, HIGH);
  digitalWrite(BUZZER_PIN, HIGH);
  gateServo.write(180);   // Open gate
  delay(500);
  digitalWrite(LED_PIN, LOW);
  digitalWrite(BUZZER_PIN, LOW);
  delay(3000);
  gateServo.write(0);     // Close gate
} else {
  Serial.println("❌ Access Denied");
  lcd.setCursor(0, 1);
  lcd.print("Access: Denied");
  for (int i = 0; i < 3; i++) {
    digitalWrite(LED_PIN, HIGH);
    digitalWrite(BUZZER_PIN, HIGH);
    delay(200);
    digitalWrite(LED_PIN, LOW);
    digitalWrite(BUZZER_PIN, LOW);
    delay(200);
  }
  gateServo.write(0);
}

mfrc522.PICC_HaltA();
mfrc522.PCD_StopCrypto1();
delay(1000);
```

RTC Module Interface (DS1302)

DS1302 Pin	ESP32 Pin	Description
VCC	3.3V	Power Supply
GND	GND	Ground
CLK	GPIO14	Clock
DAT	GPIO19	Data I/O
RST	GPIO5	Chip Enable (CE)

ESP32 Code with RTC

```C/C++
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <SPI.h>
#include <MFRC522.h>
#include <ESP32Servo.h>
#include <ThreeWire.h>
#include <RtcDS1302.h>
#include <WiFi.h>
#include <HTTPClient.h>

// ---------------------- Pins & Constants ----------------------
#define SS_PIN 5
#define RST_PIN 4
#define LED_PIN 2
#define BUZZER_PIN 15
#define SERVO_PIN 13

#define DS1302_CLK 14  // SCLK
#define DS1302_IO  27  // DAT
#define DS1302_CE  26  // RST
```

```
byte knownUID[] = { 0xED, 0x0D, 0x0C, 0x32 };  // Replace with
your card's UID

// ---------------------- Components ----------------------
LiquidCrystal_I2C lcd(0x27, 16, 2);
MFRC522 mfrc522(SS_PIN, RST_PIN);
Servo gateServo;

ThreeWire myWire(DS1302_IO, DS1302_CLK, DS1302_CE);
RtcDS1302<ThreeWire> Rtc(myWire);

// ---------------------- Wi-Fi Setup ----------------------
const char* ssid = "YOUR_WIFI_SSID";             // Replace with
your WiFi SSID
const char* password = "YOUR_WIFI_PASSWORD";     // Replace with
your WiFi password
const String scriptURL =
"https://script.google.com/macros/s/XXXXXXXXXXXX/exec"; //
Replace with your Web App URL

// ---------------------- Setup ----------------------
void setup() {
  Serial.begin(115200);
  SPI.begin();
  mfrc522.PCD_Init();

  pinMode(LED_PIN, OUTPUT);
  pinMode(BUZZER_PIN, OUTPUT);

  lcd.begin();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Scan your card");

  gateServo.setPeriodHertz(50);
  gateServo.attach(SERVO_PIN, 500, 2400);
```

```
  gateServo.write(0);  // Gate closed

  // RTC setup
  Rtc.Begin();
  RtcDateTime customTime(2025, 4, 19, 15, 45, 00); // Set initial
time if needed
  Rtc.SetDateTime(customTime);

  if (!Rtc.IsDateTimeValid()) {
    Serial.println("RTC DateTime invalid!");
  }
  if (!Rtc.GetIsRunning()) {
    Serial.println("Starting RTC...");
    Rtc.SetIsRunning(true);
  }

  // Connect Wi-Fi
  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWiFi connected.");
}

