# NFC Tag Detection and Google Spreadsheet Logging

Project to detect NFC tag card, read its content, distinguish between users, and log details on Google Spreadsheet.

## Contents:

- Introduction
- Hardware Components
  - NFC Module with ESP8266
  - 16x2 LCD with I2C Interface
  - Nfc Tag
- Software Setup
  - Installing Libraries
- Configuring ESP8266
- Reading NFC Tag
- User Identification
- Logging Details
- Google Spreadsheet Integration
- Conclusion

## Introduction

This project aims to create an attendance system using Near Field Communication (NFC) technology integrated with an ESP8266 microcontroller, a 16x2 LCD with I2C interface, and a Mifare Classic Tag. The system will detect the NFC tag, read its content, determine if the ID belongs to a specific person (e.g., Erfan or Hossein), and then log the attendance details onto a Google Spreadsheet.

## Hardware Components

Details of the hardware components used in the project.

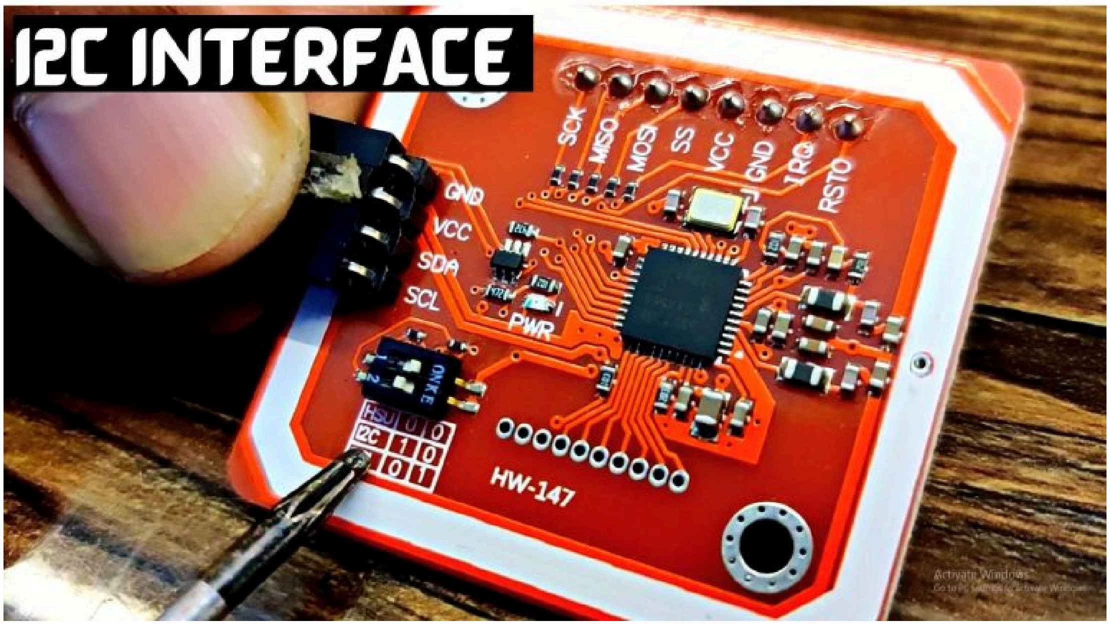
### NFC Module with ESP8266

The NFC module is used to read NFC tags. It is integrated with the NODEMCU ESP8266 Wifi microcontroller module. The key specifications of the ESP8266 module are:

- Model: ESP8266MOD
- ISM 2.4GHz
- PA +25dBm
- Supports 802.11b/g/n Wi-Fi standards

### The PN532 NFC RFID Module V3

The reason I selected the PN532 RFID module is that it's a low-cost and low-power RFID module. And it supports multiple interfaces HSU " High Speed UART", I2C, and SPI. In my previous article, I have already explained how to use all three interfaces



In this article, We are going to use the I2C interface because our 16*2 Display module is also an I2C-supported device, so using the I2C interface we can reduce the wiring. So, using only two pins on the Nodemcu ESP8266, we can communicate with both modules.

## 16x2 LCD with I2C Interface

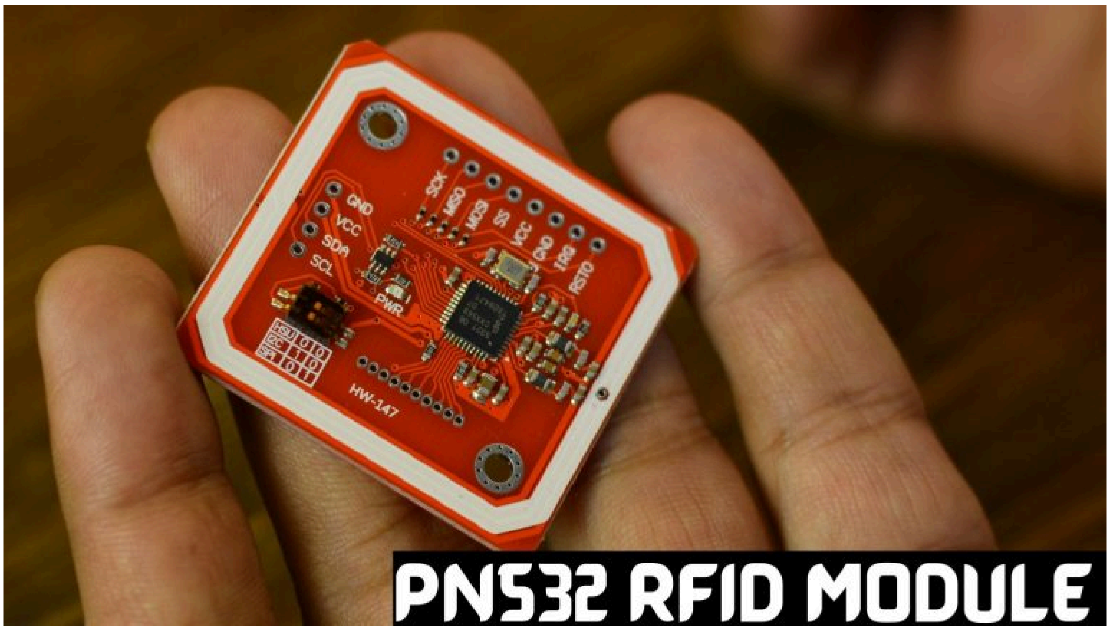
The 16x2 C0421A LCD module is used to display messages and information. It is interfaced using the I2C protocol through the NXD PCF85741 I2C module, which reduces the number of pins required for connection to the microcontroller.

## Additional Components

- Two breadboards for creating the circuit.
- Jumper wires for connections between the components.
- A USB Type-A cable for powering the ESP8266 and for serial communication with a computer.

## NFC Tag

We have used NFC Tag in This Project.



the Properties of this tag are as follows

- Writable = Yes
- Mifare Sectors : 16
- Mifare Size : 1024 bytes
- Max Transceceive Length 253 bytes

### Technologies :

- Ndef
- NfcA
- MifareClassic

## Software Setup

Configuration and setup of required software components.

## Installing Libraries

To use the libraries required for this project, follow these steps:

1. **Adafruit GFX Library**
    - Open the Arduino IDE.
    - Go to **Sketch** > **Include Library** > **Manage Libraries**.
    - In the Library Manager, search for 'Adafruit GFX'.
    - Click on the library and then click the 'Install' button.
    - **Usage:** Provides common graphics functions for various displays, including drawing shapes and text.
2. **Adafruit SSD1306 Library**
    - Open the Arduino IDE.
    - Go to **Sketch** > **Include Library** > **Manage Libraries**.
    - In the Library Manager, search for 'Adafruit SSD1306'.
    - Click on the library and then click the 'Install' button.
    - **Usage:** Controls SSD1306-based OLED displays, essential for showing messages on the screen.
3. **Arduino Library**
    - This library is included with the Arduino IDE, so no additional installation is required.
    - **Usage:** Provides basic functions for working with the Arduino hardware.
4. **ESP8266WiFi Library**
    - Open the Arduino IDE.
    - Go to **Sketch** > **Include Library** > **Manage Libraries**.
    - In the Library Manager, search for 'ESP8266WiFi'.
    - Click on the library and then click the 'Install' button.
    - **Usage:** Connects the ESP8266 to Wi-Fi networks, enabling internet communication.
5. **HTTPSRedirect Library**
    - Download from GitHub.
    - Extract the downloaded zip file and place it in the `libraries` folder of your Arduino sketchbook directory.
    - **Usage:** Makes HTTPS requests to a server, necessary for secure data transmission to Google Sheets.
6. **Display Library**
    - If custom, place it in the `libraries` folder of your Arduino sketchbook directory.
    - **Usage:** Handles display-related functions, providing high-level functions for the display module.
7. **Wire Library**
    - This library is included with the Arduino IDE, so no additional installation is required.
    - **Usage:** Facilitates I2C communication between the Arduino and I2C devices like the NFC module and LCD.

