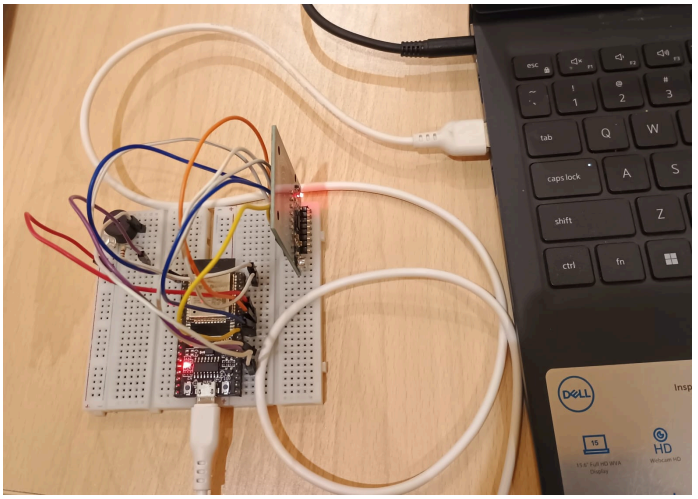
IoT-Based RFID Access Control System with Real-Time Logging to Google Sheets

INTERFACE RFID WITH ESP32

RC522 RFID:

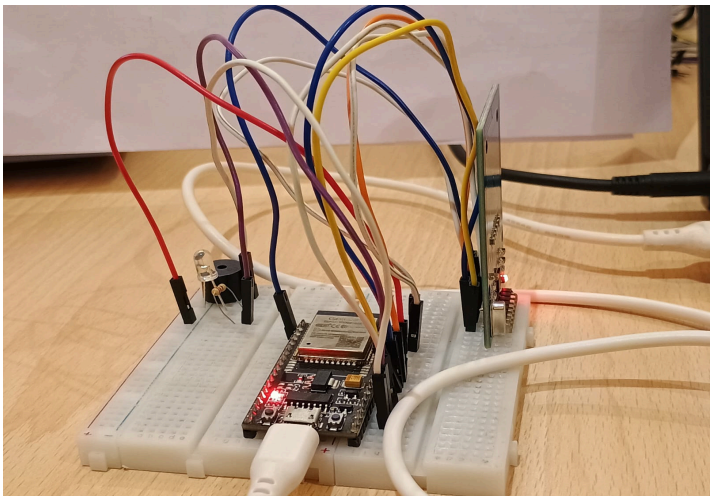
RFID Module ↔ ESP32:

- SDA → GPIO5
- SCK → GPIO18
- MOSI → GPIO23
- MISO → GPIO19
- RST → GPIO4
- VCC → 3.3V
- GND → GND



LED:

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



Buzzer:

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

Code:

```
#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);

byte validUID[4] = {0xDE, 0xAD, 0xBE, 0xEF};

void setup() {
  Serial.begin(115200);
  SPI.begin(); mfrc522.PCD_Init();
  pinMode(LED_PIN, OUTPUT);
  pinMode(BUZZER_PIN, OUTPUT); }

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

if (isUIDMatch(mfrc522.uid.uidByte, validUID)) { blinkAndBeep(LED_PIN,
BUZZER_PIN, 1, 200);
Serial.println("Valid User"); } else { blinkAndBeep(LED_PIN, BUZZER_PIN, 3, 200);
Serial.println("Unknown User"); }

mfrc522.PICC_HaltA(); }

bool isUIDMatch(byte *a, byte *b) {
  for (byte i = 0; i < 4; i++) {
    if (a[i] != b[i]) return false; } return true; }

void blinkAndBeep(int ledPin, int buzzerPin, int times, int delayTime) {
  for (int i = 0; i < times; i++) {
    digitalWrite(ledPin, HIGH);
    digitalWrite(buzzerPin, HIGH);
    delay(delayTime); digitalWrite(ledPin, LOW);
    digitalWrite(buzzerPin, LOW);
    delay(delayTime); }
 }
```

OUTPUT LINK : **BUZZER.mp4**

1. To replace that dummy UID with your actual RFID tag's UID, follow these steps:

1. Read your tag's UID:

Upload this minimal sketch to your ESP32:

```
#include <SPI.h>
#include <MFRC522.h>
#define SS_PIN 5     // SDA pin (connect to D5 on ESP32 or pin 10 on Arduino Uno)
#define RST_PIN 4    // RST pin (connect to D4 on ESP32 or pin 9 on Arduino Uno)
MFRC522 mfrc522(SS_PIN, RST_PIN); // Create MFRC522 instance
void setup() {
  Serial.begin(115200);  // Start serial communication
  SPI.begin();           // Init SPI bus
  mfrc522.PCD_Init();    // Init MFRC522
  Serial.println("Scan your card...");
}
void loop() {
  // Look for a new card
  if (!mfrc522.PICC_IsNewCardPresent() || !mfrc522.PICC_ReadCardSerial()) {
    return;
  }
  // Show UID
  Serial.print("UID: ");
  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(" ");
```

```
  }
  Serial.println();
  delay(1000); // Add delay to avoid flooding the output
}
```

Open the Serial Monitor (set to 115200 baud), then scan your RFID tag. You'll see something like: **UID: 0xA1, 0xB2, 0xC3, 0xD4**

