Roll Number: 22BCE502(Azim Baldiwala),
22BCE543(Tanmay Trivedi).

Project Title: Attendance System using RDIF.

Abstract: This project scans a RFID card, reads card's data and sends data to Google sheet along with date and time at which card is scanned.

Components used:

1] NodeMcu 1.0

2] RFID-RC522 Module

3] Bread Board & Jumper wires

4] Buzzer

Working:

1] When the tag is brought close to the reader, the reader generates an electromagnetic field. This causes electrons to move through the tag's antenna and subsequently powers the chip.

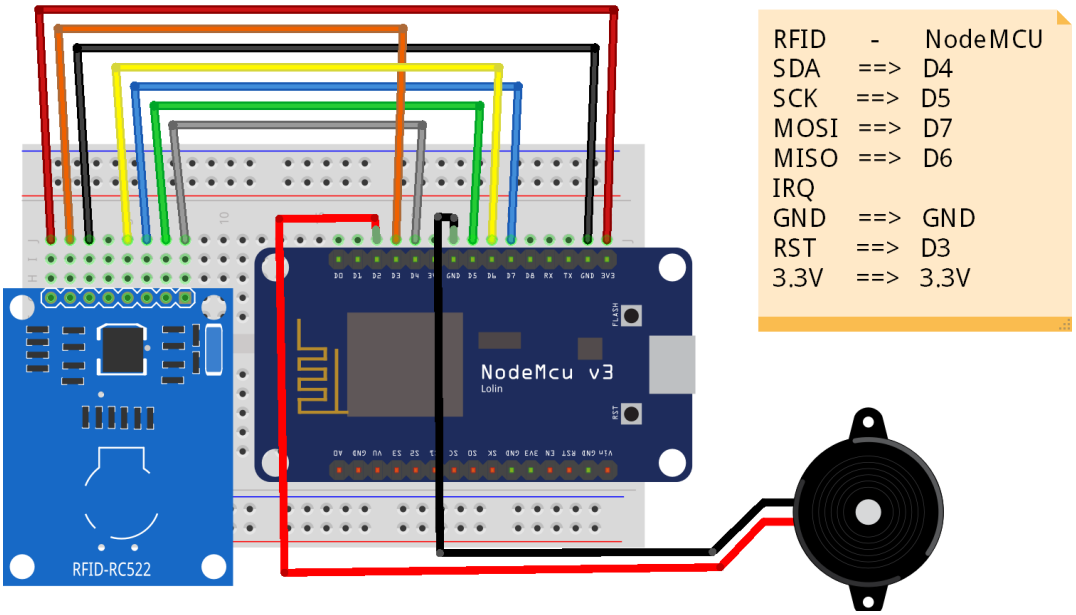
The chip then responds by sending its stored information back to the reader in the form of another radio signal. This is called a backscatter. The reader detects and interprets this backscatter and sends the data to a microcontroller(NodeMcu in our case).

2] Once the Microcontroller receives the data, it process the data, the data is a string of 16 characters which is identity of user. Now the microcontroller processes the data by adding data and time along with the card data.

3] Microcontroller sends a get request to google sheet url.

4] The request send by microcontroller is handled by javaScript function which adds the parameters(name, date and time) into google sheet.

Circuit Diagram:



Code to Write data on card:

```
#include <SPI.h>

#include <MFRC522.h>
```

```
constexpr uint8_t RST_PIN = D3;     // Configurable, see typical pin
layout above
constexpr uint8_t SS_PIN = D4;     // Configurable, see typical pin
layout above

MFRC522 mfrc522(SS_PIN, RST_PIN); // Instance of the class
MFRC522::MIFARE_Key key;

/* Set the block to which we want to write data */
/* Be aware of Sector Trailer Blocks */
int blockNum = 2;
/* Create an array of 16 Bytes and fill it with data */
/* This is the actual data which is going to be written into the card
*/
// byte blockData [16] = {"AZIM_22BCE502"};
byte blockData [16] = {"TANMAY_22BCE543"};

/* Create another array to read data from Block */
/* Legthn of buffer should be 2 Bytes more than the size of Block (16
Bytes) */
byte bufferLen = 18;
byte readBlockData[18];

MFRC522::StatusCode status;

void setup()
{
```

```
  /* Initialize serial communications with the PC */
  Serial.begin(9600);
  /* Initialize SPI bus */
  SPI.begin();
  /* Initialize MFRC522 Module */
  mfrc522.PCD_Init();
  Serial.println("Scan a MIFARE 1K Tag to write data...");
}

void loop()
{
  /* Prepare the ksy for authentication */
  /* All keys are set to FFFFFFFFFFFFh at chip delivery from the
factory */
  for (byte i = 0; i < 6; i++)
  {
    key.keyByte[i] = 0xFF;
  }
  /* Look for new cards */
  /* Reset the loop if no new card is present on RC522 Reader */
  if ( ! mfrc522.PICC_IsNewCardPresent())
  {
    return;
  }

  /* Select one of the cards */
  if ( ! mfrc522.PICC_ReadCardSerial())
```

```
  {
    return;
  }
  Serial.print("\n");
  Serial.println("**Card Detected**");
  /* Print UID of the Card */
  Serial.print(F("Card UID:"));
  for (byte i = 0; i < mfrc522.uid.size; i++)
  {
    Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
    Serial.print(mfrc522.uid.uidByte[i], HEX);
  }
  Serial.print("\n");
  /* Print type of card (for example, MIFARE 1K) */
  Serial.print(F("PICC type: "));
  MFRC522::PICC_Type piccType = mfrc522.PICC_GetType(mfrc522.uid.sak);
  Serial.println(mfrc522.PICC_GetTypeName(piccType));

   /* Call 'WriteDataToBlock' function, which will write data to the
block */
   Serial.print("\n");
   Serial.println("Writing to Data Block...");
   WriteDataToBlock(blockNum, blockData);

   /* Read data from the same block */
   Serial.print("\n");
   Serial.println("Reading from Data Block...");
```

```
   ReadDataFromBlock(blockNum, readBlockData);
   /* If you want to print the full memory dump, uncomment the next
line */
   //mfrc522.PICC_DumpToSerial(&(mfrc522.uid));

   /* Print the data read from block */
   Serial.print("\n");
   Serial.print("Data in Block:");
   Serial.print(blockNum);
   Serial.print(" --> ");
   for (int j=0 ; j<16 ; j++)
   {
     Serial.write(readBlockData[j]);
   }
   Serial.print("\n");
}



void WriteDataToBlock(int blockNum, byte blockData[])
{
  /* Authenticating the desired data block for write access using Key
A */
  status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A,
blockNum, &key, &(mfrc522.uid));
  if (status != MFRC522::STATUS_OK)
  {
```

```
    Serial.print("Authentication failed for Write: ");
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
  }
  else
  {
    Serial.println("Authentication success");
  }


  /* Write data to the block */
  status = mfrc522.MIFARE_Write(blockNum, blockData, 16);
  if (status != MFRC522::STATUS_OK)
  {
    Serial.print("Writing to Block failed: ");
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
  }
  else
  {
    Serial.println("Data was written into Block successfully");
  }

}

void ReadDataFromBlock(int blockNum, byte readBlockData[])
{
```

```
  /* Authenticating the desired data block for Read access using Key A
*/
  status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A,
blockNum, &key, &(mfrc522.uid));

  if (status != MFRC522::STATUS_OK)
  {
     Serial.print("Authentication failed for Read: ");
     Serial.println(mfrc522.GetStatusCodeName(status));
     return;
  }
  else
  {
    Serial.println("Authentication success");
  }

  /* Reading data from the Block */
  status = mfrc522.MIFARE_Read(blockNum, readBlockData, &bufferLen);
  if (status != MFRC522::STATUS_OK)
  {
    Serial.print("Reading failed: ");
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
  }
  else
  {
    Serial.println("Block was read successfully");
```

```
  }


}
```