// ---------------------- Main Loop ----------------------
void loop() {
  if (!mfrc522.PICC_IsNewCardPresent() ||
!mfrc522.PICC_ReadCardSerial()) {
    return;
  }

  bool match = true;
  String uidStr = "";
  lcd.clear();
```

```
lcd.setCursor(0, 0);
lcd.print("UID: ");

for (byte i = 0; i < mfrc522.uid.size; i++) {
  byte val = mfrc522.uid.uidByte[i];
  uidStr += String(val, HEX);
  lcd.print(val, HEX);
  if (i < mfrc522.uid.size - 1) {
    lcd.print(":");
    uidStr += ":";
  }
  if (val != knownUID[i]) match = false;
}

uidStr.toUpperCase();

if (match) {
  Serial.println("✅ Access Granted");
  lcd.setCursor(0, 1);
  lcd.print("Access: Granted");
  digitalWrite(LED_PIN, HIGH);
  digitalWrite(BUZZER_PIN, HIGH);
  gateServo.write(180);  // Open gate
  delay(500);
  digitalWrite(LED_PIN, LOW);
  digitalWrite(BUZZER_PIN, LOW);
  delay(3000);
  gateServo.write(0);    // Close gate
  sendToGoogleSheet(uidStr, "Granted");
} else {
  Serial.println("❌ Access Denied");
  lcd.setCursor(0, 1);
  lcd.print("Access: Denied");
  for (int i = 0; i < 3; i++) {
    digitalWrite(LED_PIN, HIGH);
    digitalWrite(BUZZER_PIN, HIGH);
```

```
      delay(200);
      digitalWrite(LED_PIN, LOW);
      digitalWrite(BUZZER_PIN, LOW);
      delay(200);
    }
    gateServo.write(0);
    sendToGoogleSheet(uidStr, "Denied");
  }

  RtcDateTime now = Rtc.GetDateTime();
  printDateTime(now);

  mfrc522.PICC_HaltA();
  mfrc522.PCD_StopCrypto1();
  delay(1000);
}

// ---------------------- Google Sheet Logging
----------------------
void sendToGoogleSheet(String uid, String status) {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    String fullURL = scriptURL + "?uid=" + uid + "&status=" +
status;
    http.begin(fullURL);
    int httpCode = http.GET();
    if (httpCode > 0) {
      Serial.print("Data sent: ");
      Serial.println(httpCode);
    } else {
      Serial.print("Failed to send. Code: ");
      Serial.println(httpCode);
    }
    http.end();
  } else {
    Serial.println("WiFi not connected");
```

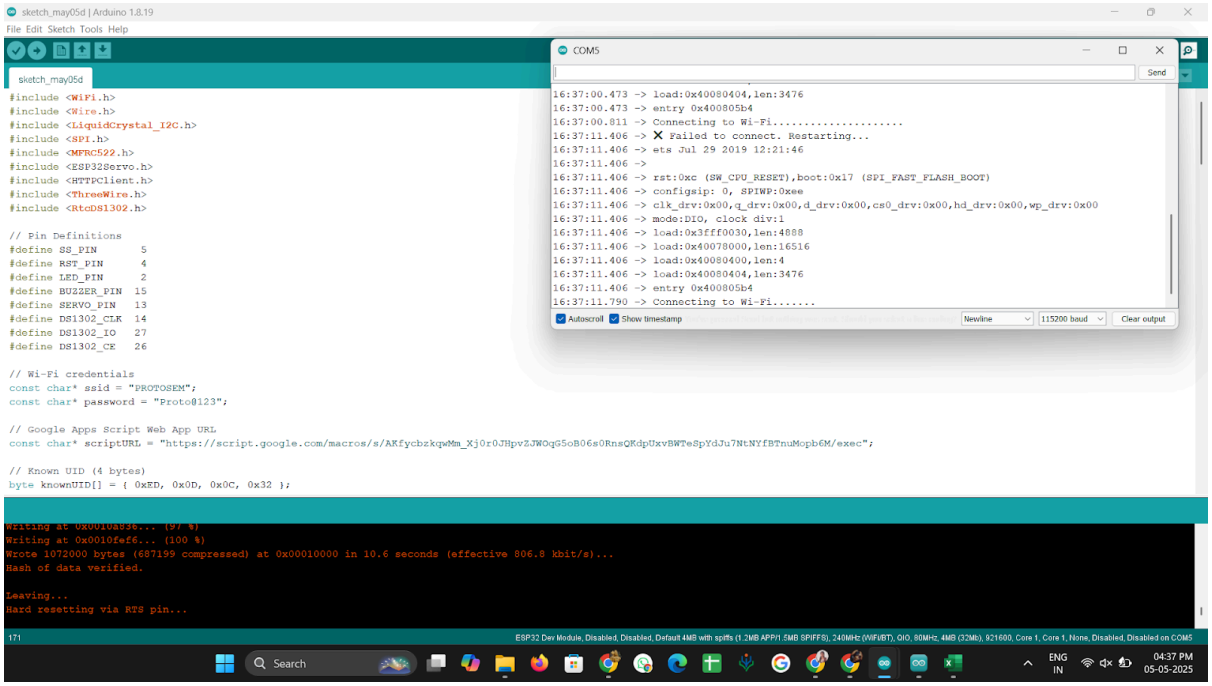
```
  }
}