REFERENCE LINK : **RFID UID.mp4** **UID.mp4**

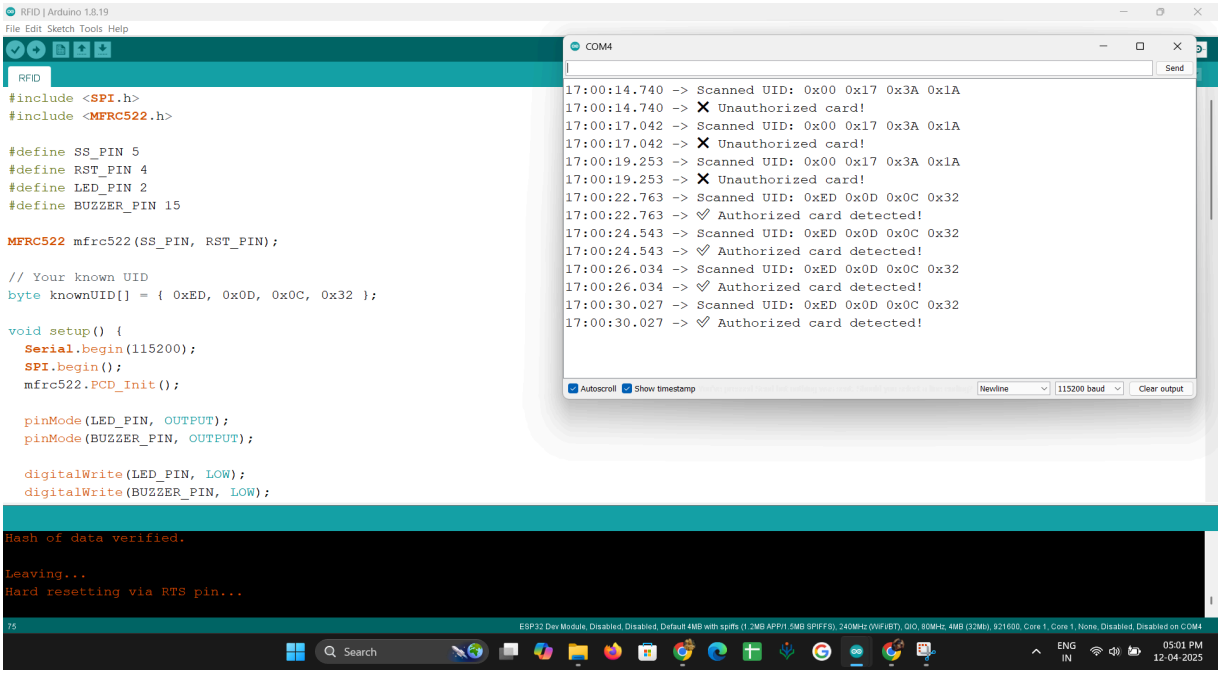
2. Updated Full Code with LED & Buzzer for both Authorized and Unauthorized Cards:

```
#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:



REFERENCE LINK: RFID.mp4

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

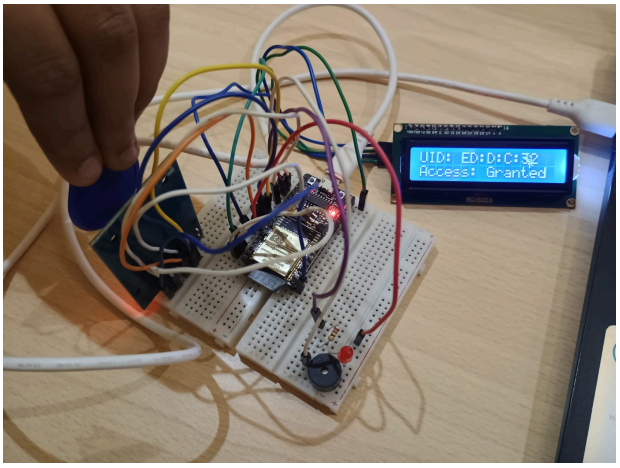
Code:

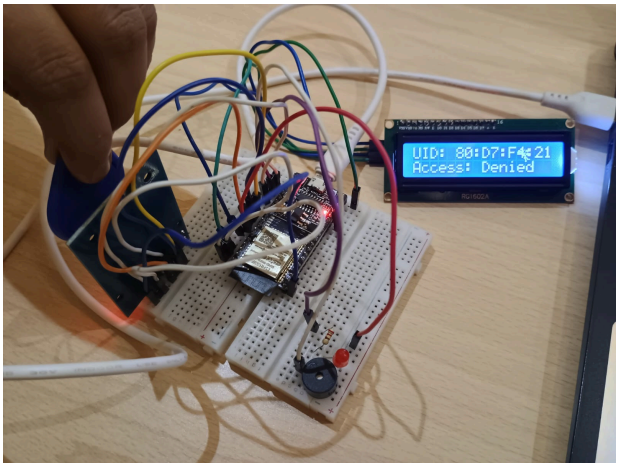
```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

```
#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
```