8. **PN532_I2C, PN532, NfcAdapter Libraries**

   - Open the Arduino IDE.
   - Go to **Sketch** > **Include Library** > **Manage Libraries**.
   - In the Library Manager, search for 'PN532' and install the related libraries.
   - **Usage:** Interfaces with the PN532 NFC/RFID module, providing functions to read and write NFC tags.

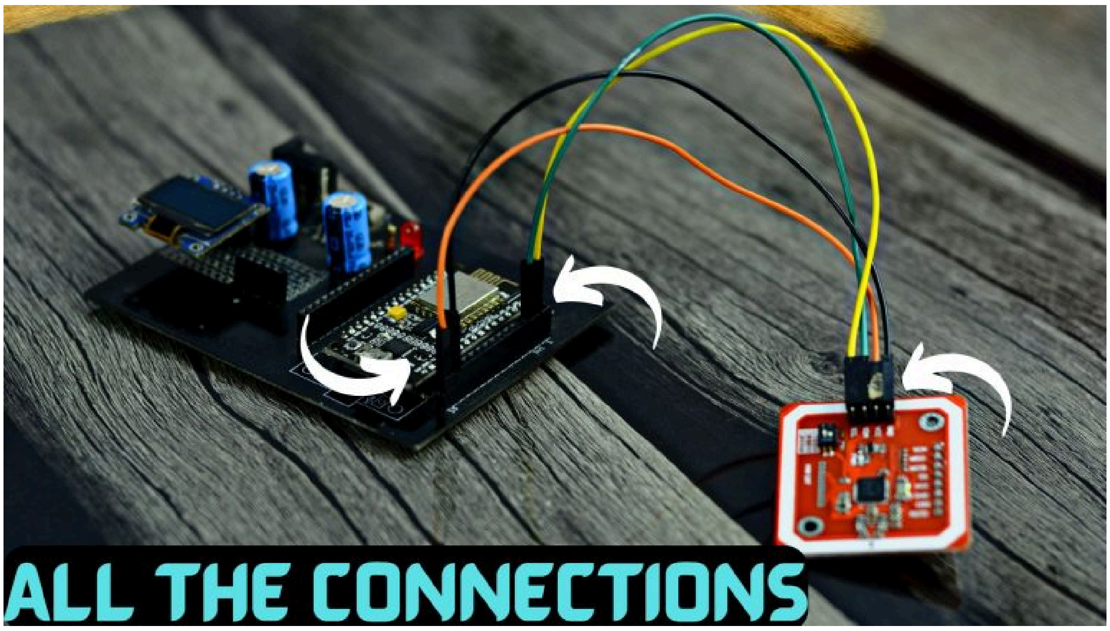
9. **LiquidCrystal I2C Library**

   - Open the Arduino IDE.
   - Go to **Sketch** > **Include Library** > **Manage Libraries**.
   - In the Library Manager, search for 'LiquidCrystal I2C'.
   - Click on the library and then click the 'Install' button.
   - **Usage:** Controls I2C LCD displays, simplifying the interaction with the 16x2 LCD module.

## Configuring Modules

### Configuring ESP8266 and nfc Tag

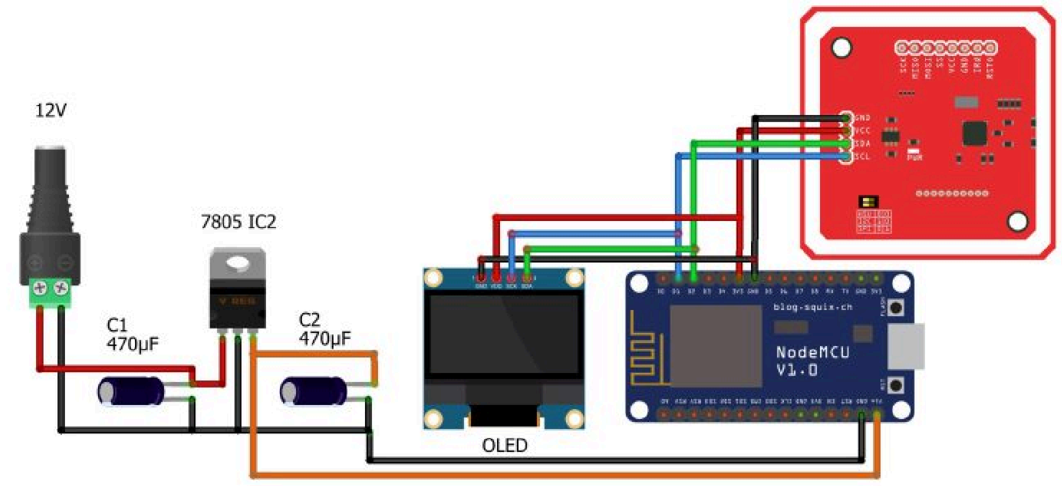
Steps to configure ESP8266 for NFC tag detection.



First we will connect the RFID with ESP8266:

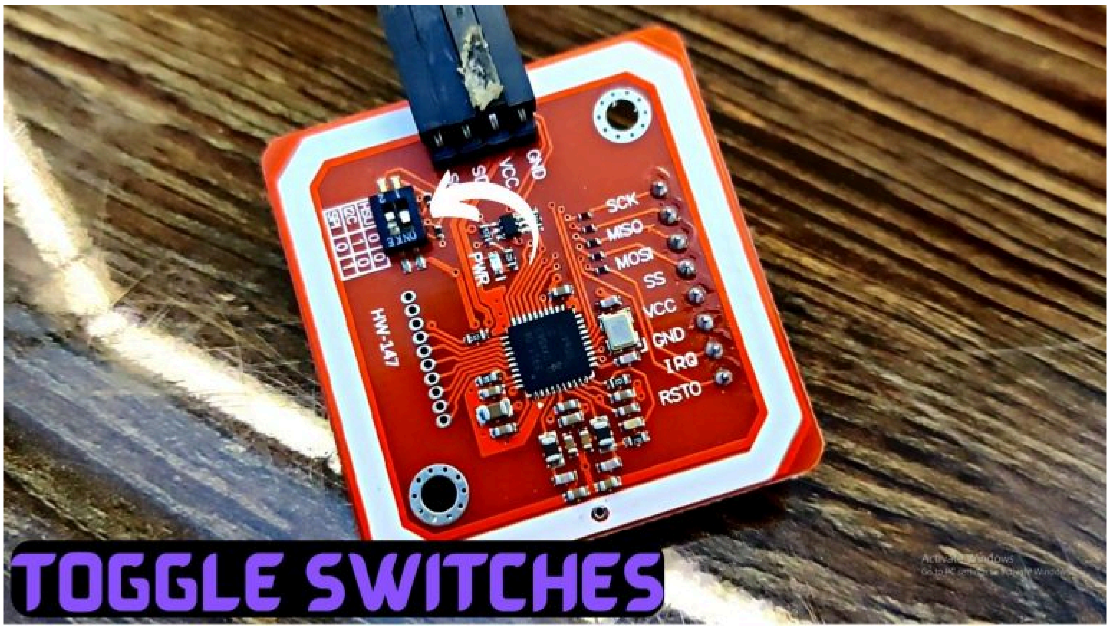
- Connect the SCL pin of the RFID with D1 pin of the ESP8266
- Connect the SDA pin of the RFID with D2 pin of the ESP8266
- Connect the VCC of the RFID with 3.3V of the ESP8266
- Connect the GND of the RFID with GND of the ESP8266

### Configuring LCD Display Module



connect the OLED with ESP8266

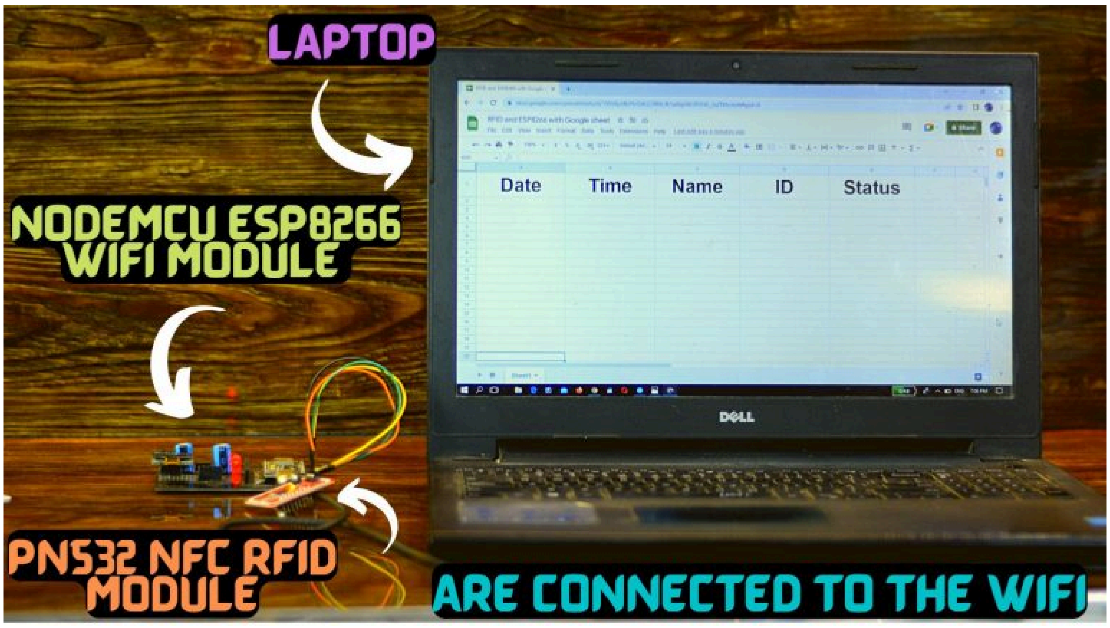
- Connect the SCL pin of the OLED with D1 pin of the ESP8266
- Connect the SDA pin of the OLED with D2 pin of the ESP8266
- Connect the VCC of the OLED with 3.3V of the ESP8266
- Connect the VCC of the OLED with 3.3V of the ESP8266



