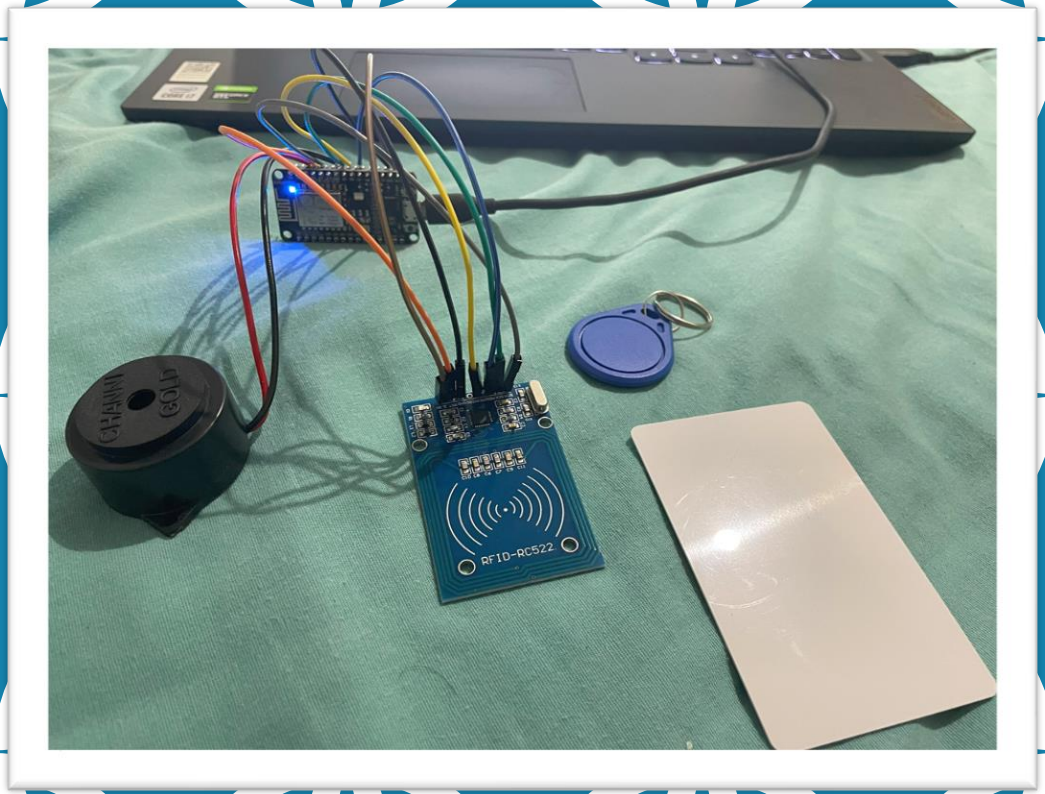
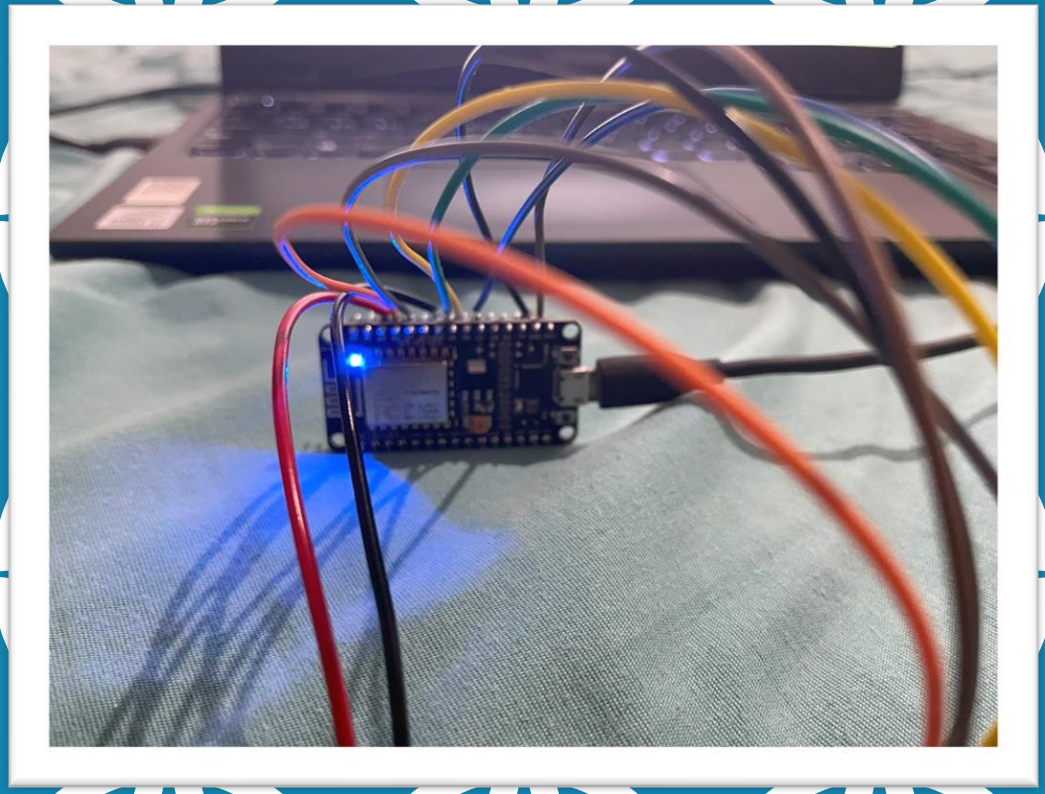