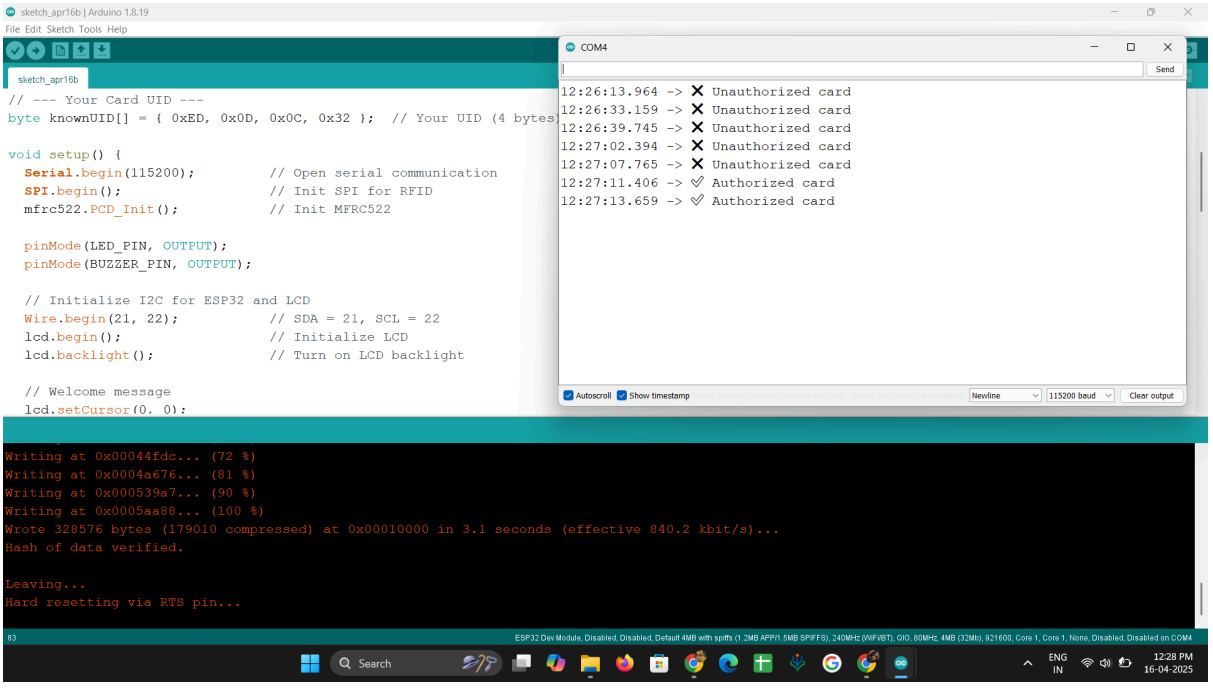
```
  if (!mfrc522.PICC_IsNewCardPresent() ||
!mfrc522.PICC_ReadCardSerial()) {
    return;
  }

  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:



REFERENCE LINK: WhatsApp Video 2025-04-16 at 12.28.09 PM.mp4

SERVO INTERFACE WITH ALL:

Connections:

- **Orange (Signal):** GPIO 13 (ESP32)
- **Red (VCC):** 5V (External Power Supply)
- **Brown (GND):** Common GND (ESP32 + Power Supply)

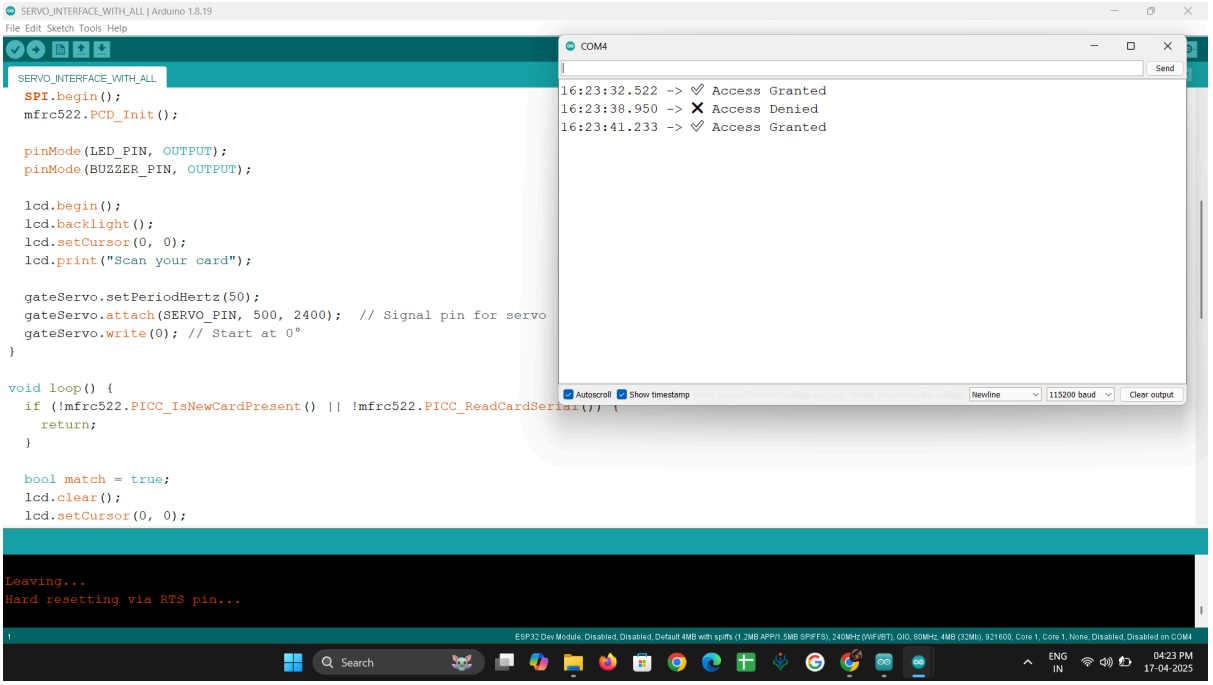
Code:

```
#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);
}
```

Output:



REFERENCE LINK

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

RTC MODULE INTERFACE WITH ALL:

DS1302 RTC Module ↔ ESP32 Wiring:

- VCC → 3.3V (Power supply)
- GND → GND (Ground)
- CLK → GPIO14 (Clock)
- DAT → GPIO19 (Data I/O)
- RST → GPIO5 (Chip Enable - CE)

Code:

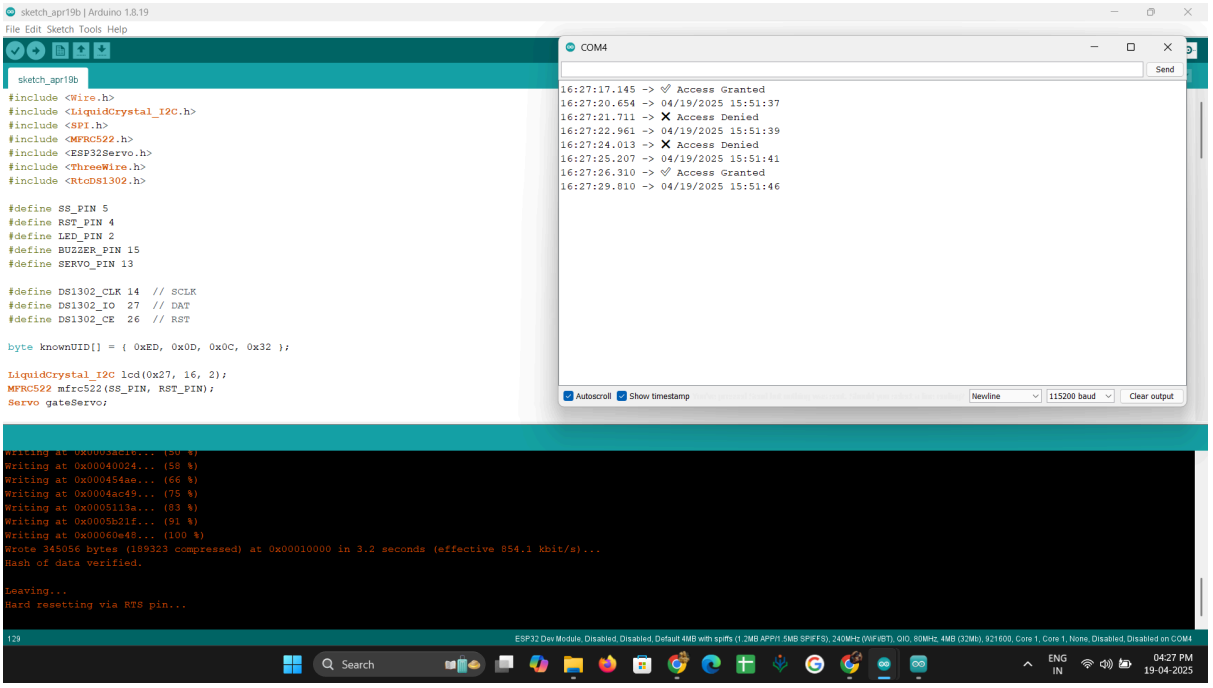
```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <SPI.h>
#include <MFRC522.h>
#include <ESP32Servo.h>
#include <ThreeWire.h>
#include <RtcDS1302.h>

#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 };
LiquidCrystal_I2C lcd(0x27, 16, 2);
MFRC522 mfrc522(SS_PIN, RST_PIN);
Servo gateServo;
ThreeWire myWire(DS1302_IO, DS1302_CLK, DS1302_CE); // IO, SCLK, CE
RtcDS1302<ThreeWire> Rtc(myWire);
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°
  // RTC Setup
  Rtc.Begin();
  RtcDateTime customTime(2025, 4, 19, 15, 45, 00); // YYYY, MM, DD, HH, MM, SS
  Rtc.SetDateTime(customTime);
 if (!Rtc.IsDateTimeValid()) {
    Serial.println("RTC lost confidence in the DateTime!");
  }
 if (!Rtc.GetIsRunning()) {
    Serial.println("RTC was not running. Starting it now...");
    Rtc.SetIsRunning(true);
  }
// Print initial time
  RtcDateTime now = Rtc.GetDateTime();
  printDateTime(now);
}
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
  }
 // Get current time from RTC
 RtcDateTime now = Rtc.GetDateTime();
```