// ---------------------- RTC Time Printer ----------------------
void printDateTime(const RtcDateTime& dt) {
  char buf[20];
  snprintf_P(buf, sizeof(buf), PSTR("%02u/%02u/%04u
%02u:%02u:%02u"),
             dt.Month(), dt.Day(), dt.Year(),
             dt.Hour(), dt.Minute(), dt.Second());
  Serial.println(buf);
}
```

Google Apps Script for Data Logging

JavaScript

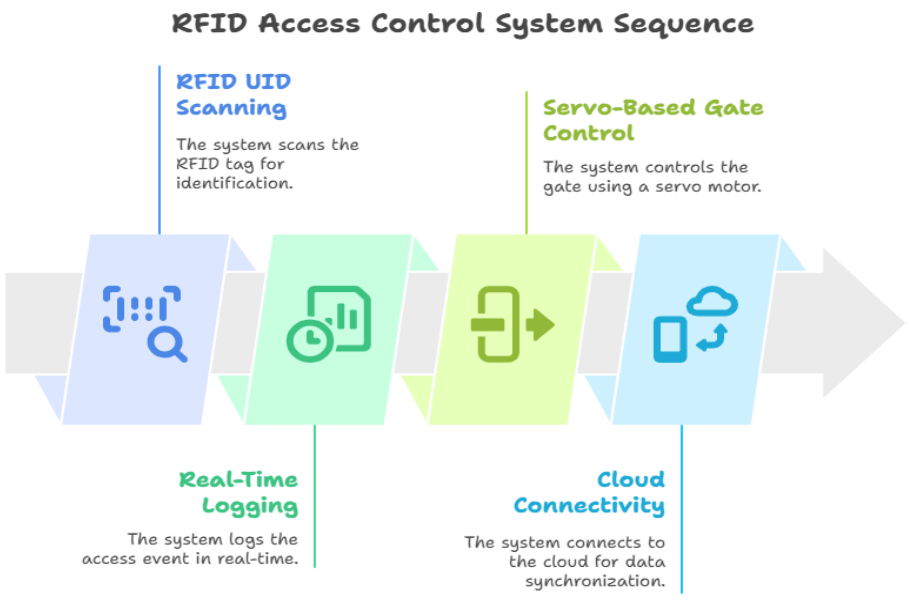
```
function doGet(e) {
  var ss = SpreadsheetApp.openById("YOUR_SPREADSHEET_ID");
  var sheet = ss.getSheetByName("Sheet1");

  var uid = e.parameter.uid;
  var status = e.parameter.status;
  var timestamp = new Date();

  sheet.appendRow([timestamp, uid, status]);