RFID CARD TAG PROJECT: IOT BASED RFID OBJECT TRACKER SYSTEM

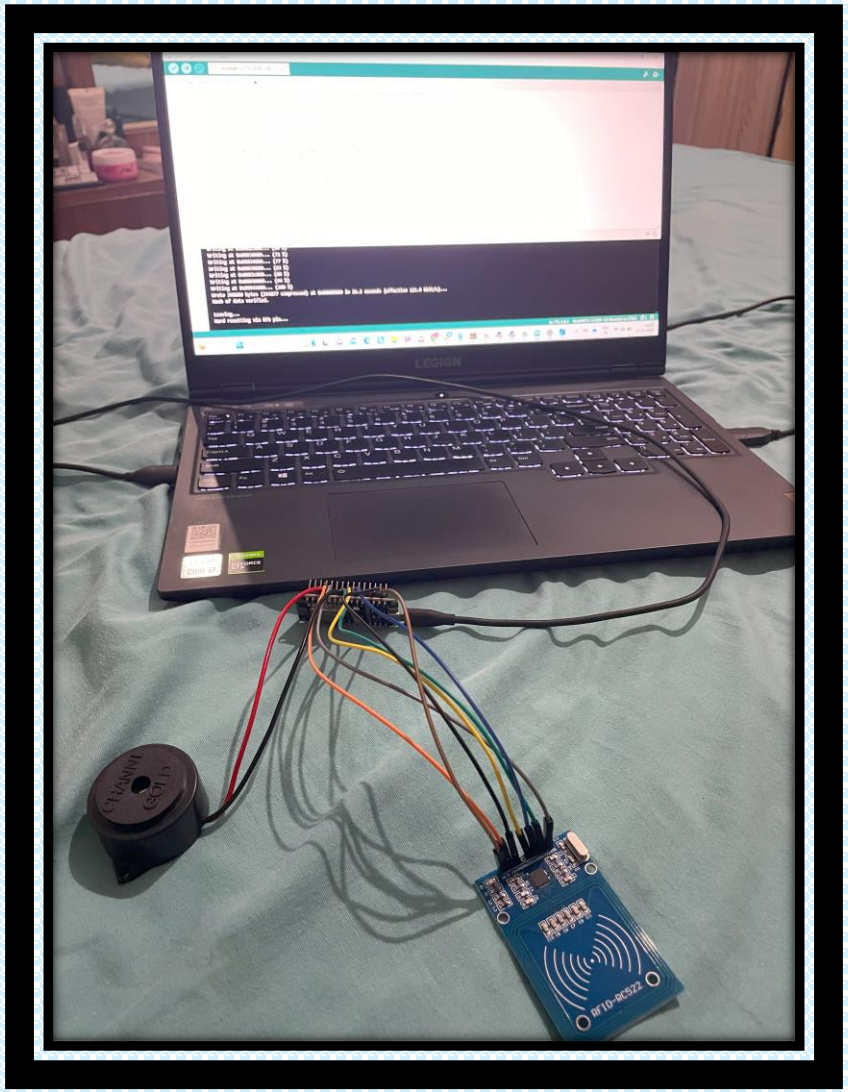
By : Amiya Jha
Rijul Jana
Mentored by : Aditya Raj
Souhardya Chaterjee