```
 printDateTime(now);  // Print the current time from RTC
 mfrc522.PICC_HaltA();
  mfrc522.PCD_StopCrypto1();
  delay(1000);
}
void printDateTime(const RtcDateTime dt) {
  char datestring[20];
  snprintf_P(datestring,
             sizeof(datestring),
             PSTR("%02u/%02u/%04u %02u:%02u:%02u"),
             dt.Month(),
             dt.Day(),
             dt.Year(),
             dt.Hour(),
             dt.Minute(),
             dt.Second());
  Serial.println(datestring);
}
```

Output:



REFERENCE LINK: WhatsApp Video 2025-04-19 at 4.29.08 PM.mp4

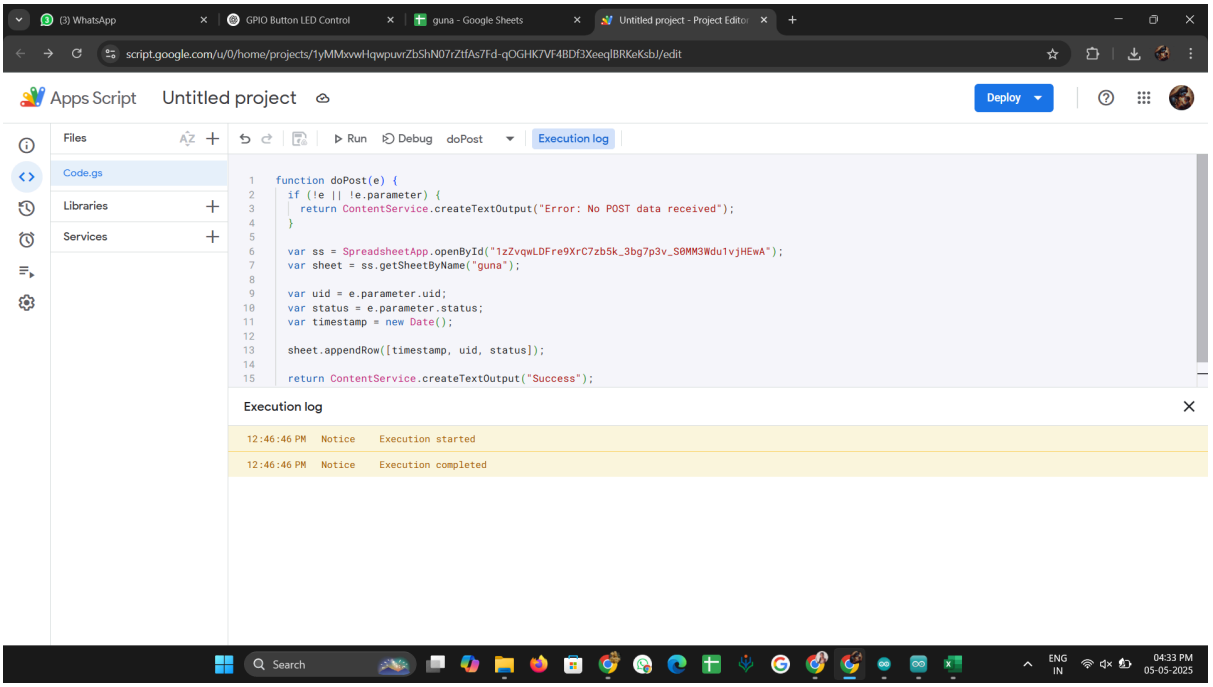
1. Google Apps Script (Web App):

This script receives data (e.g., UID, timestamp) via HTTP POST and logs it to a Google Sheet.

Create Script:

1. Go to https://script.google.com
2. Click New Project
3. Paste the following code:

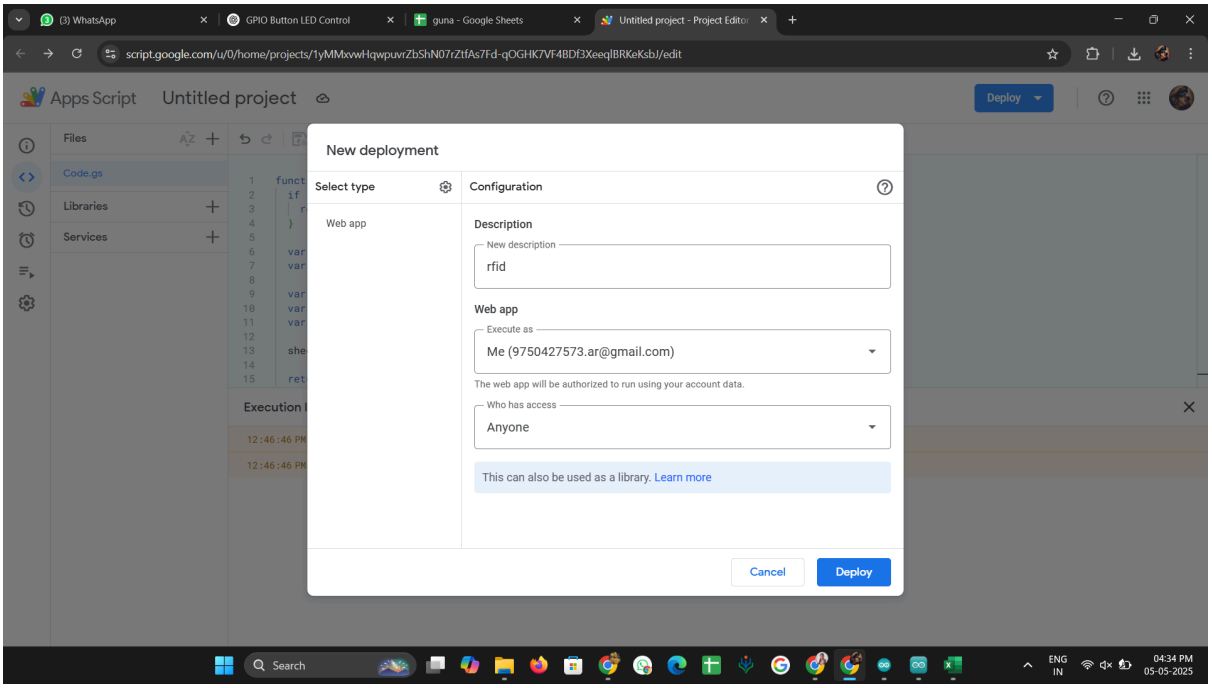
```
function doPost(e) {