One more thing that I would like to talk about is, as I said earlier the PN532 supports multiple interfaces which are UART, I2C, and SPI. For selecting any of these interfaces you will need to accordingly adjust the toggle switches. Since I am using an I2C interface so that's why channel 1 is ON and channel 2 is OFF.

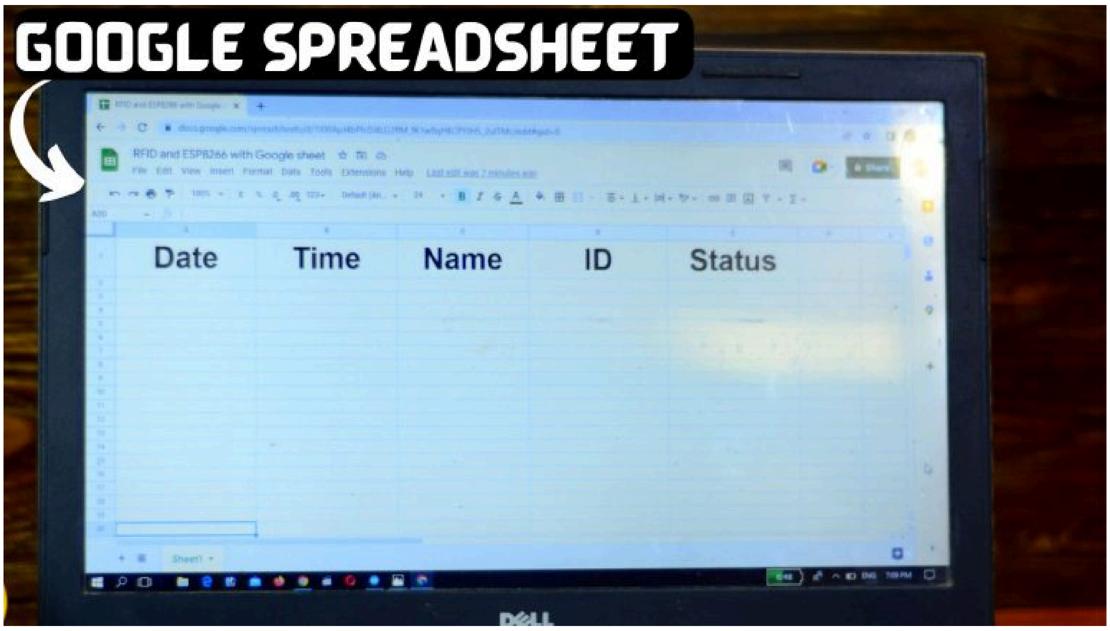
As usual, before, I am going to explain how to set up your Google Spreadsheet and how to connect it with the Nodemcu ESP8266 WiFi module. First, let's watch this project in action.

## Practical demonstration

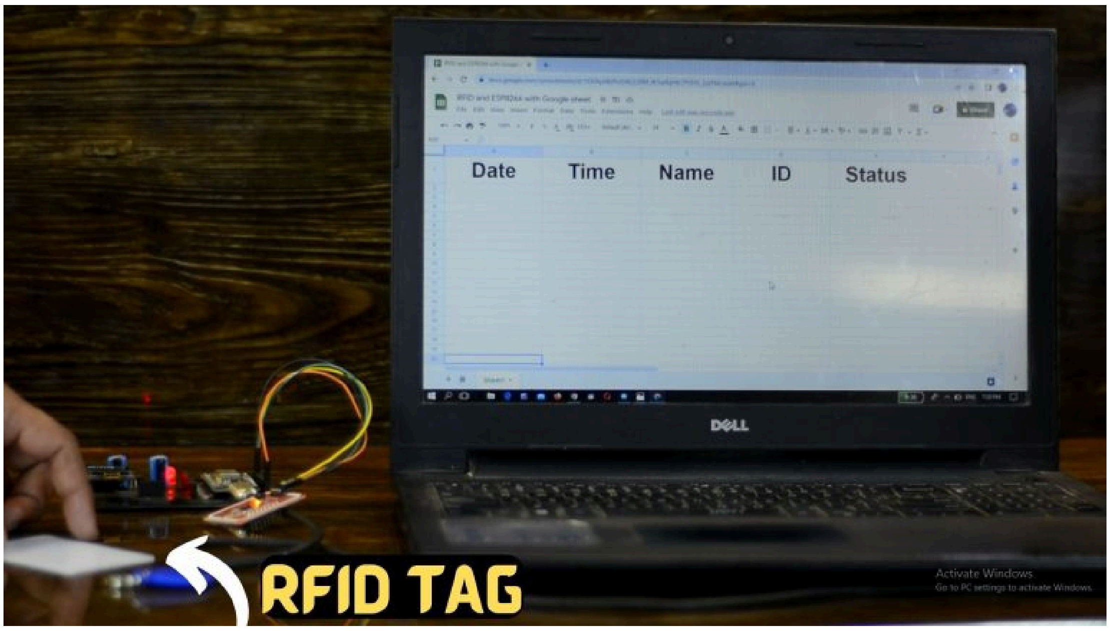


Right now, our Nodemcu ESP8266 WiFi Module and Laptop are connected to the WiFi. Let me also tell you, it's not necessary to use the same WiFi network, you can use different WiFi networks. It's an IoT project, you can monitor the employees or students in/out of time from any part of the world.

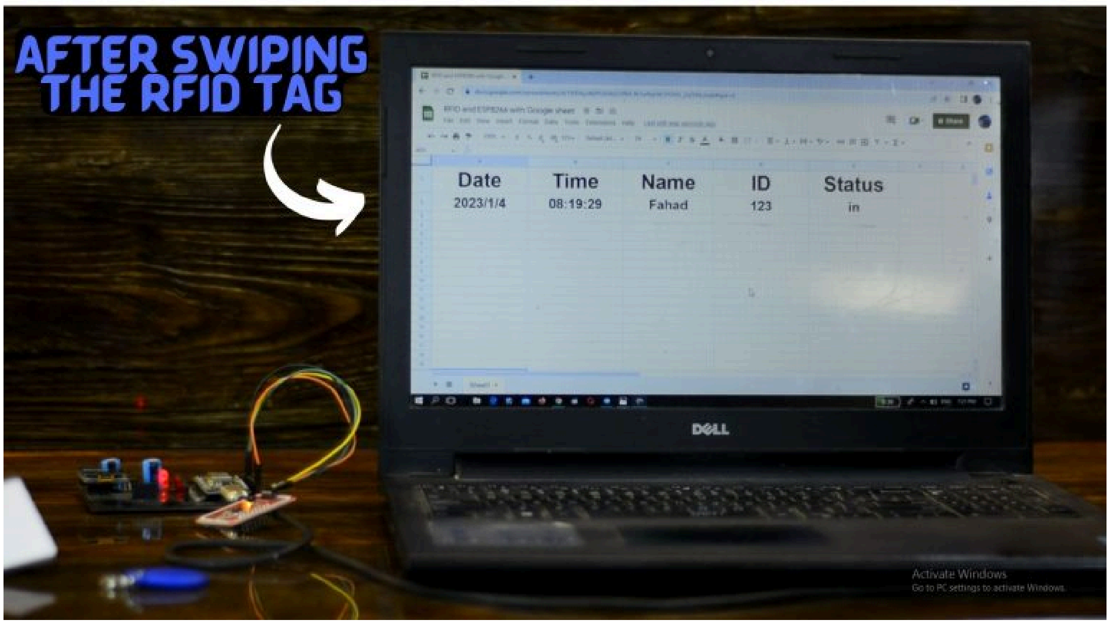
## GoogleSheet setup



Anyway, on the Google spreadsheet, I can monitor the Date, Time, Name of the employee or a student, the ID, and the in/out status



We are going to swipe the RFID tags and you will practically see how it works.



| Date | Time | Name | ID | Status |
|---|---|---|---|---|
| 2023/1/4 | 08:19:29 | Fahad | 123 | in |

The LED display module is optional, if you remove this it won't have any effect on the project. But it's good to use a display, because without the display you will be confused and you won't be sure if the card is successfully swiped.