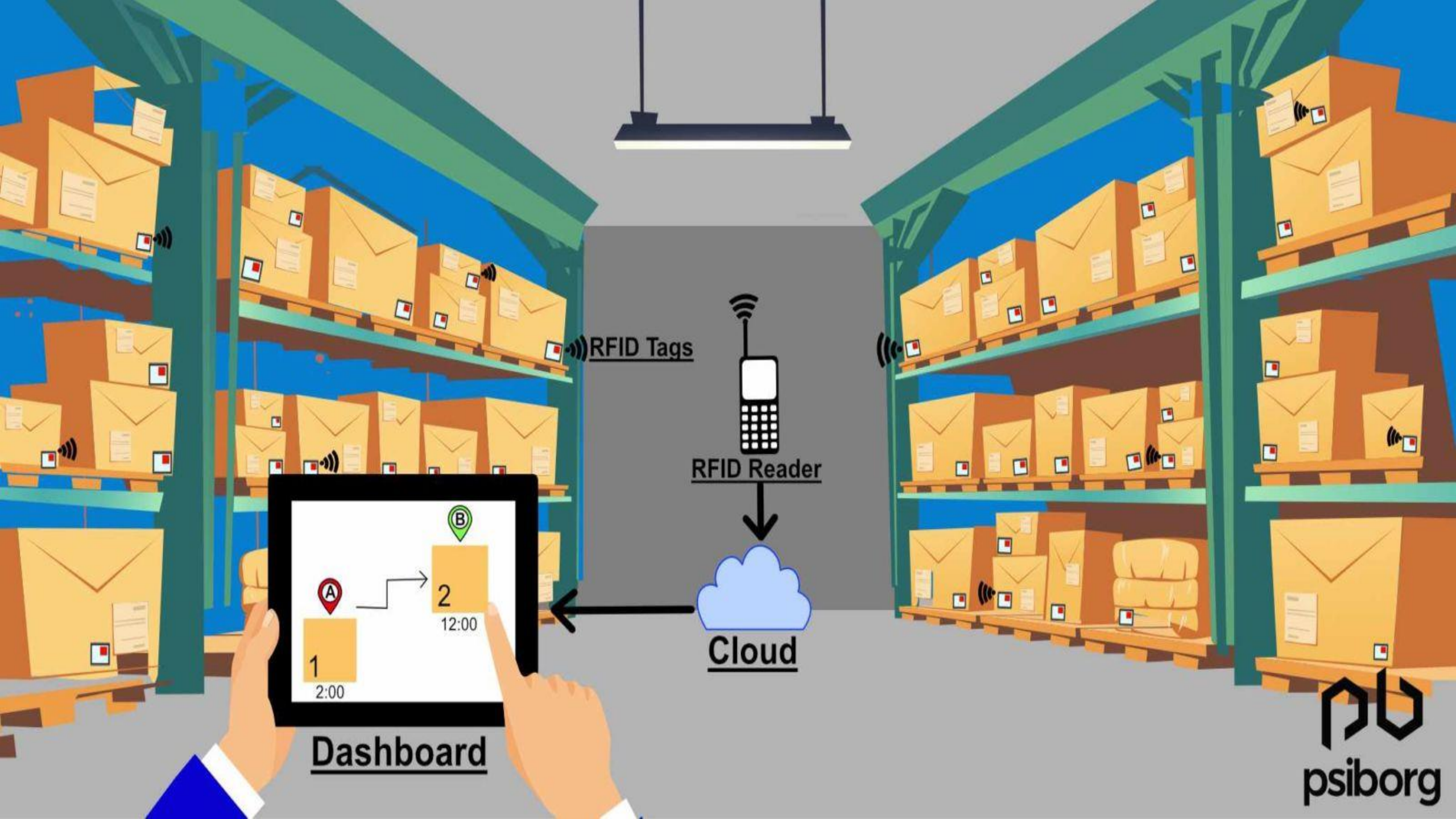
OVERVIEW

- Introduction
- What is RFID ?
- Hardware Components
- Circuit Diagram
- Working
- Code
- Output Video
- Benefits and Applications
- References

INTRODUCTION

The IoT-based RFID object tracker system is designed to track and monitor objects using Radio Frequency Identification (RFID) technology. By attaching RFID tags to objects, data can be captured and transmitted through RFID readers to an IoT network. This network connects to a central server or cloud platform, enabling real-time monitoring, accurate location tracking, and valuable insights into object movement and status. The system utilizes NodeMCU, an open-source IoT platform, to connect RFID readers with Google Sheets, a cloud-based spreadsheet application.