var ss = SpreadsheetApp.openById("YOUR_SPREADSHEET_ID")

var sheet = ss.getSheetByName("Sheet1");
// Get data from POST request
var uid = e.parameter.uid;
var status = e.parameter.status;
var timestamp = new Date();
// Append data to sheet
sheet.appendRow([timestamp, uid, status]);
return ContentService.createTextOutput("Success");
}
```

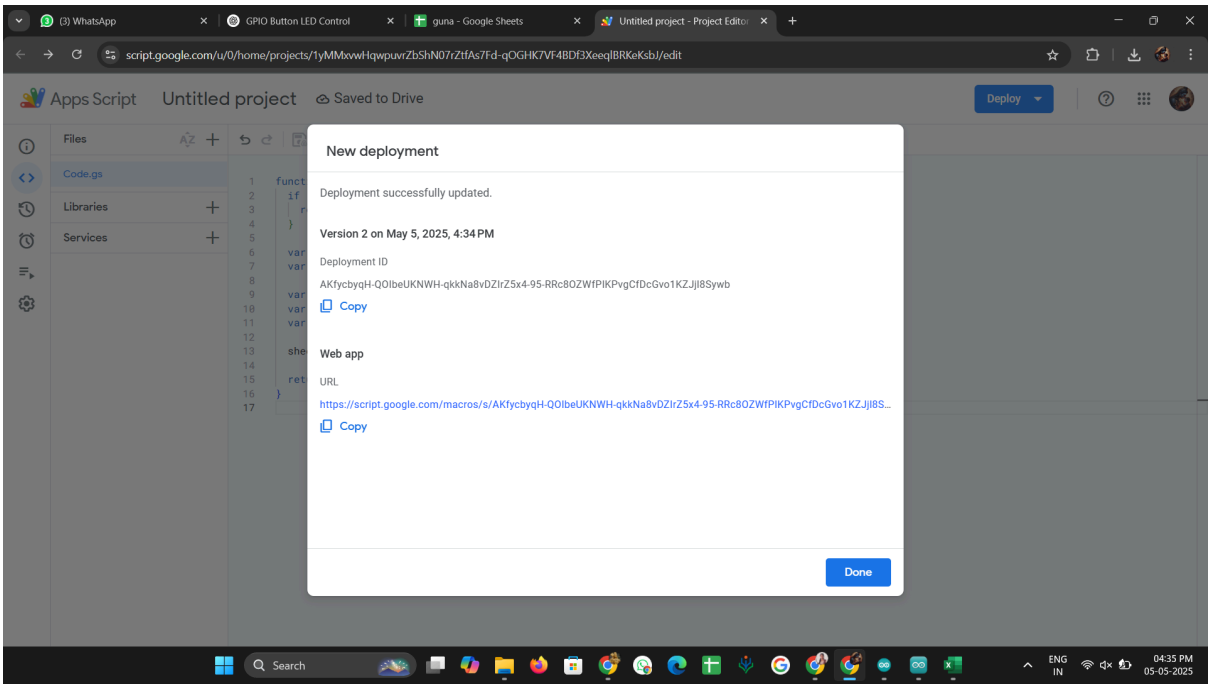


4. Click File > Save, name the project.
5. Replace `"YOUR_SPREADSHEET_ID"` with your actual Sheet ID (from URL).
6. Go to Deploy > Manage Deployments > New deployment:
    - Select "Web app"
    - Execute as: Me
    - Access: Anyone

7. Click Deploy, then Authorize, and copy the Web App URL. Example:

    https://script.google.com/macros/s/AKf.../exec





2. ESP32 Arduino Code (HTTP POST Request)

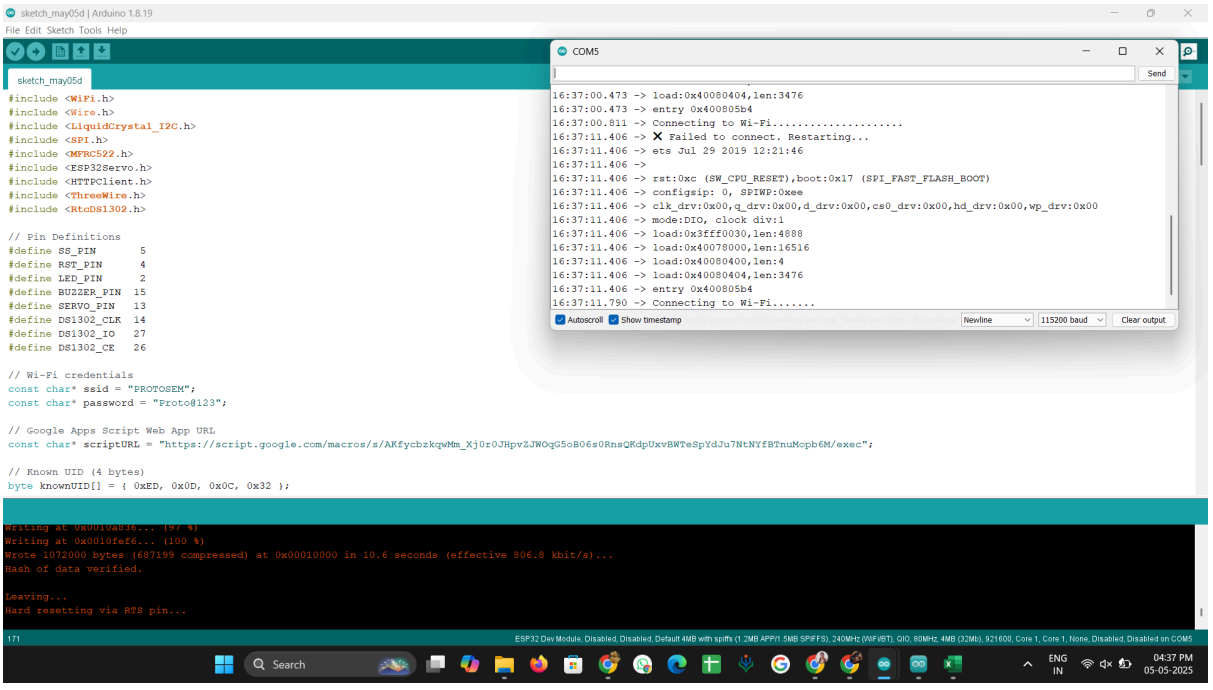
```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <SPI.h>
#include <MFRC522.h>
#include <ESP32Servo.h>
#include <ThreeWire.h>
#include <RtcDS1302.h>
#include <WiFi.h>
#include <HTTPClient.h>
// ---------------------- Pins & Constants ----------------------
#define SS_PIN 5
#define RST_PIN 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 };