## Reading NFC Tag

Code snippets for reading NFC tag content.

## User Identification

Algorithm for distinguishing between users based on tag ID.

## Logging Details

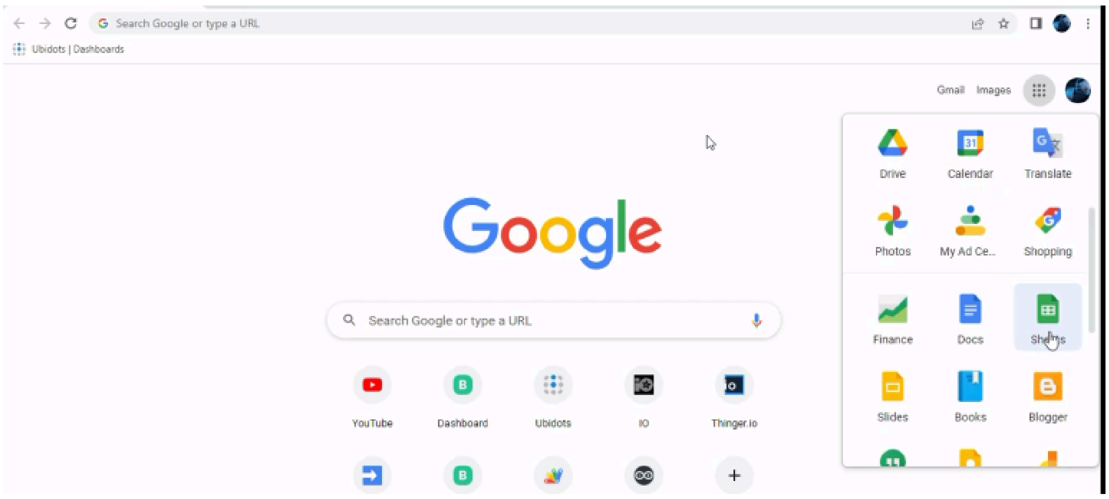
Implementation of logging details functionality.

## Google Spreadsheet Setup

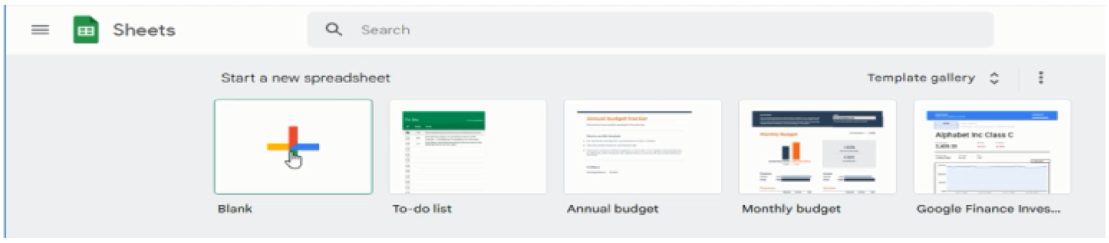
Guide for integrating with Google Spreadsheet for logging.

You will need to carefully set up your Google Spreadsheet or Google sheet for logging the required data. If you miss anything, you won't be able to get it connected to your Nodemcu ESP8266 WiFi Module. I am not using Google Spreadsheet for the first time, I have used it in some other projects too. So, for more information on Google spreadsheets, watch my previous articles. Right now you can follow the same exact steps.

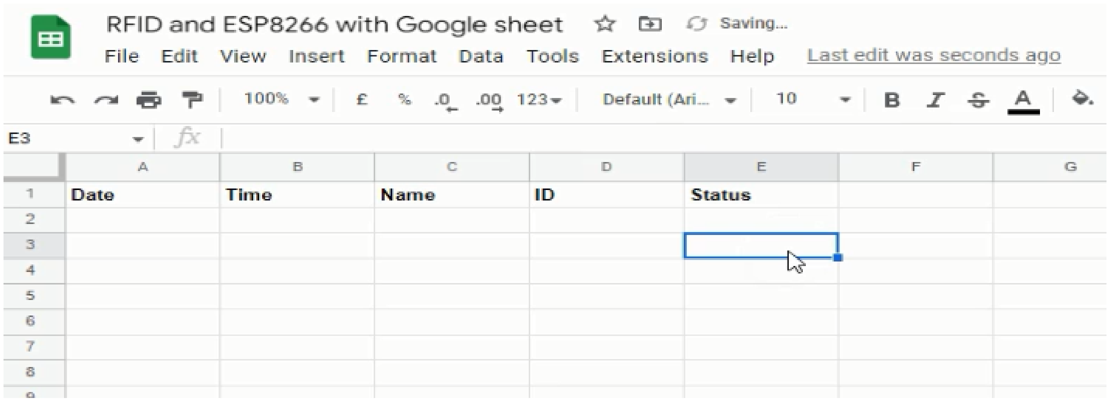
Click on the Google apps and click on the Sheet



Then click on the blank sheet

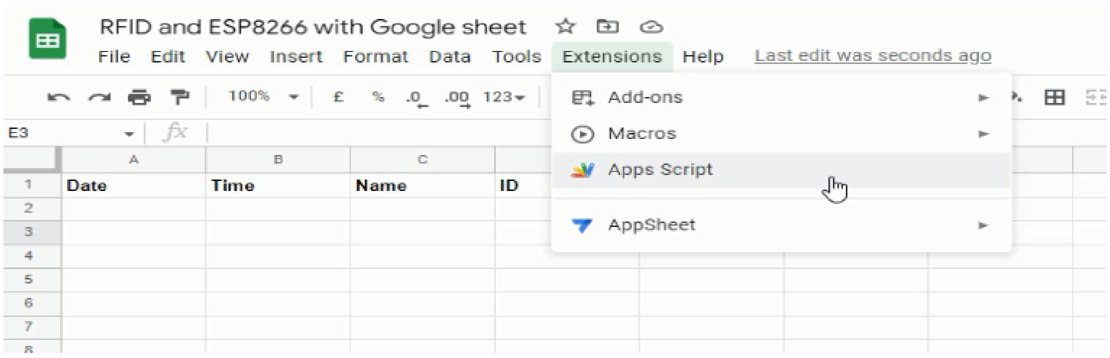


Then give a name to the sheet and enter the date, time, id, and status.

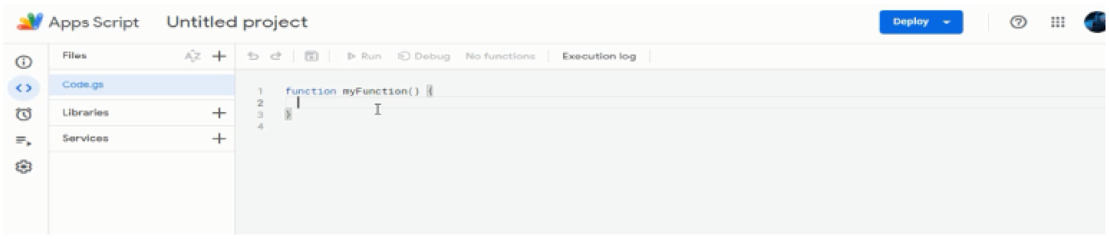


| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Date | Time | Name | ID | Status |

Click on the extension and click on the App Script


RFID and ESP8266 with Google sheet
File Edit View Insert Format Data Tools Extensions Help
Last edit was seconds ago
Add-ons
Macros
Apps Script
AppSheet
Date
Time
Name
ID

Then the App script will open



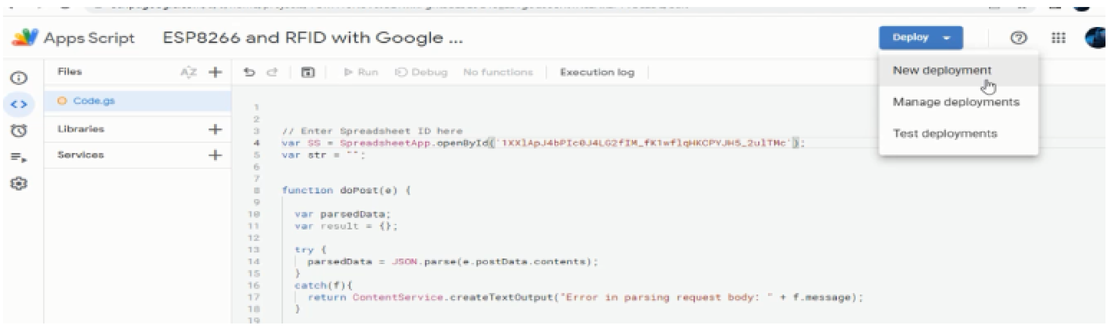
```
function myFunction() {

}
```

Paste the following code:

```
In [ ]: // Enter Spreadsheet ID here
        var SS = SpreadsheetApp.openById('1XFghoRmZzNLlYEkPPEEs1gjxM0aUsz_5HDJ58aQ_cKg')
        var str = "";


        function doPost(e) {

          var parsedData;
          var result = {};

          try {
            parsedData = JSON.parse(e.postData.contents);
          }
          catch(f){
            return ContentService.createTextOutput("Error in parsing request body: " + f
          }

          if (parsedData !== undefined){
            var flag = parsedData.format;
            if (flag === undefined){
              flag = 0;
            }

            var sheet = SS.getSheetByName(parsedData.sheet_name); // sheet name to publi
            var dataArr = parsedData.values.split(","); // creates an array of the value

            var date_now = Utilities.formatDate(new Date(), "CST", "yyyy/MM/dd"); // get
            var time_now = Utilities.formatDate(new Date(), "CST", "hh:mm:ss a"); // get

            var value0 = dataArr [0]; // value0 from Arduino code
            var value1 = dataArr [1]; // value1 from Arduino code
            var value2 = dataArr [2]; // value2 from Arduino code


            // read and execute command from the "payload_base" string specified in Ardu
            switch (parsedData.command) {
```