Code for sending data to google sheet:

```
#include <SPI.h>
#include <MFRC522.h> //13.56 MHz
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <ESP8266WiFiMulti.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>
#include <WiFiClientSecureBearSSL.h>

const uint8_t fingerprint[20] = {0x14, 0xE1, 0x36, 0x8E, 0xEA, 0xF3,
0x30, 0x6C, 0x1E, 0xDB, 0x9E, 0x43, 0x1E, 0xEF, 0xAA, 0x94, 0x1C,
0x9E, 0xA7, 0x74};

// 14 E1 36 8E EA F3 30 6C 1E DB 9E 43 1E EF AA 94 1C 9E A7 74

#define RST_PIN  D3     // Configurable, see typical pin layout above
#define SS_PIN   D4     // Configurable, see typical pin layout above
#define BUZZER   D2     // Configurable, see typical pin layout above
```

```
MFRC522 mfrc522(SS_PIN, RST_PIN);  // Instance of the class
MFRC522::MIFARE_Key key;
ESP8266WiFiMulti WiFiMulti;
MFRC522::StatusCode status;

/* Be aware of Sector Trailer Blocks */
int blockNum = 2;

/* Create another array to read data from Block */
/* Legthn of buffer should be 2 Bytes more than the size of Block (16
Bytes) */
byte bufferLen = 18;
byte readBlockData[18];

String data2;
/* Google sheet url  */
const String data1 =
"https://script.google.com/macros/s/AKfycbzvyV3YUf0d3EEcVUlKUNNgUcfrQL
D6LazFsRBxZpENXZUOFylDHK5QVATXSJzAcD2Z/exec?name=";

void setup()
{
  /* Initialize serial communications with the PC */
  Serial.begin(9600);
  // Serial.setDebugOutput(true);

  Serial.println();
```

```
  Serial.println();
  Serial.println();

  for (uint8_t t = 4; t > 0; t--)
  {
    Serial.printf("[SETUP] WAIT %d...\n", t);
    Serial.flush();
    delay(1000);
  }

  WiFi.mode(WIFI_STA);

  /* Put your WIFI Name and Password here */
  Serial.printf("ready to connet\n");
/* wifi creds */
  WiFiMulti.addAP("TEST", "test123456");
  Serial.printf("Conneceted to TEST");

  /* Set BUZZER as OUTPUT */
  pinMode(BUZZER, OUTPUT);
  /* Initialize SPI bus */
  SPI.begin();
}

void loop()
{
  /* Initialize MFRC522 Module */
```

```
  mfrc522.PCD_Init();
  /* Look for new cards */
  /* Reset the loop if no new card is present on RC522 Reader */
  if ( ! mfrc522.PICC_IsNewCardPresent())
  {
    return;
  }
  /* Select one of the cards */
  if ( ! mfrc522.PICC_ReadCardSerial())
  {
    return;
  }
  /* Read data from the same block */
  Serial.println();
  Serial.println(F("Reading last data from RFID..."));
  ReadDataFromBlock(blockNum, readBlockData);
  /* If you want to print the full memory dump, uncomment the next
line */
  //mfrc522.PICC_DumpToSerial(&(mfrc522.uid));

  /* Print the data read from block */
  Serial.println();
  Serial.print(F("Last data in RFID:"));
  Serial.print(blockNum);
  Serial.print(F(" --> "));
  for (int j=0 ; j<16 ; j++)
  {
```

```
    Serial.write(readBlockData[j]);
  }
  Serial.println();
  digitalWrite(BUZZER, HIGH);
  delay(200);
  digitalWrite(BUZZER, LOW);
  delay(200);
  digitalWrite(BUZZER, HIGH);
  delay(200);
  digitalWrite(BUZZER, LOW);

  // wait for WiFi connection
  if ((WiFiMulti.run() == WL_CONNECTED))
  {
    std::unique_ptr<BearSSL::WiFiClientSecure>client(new
BearSSL::WiFiClientSecure);

    client->setFingerprint(fingerprint);
    // Or, if you happy to ignore the SSL certificate, then use the
following line instead:
    // client->setInsecure();

    data2 = data1 + String((char*)readBlockData);
    data2.trim();
    Serial.println(data2);

    HTTPClient https;
```

```
    Serial.print(F("[HTTPS] begin...\n"));
    if (https.begin(*client, (String)data2))
    {
      // HTTP
      Serial.print(F("[HTTPS] GET...\n"));
      // start connection and send HTTP header
      int httpCode = https.GET();

      // httpCode will be negative on error
      if (httpCode > 0)
      {
        // HTTP header has been send and Server response header has
been handled
        Serial.printf("[HTTPS] GET... code: %d\n", httpCode);
        // file found at server
      }
      else
      {
        Serial.printf("[HTTPS] GET... failed, error: %s\n",
https.errorToString(httpCode).c_str());
      }
      https.end();
      delay(1000);
    }
    else
    {
      Serial.printf("[HTTPS} Unable to connect\n");
```

```
    }
  }
}

void ReadDataFromBlock(int blockNum, byte readBlockData[])
{
  /* Prepare the ksy for authentication */
  /* All keys are set to FFFFFFFFFFFFh at chip delivery from the
factory */
  for (byte i = 0; i < 6; i++)
  {
    key.keyByte[i] = 0xFF;
  }
  /* Authenticating the desired data block for Read access using Key A
*/
  status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A,
blockNum, &key, &(mfrc522.uid));

  if (status != MFRC522::STATUS_OK)
  {
     Serial.print("Authentication failed for Read: ");
     Serial.println(mfrc522.GetStatusCodeName(status));
     return;
  }
  else
  {
    Serial.println("Authentication success");
```

```
  }

  /* Reading data from the Block */
  status = mfrc522.MIFARE_Read(blockNum, readBlockData, &bufferLen);
  if (status != MFRC522::STATUS_OK)
  {
    Serial.print("Reading failed: ");
    Serial.println(mfrc522.GetStatusCodeName(status));
    return;
  }
  else
  {
    Serial.println("Block was read successfully");
  }
}
```

JavaScript function which receives the data and adds data to google sheet.

```
function doGet(e) {
  Logger.log( JSON.stringify(e) );
  var result = 'Ok';
  if (e.parameter == 'undefined') {
    result = 'No Parameters';
```

```
  }
  else {
    var sheet_id = '';  // Spreadsheet ID
    var sheet = SpreadsheetApp.openById(sheet_id).getActiveSheet();
    var newRow = sheet.getLastRow() + 1;
    var rowData = [];

    var Curr_Date = new Date();
    rowData[0] = Curr_Date; // Date in column A

    var Curr_Time = Utilities.formatDate(Curr_Date, "Asia/Kolkata",
'HH:mm:ss');
    rowData[1] = Curr_Time; // Time in column B

    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 'name':
          rowData[2] = value; // Employee Name in column C
          result = 'Employee Name Written on column C';
          break;
        default:
          result = "unsupported parameter";
      }
    }
```

```
    Logger.log(JSON.stringify(rowData));
    var newRange = sheet.getRange(newRow, 1, 1, rowData.length);
    newRange.setValues([rowData]);
  }
  return ContentService.createTextOutput(result);
}
function stripQuotes( value ) {
  return value.replace(/^["']|['"]$/g, "");
}
```

Thankyou!