// ---------------------- 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";             // <-- Change this
const char* password = "YOUR_WIFI_PASSWORD";     // <-- Change this
const String scriptURL =
"https://script.google.com/macros/s/XXXXXXXXXXXX/exec"; // <-- Your
Google Script 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);

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

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

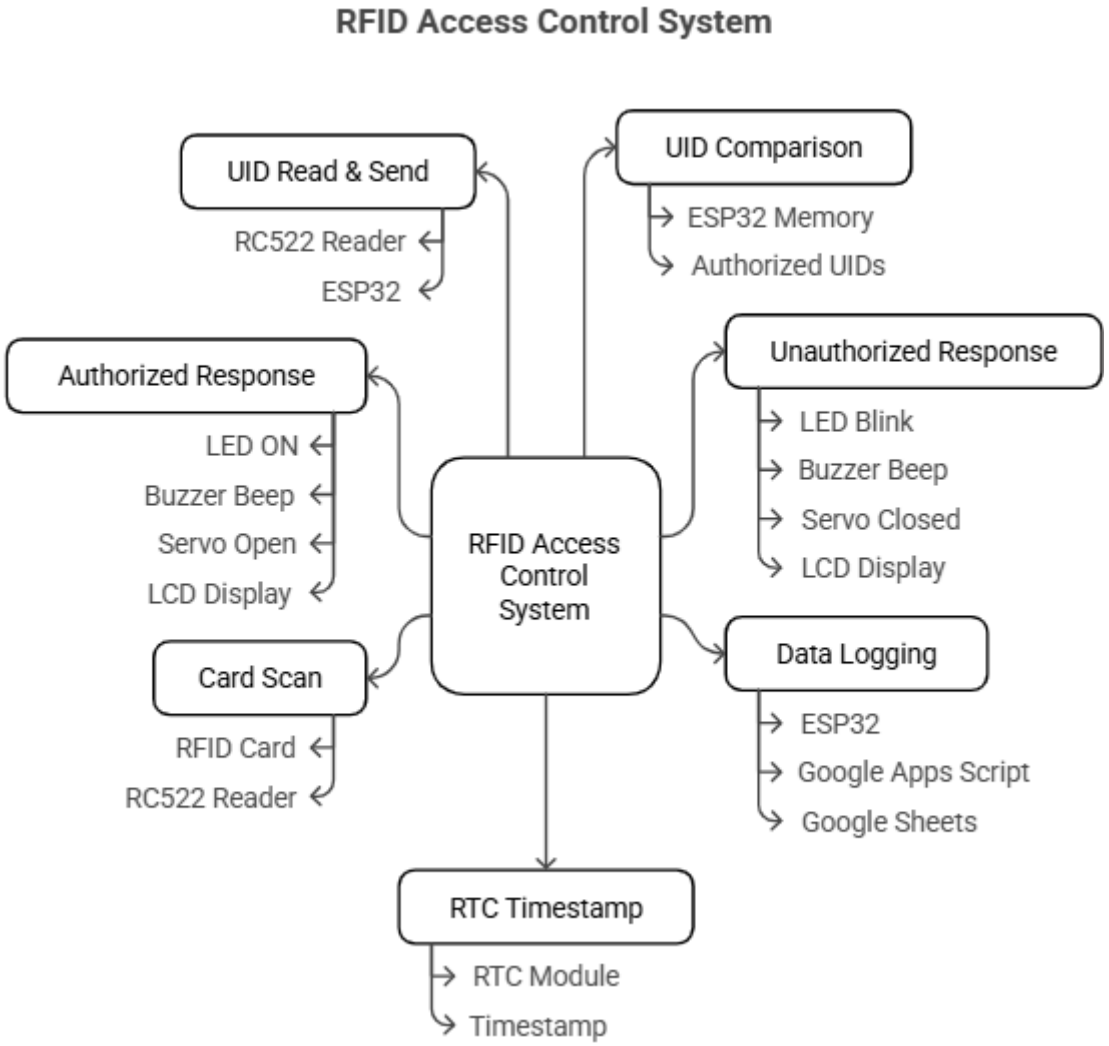
```
  // Wi-Fi connect
  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWiFi connected.");
}
// ---------------------- Main Loop ----------------------
void loop() {
  if (!mfrc522.PICC_IsNewCardPresent() ||
!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(); // Optional: Make UID uppercase

  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);
    delay(500);
    digitalWrite(LED_PIN, LOW);
    digitalWrite(BUZZER_PIN, LOW);
    delay(3000);
    gateServo.write(0);
    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);
    }
    sendToGoogleSheet(uidStr, "Denied");
  }

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

  mfrc522.PICC_HaltA();
  mfrc522.PCD_StopCrypto1();
  delay(1000);
}
// ---------------------- Send to Google Sheet----------------------
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: ");
```