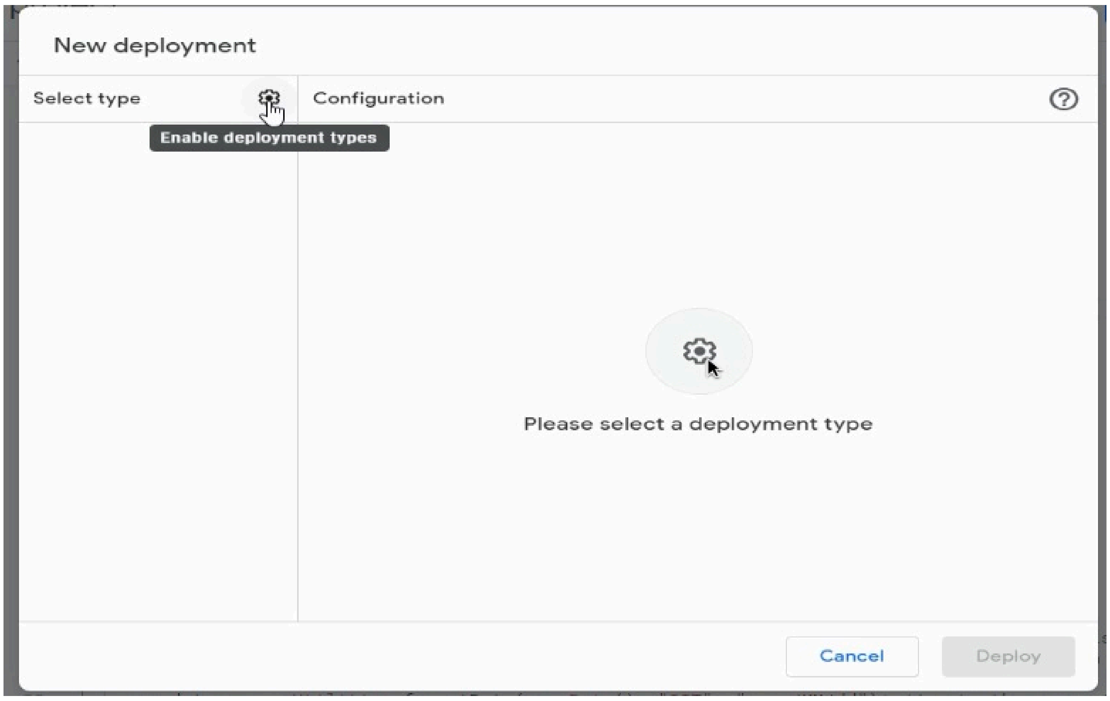
```
      case "insert_row":

         sheet.insertRows(2); // insert full row directly below header text

         //var range = sheet.getRange("A2:D2");              // use this to inse
         //range.insertCells(SpreadsheetApp.Dimension.ROWS); // use this to inse

         sheet.getRange('A2').setValue(date_now); // publish current date to cel
         sheet.getRange('B2').setValue(time_now); // publish current time to cel
         sheet.getRange('C2').setValue(value0);   // publish value0 from Arduinc
         sheet.getRange('D2').setValue(value1);   // publish value1 from Arduinc
         sheet.getRange('E2').setValue(value2);   // publish value2 from Arduinc

         str = "Success"; // string to return back to Arduino serial console
         SpreadsheetApp.flush();
         break;

      case "append_row":

         var publish_array = new Array(); // create a new array

         publish_array [0] = date_now; // add current date to position 0 in publ
         publish_array [1] = time_now; // add current time to position 1 in publ
         publish_array [2] = value0;   // add value0 from Arduino code to positi
         publish_array [3] = value1;   // add value1 from Arduino code to positi
         publish_array [4] = value2;   // add value2 from Arduino code to positi

         sheet.appendRow(publish_array); // publish data in publish_array after

         str = "Success"; // string to return back to Arduino serial console
         SpreadsheetApp.flush();
         break;

    }

    return ContentService.createTextOutput(str);
  } // endif (parsedData !== undefined)

  else {
    return ContentService.createTextOutput("Error! Request body empty or in inco
  }
}
```

Then click on Deploy and click on the New Development

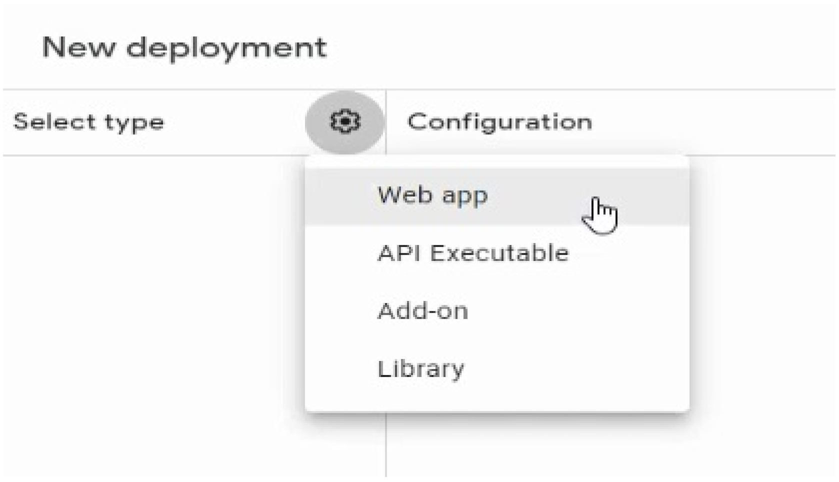


```
// Enter Spreadsheet ID here
var SS = SpreadsheetApp.openById('1XXlApJ4bPIc0J4LG2fIM_fK1wflqHKCPYJH5_2ulTMc');
var str = "";


function doPost(e) {

  var parsedData;
  var result = {};

  try {
    parsedData = JSON.parse(e.postData.contents);
  }
  catch(f){
    return ContentService.createTextOutput("Error in parsing request body: " + f.message);
  }
```

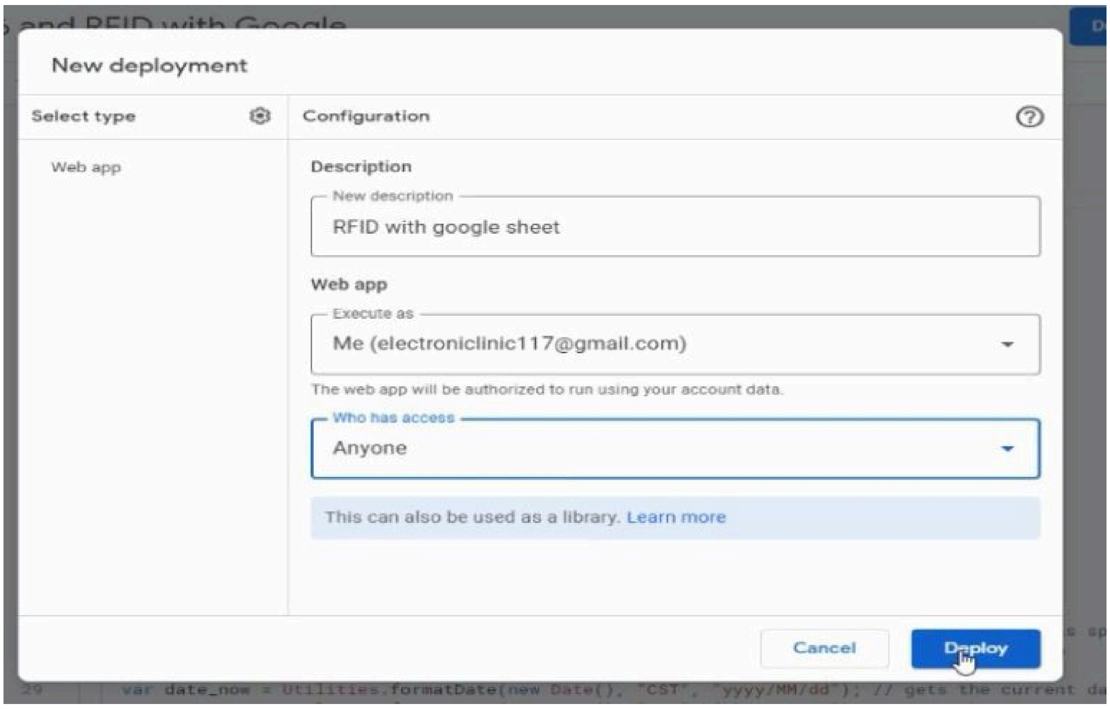
Then click on the enable deployment types.