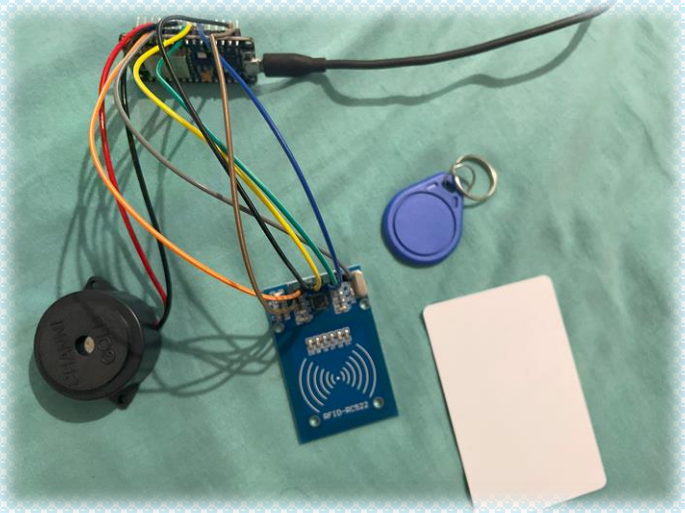



RFID Tags
RFID Reader
Cloud
A
B
1
2:00
2
12:00
Dashboard
psiborg

WHAT IS RFID ?

RFID, or Radio Frequency Identification, is a technology that utilizes radio waves to identify and track objects or individuals. It involves small electronic devices called RFID tags, which store information and communicate with RFID readers via antennas. The readers emit radio waves to power the tags and retrieve the stored data, which is then processed by a backend system. RFID finds applications in inventory management, supply chain tracking, access control, payment systems, transportation, logistics, and asset tracking, among others.