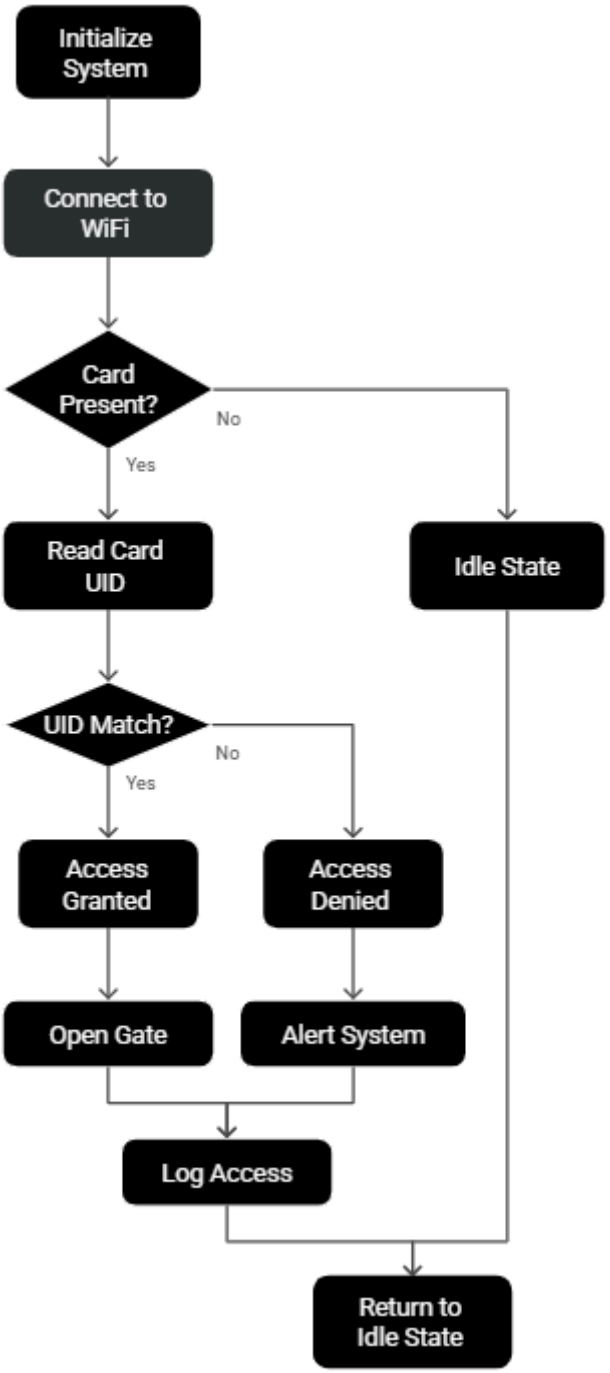
```
      Serial.println(httpCode);
    } else {
      Serial.print("Failed to send. Code: ");
      Serial.println(httpCode);
    }
    http.end();
  } else {
    Serial.println("WiFi not connected");
  }
}
// ---------------------- Print RTC Time ----------------------
void printDateTime(const RtcDateTime& 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);
}
```



Block Diagram:



Flowchart:


Initialize System
Connect to WiFi
Card Present?
No
Yes
Read Card UID
Idle State
UID Match?
No
Yes
Access Granted
Access Denied
Open Gate
Alert System
Log Access
Return to Idle State