Now select deploy as web

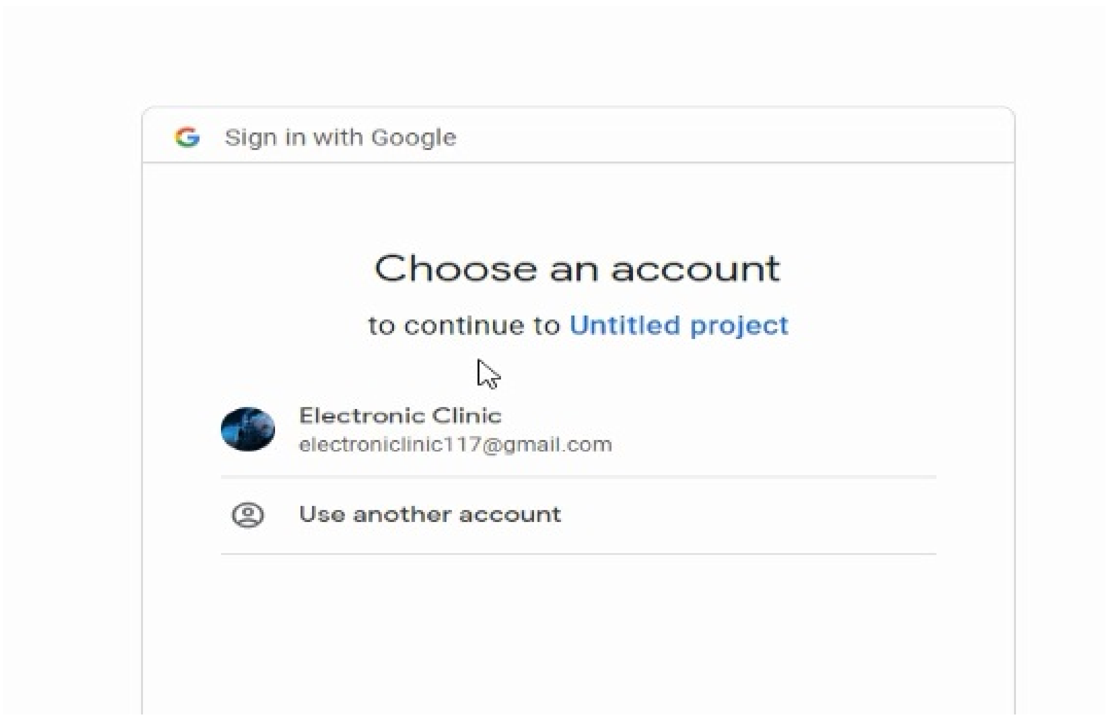


Now select anyone and click on the deploy



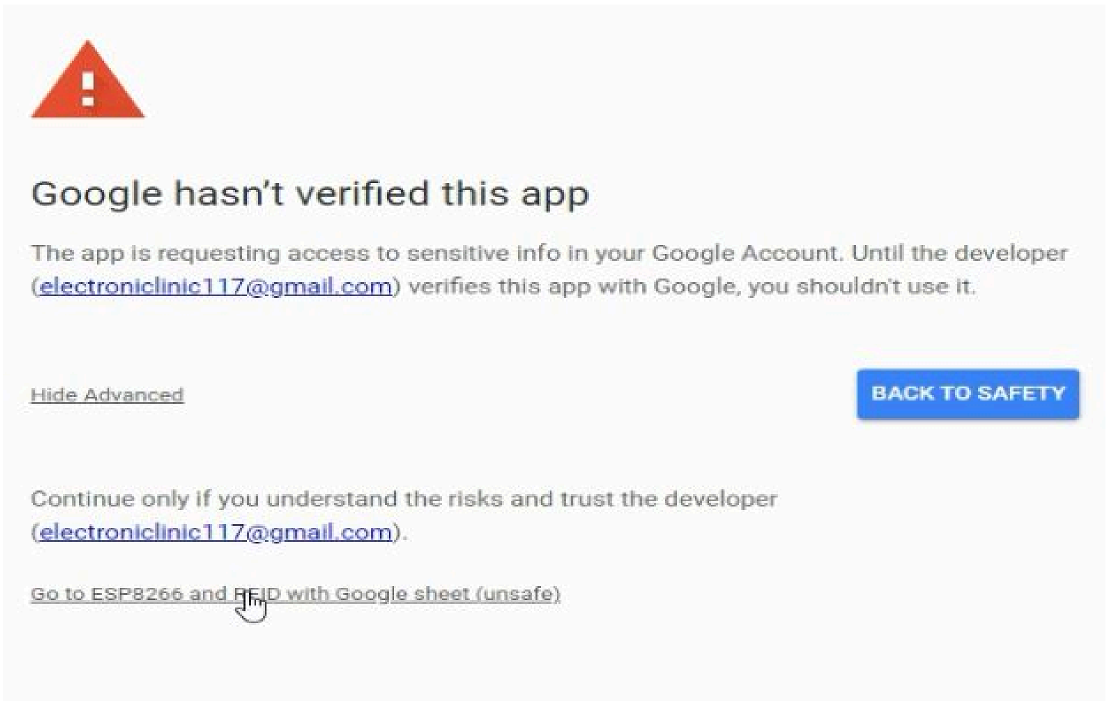
Description
New description: RFID with google sheet
Web app
Execute as: Me (electronicclinic117@gmail.com)
The web app will be authorized to run using your account data.
Who has access: Anyone
This can also be used as a library. Learn more

Now select the account in which you have created the Google sheet.



Choose an account
to continue to Untitled project
Electronic Clinic
electroniclinic117@gmail.com

Now click on the go to ESP8266 and RFID with Google sheet unsafe

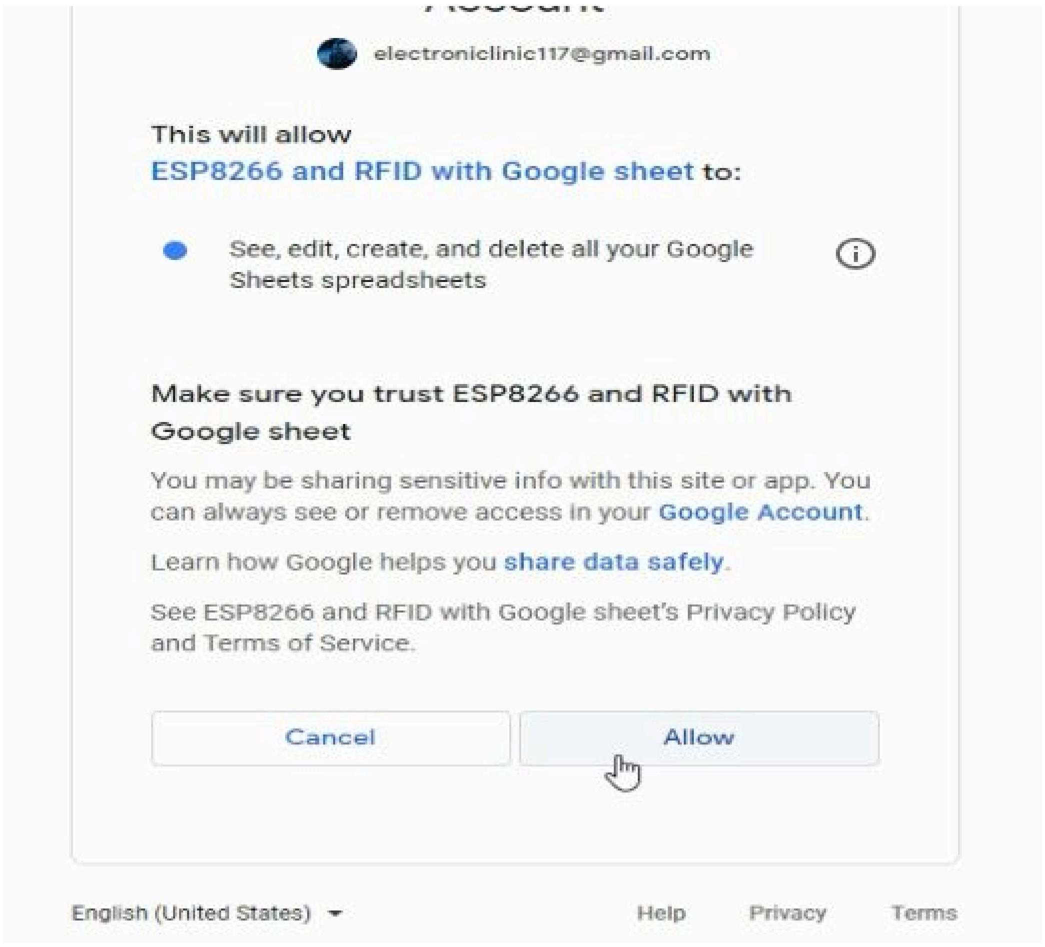


Google hasn't verified this app

The app is requesting access to sensitive info in your Google Account. Until the developer (electroniclinic117@gmail.com) verifies this app with Google, you shouldn't use it.