  return ContentService.createTextOutput("Success");
}
```

Deploy as a Web App with access set to "Anyone".

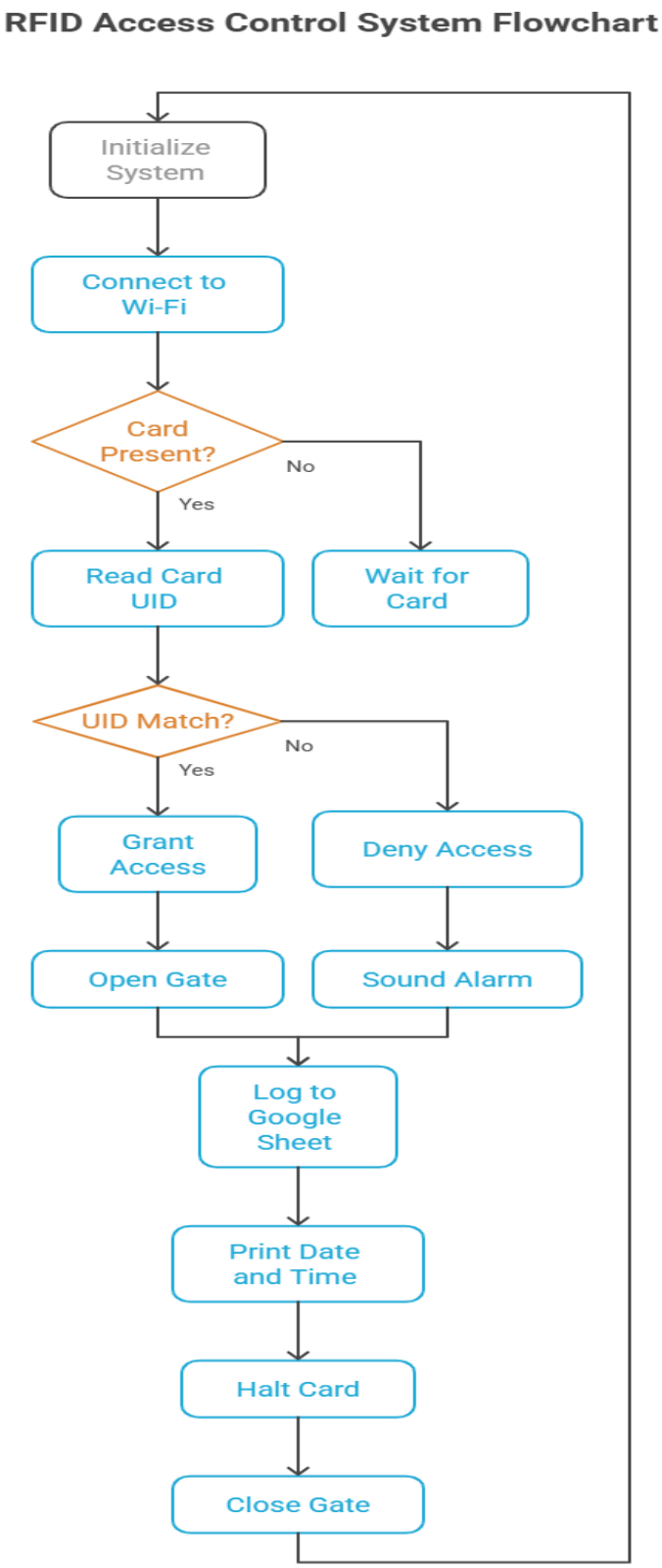


C/C++

Block Diagram:



Flowchart:


RFID Access Control System Flowchart
Initialize System
Connect to Wi-Fi
Card Present?
No
Yes
Read Card UID
Wait for Card
UID Match?
No
Yes
Grant Access
Deny Access
Open Gate
Sound Alarm
Log to Google Sheet
Print Date and Time
Halt Card
Close Gate

Summary:

This project showcases a practical implementation of an RFID access control system integrated with RTC timekeeping and cloud-based logging using Google Sheets. It’s ideal for school/college entry logs, attendance systems, or smart home security gates.