Continue only if you understand the risks and trust the developer (electroniclinic117@gmail.com).

Now click on the allow button to give access to the account



electroniclinic117@gmail.com

This will allow
ESP8266 and RFID with Google sheet to:

- See, edit, create, and delete all your Google Sheets spreadsheets

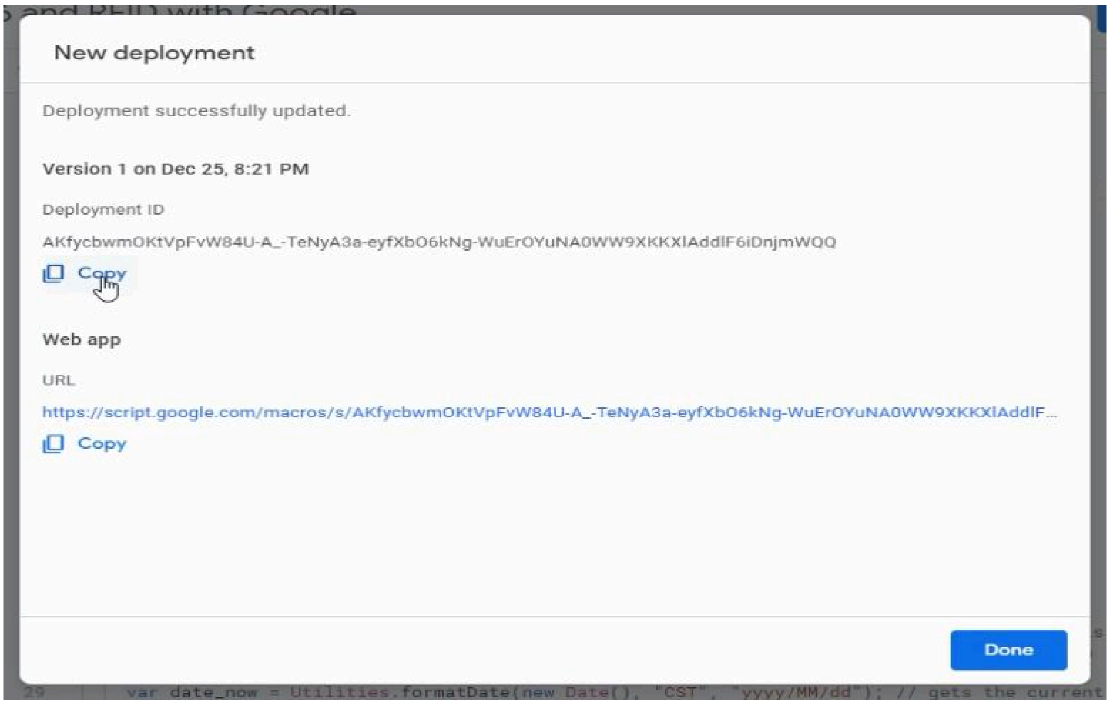
Make sure you trust ESP8266 and RFID with Google sheet

You may be sharing sensitive info with this site or app. You can always see or remove access in your Google Account.

Learn how Google helps you share data safely.

See ESP8266 and RFID with Google sheet's Privacy Policy and Terms of Service.

Then copy the deployment id



New deployment

Deployment successfully updated.

Version 1 on Dec 25, 8:21 PM

Deployment ID

AKfycbwmOKtVpFvW84U-A_-TeNyA3a-eyfXbO6kNg-WuErOYuNA0WW9XKKXlAddlF6iDnjmWQQ

Copy

Web app

URL

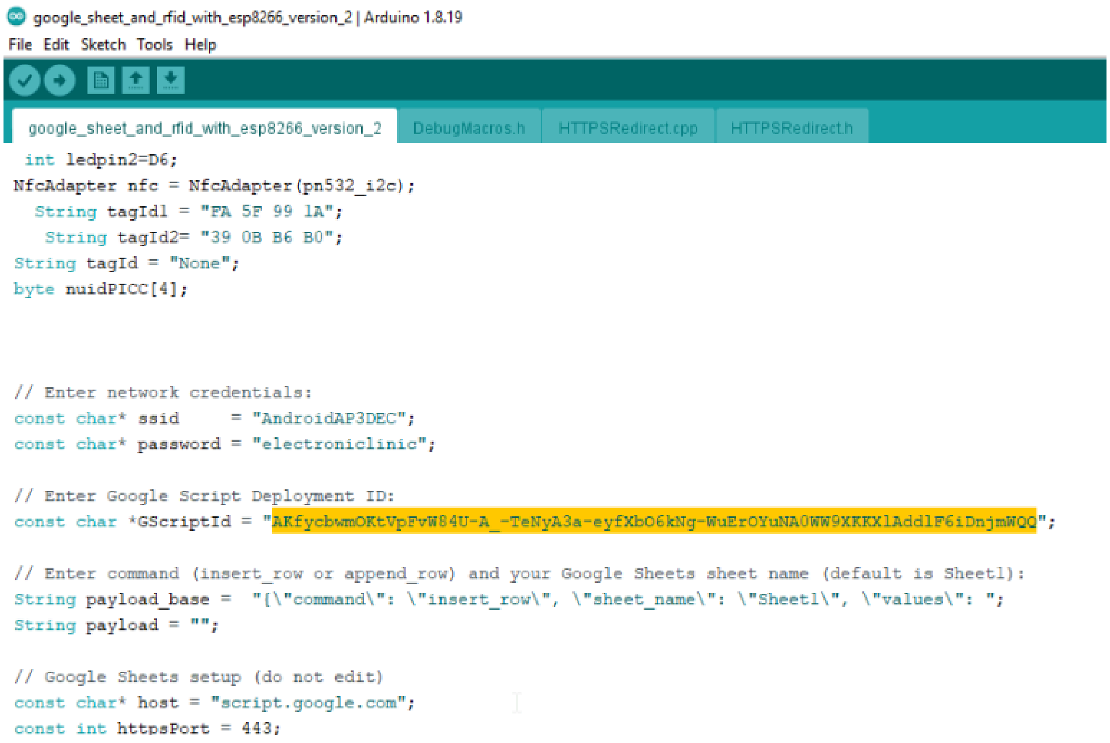
https://script.google.com/macros/s/AKfycbwmOKtVpFvW84U-A_-TeNyA3a-eyfXbO6kNg-WuErOYuNA0WW9XKKXlAddlF...

Copy

Done

```
var date_now = Utilities.formatDate(new Date(), "CST", "yyyy/MM/dd"); // gets the current
```

Paste the deployment id in the code



google_sheet_and_rfid_with_esp8266_version_2 | Arduino 1.8.19

File Edit Sketch Tools Help
google_sheet_and_rfid_with_esp8266_version_2
DebugMacros.h
HTTPSRedirect.cpp
HTTPSRedirect.h

```
 int ledpin2=D6;
NfcAdapter nfc = NfcAdapter(pn532_i2c);
  String tagId1 = "FA 5F 99 1A";
   String tagId2= "39 0B B6 B0";
String tagId = "None";
byte nuidPICC[4];



// Enter network credentials:
const char* ssid     = "AndroidAP3DEC";
const char* password = "electroniclinic";

// Enter Google Script Deployment ID:
const char *GScriptId = "AKfycbwmOKtVpFvW84U-A_-TeNyA3a-eyfXbO6kNg-WuErOYuNA0WW9XKKXlAddlF6iDnjmWQQ";

// Enter command (insert_row or append_row) and your Google Sheets sheet name (default is Sheet1):
String payload_base =  "{\"command\": \"insert_row\", \"sheet_name\": \"Sheet1\", \"values\": ";
String payload = "";

// Google Sheets setup (do not edit)
const char* host = "script.google.com";
const int httpsPort = 443;
```

## Project code :

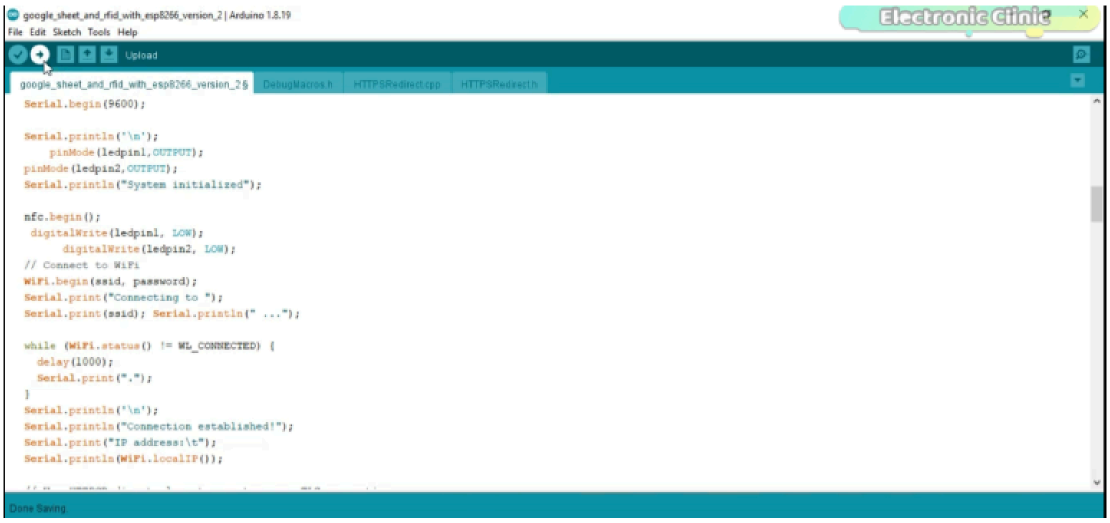
In [ ]:

```
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include "HTTPSRedirect.h"
#include "Display.h"
#include <Wire.h>
#include <PN532_I2C.h>
#include <PN532.h>
#include <NfcAdapter.h>
```

```
//added
#include <LiquidCrystal_I2C.h>

PN532_I2C pn532_i2c(Wire);
// int ledpin1 = D5;
// int ledpin2 = D6;
NfcAdapter nfc = NfcAdapter(pn532_i2c);
// Primitive Values
String tagId1 = "69 C9 D2 35";
String tagId2= "39 0B B6 B0";
String tagId = "None";
byte nuidPICC[4];
bool isPend = false;
bool IN = false;
bool IN2 = false;

// Enter network credentials:
const char* ssid     = "Lucerfan";
const char* password = "erfan_m1383";

// Enter Google Script Deployment ID:
const char *GScriptId = "AKfycby3o0voFb1YA4voIMcL9oC3iDWXYdOscl-w0u8fOQLO3EohFxk

// Enter command (insert_row or append_row) and your Google Sheets sheet name (d
String payload_base =  "{\"command\": \"insert_row\", \"sheet_name\": \"Sheet1\"
String payload = "";

// Google Sheets setup (do not edit)
const char* host = "script.google.com";
const int httpsPort = 443;
const char* fingerprint = "";
String url = String("/macros/s/") + GScriptId + "/exec";
HTTPSRedirect* client = nullptr;

// Declare variables that will be published to Google Sheets
String user = "";
String id = "";
String enter ="";

Display display(isPend);

void setup() {

  Serial.begin(9600);

  Serial.println('\n');
  display.initialize();
  // pinMode(ledpin1,OUTPUT);
  // pinMode(ledpin2,OUTPUT);
  Serial.println("System initialized");

  nfc.begin();
  // digitalWrite(ledpin1, LOW);
  // digitalWrite(ledpin2, LOW);
  // Connect to WiFi
  WiFi.begin(ssid, password);
  Serial.print("Connecting to ");
  Serial.print(ssid); Serial.println(" ...");

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println('\n');
  Serial.println("Connection established!");
  Serial.print("IP address:\t");
  Serial.println(WiFi.localIP());

  // Use HTTPSRedirect class to create a new TLS connection
  client = new HTTPSRedirect(httpsPort);
  client->setInsecure();
  client->setPrintResponseBody(true);
  client->setContentTypeHeader("application/json");

  Serial.print("Connecting to ");
  Serial.println(host);

  // Try to connect for a maximum of 5 times
  bool flag = false;
  for (int i=0; i<5; i++) {
    int retval = client->connect(host, httpsPort);
    if (retval == 1) {
       flag = true;
       Serial.println("Connected");
       break;
    }
    else
      Serial.println("Connection failed. Retrying...");
  }
  if (!flag){
    Serial.print("Could not connect to server: ");
    Serial.println(host);
    return;
  }
  delete client;    // delete HTTPSRedirect object
  client = nullptr; // delete HTTPSRedirect object

}


void loop() {

    if(readNFC()) {

        if (tagId==tagId1) {
            if (IN) {
                // if( digitalRead(ledpin1) == 1) {
                // digitalWrite(ledpin1, LOW);
                enter="";
                user="Erfan";
                id="123";
                enter="out";
                IN = false;
                updatesheet(user,id, enter);
                display.successOUT(user, enter);
                tagId = "";
                delay(500);
              // }

            } else {
                // if( digitalRead(ledpin1) == 0) {
                // digitalWrite(ledpin1, HIGH);
                user="Erfan";
                id="123";
                enter="in";
                IN = true;
                updatesheet(user,id, enter);
                tagId = "";
                display.successIN(user, enter);
                delay(500);
              // }
            }
        } else {
            if (IN2) {
                // if( digitalRead(ledpin2) == 1) {
                // digitalWrite(ledpin2, LOW);
                user="Hossein";
                id="456";
                enter="out";
                IN2 = false;
                updatesheet(user,id, enter);
                display.successOUT(user, enter);
                tagId = "";
                delay(500);
              // }
            } else {
                // if( digitalRead(ledpin2) == 0) {
                // digitalWrite(ledpin2, HIGH);
                user="";
                id="";
                enter="";
                user="Hossein";
                id="456";
                enter="in";
                IN2 = true;
                updatesheet(user,id, enter);
                display.successIN(user, enter);
                tagId = "";
                delay(500);
              // }
            }
        }
    }
    else
        display.pend();
}

void updatesheet(String user, String id, String enter) {

  static bool flag = false;
  if (!flag) {
    client = new HTTPSRedirect(httpsPort);
    client->setInsecure();
    flag = true;
    client->setPrintResponseBody(true);
    client->setContentTypeHeader("application/json");
  }
  if (client != nullptr) {
    if (!client->connected()) {
      client->connect(host, httpsPort);
    }
  }
  else {
    Serial.println("Error creating client object!");
  }

  // Create json object string to send to Google Sheets
  payload = payload_base + "\"" + user + "," + id + "," + enter + "\"}";

  // Publish data to Google Sheets
  Serial.println("Publishing data...");
  Serial.println(payload);
  if(client->POST(url, host, payload)) {
    // do stuff here if publish was successful
  }
  else {
    // do stuff here if publish was not successful
    Serial.println("Error while connecting");
  }

  // a delay of several seconds is required before publishing again
  delay(1000);
  }

bool readNFC() {
  if (nfc.tagPresent()) {
    NfcTag tag = nfc.read();
    tag.print();
    tagId = tag.getUidString();
    Serial.println("tag details :");
    String content= "";
    byte myByte[] = {0x00, 0xFF};
    // const unsigned char* myByte_ptr = myByte;
    Serial.println();
    tag.getUid(myByte, tag.getUidLength());
    for (int i = 0 ; i < tag.getUidLength() ; i++) {
        Serial.println(myByte[i]);
    }
    content.toUpperCase();
    tag.print();
    Serial.println("Tag id");
    Serial.println(tagId);
    delay(1000);
    return true;
  }
  return false;
}
```

```
  Cell In[1], line 11
    //added
    ^
SyntaxError: invalid syntax
```

The Google sheet setup is completed and finally, you can click on the Upload button.


google_sheet_and_rfid_with_esp8266_version_2 | Arduino 1.8.19
File Edit Sketch Tools Help
Upload
google_sheet_and_rfid_with_esp8266_version_2 §
DebugMacros.h
HTTPSRedirect.cpp
HTTPSRedirect.h
Electronic Clinic
Done Saving.

```
Serial.begin(9600);

Serial.println('\n');
    pinMode(ledpin1,OUTPUT);
pinMode(ledpin2,OUTPUT);
Serial.println("System initialized");

nfc.begin();
 digitalWrite(ledpin1, LOW);
      digitalWrite(ledpin2, LOW);
// Connect to WiFi
WiFi.begin(ssid, password);
Serial.print("Connecting to ");
Serial.print(ssid); Serial.println(" ...");

while (WiFi.status() != WL_CONNECTED) {
  delay(1000);
  Serial.print(".");
}
Serial.println('\n');
Serial.println("Connection established!");
Serial.print("IP address:\t");
Serial.println(WiFi.localIP());
```

Make sure you keep the main programming file and all the header files in the same folder. For the Nodemcu ESP8266 WiFi board installation and libraries installation, you can read my getting started articles on the Google spreadsheet, PN532 RFID module, and SSD1306 Oled Display Module.

## Conclusion

Summary and final thoughts.