HARDWARE COMPONENTS

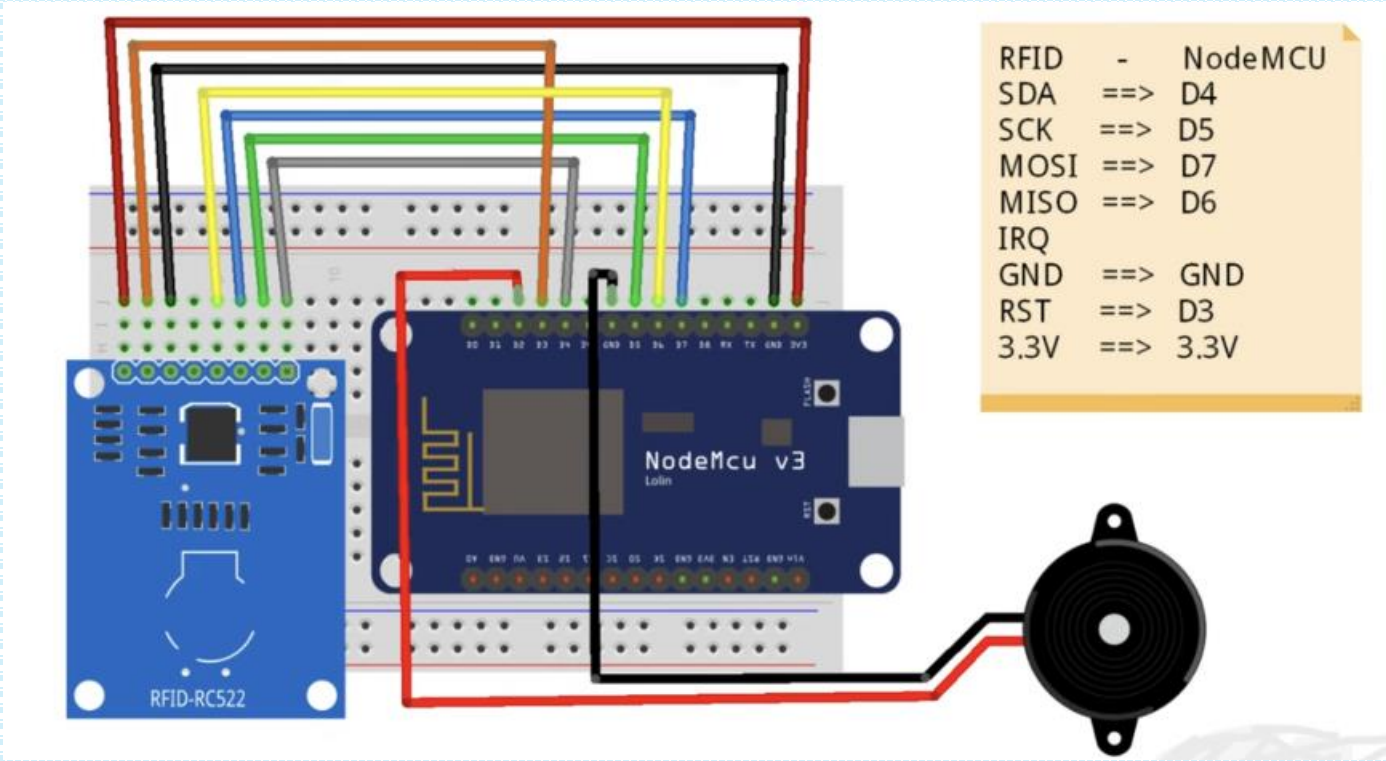


- NodeMCU ESP8266
- RFID-RC522 Sensor (Reader)
- RFID Card Tag
- 5 V Buzzer
- Jumper Wires
- Breadboard

SOFTWARE USED : Arduino IDE

ARDUINO

CIRCUIT DIAGRAM



WORKING

The workflow of the IoT-based RFID object tracker system can be summarized as follows:

- RFID Tag Reading: The RFID readers are placed in strategic locations to detect the RFID tags attached to objects. When an object with an RFID tag comes in proximity to a reader, the reader captures the UID of the tag.
- NodeMCU Data Processing: The NodeMCU board receives the UID data from the RFID readers and processes it. The processing involves converting the data into a suitable format for transmission to Google Sheets.
- Wi-Fi Connectivity: The NodeMCU board establishes a connection with the configured Wi-Fi network. It utilizes the ESP8266 Wi-Fi module to connect to the network and ensure reliable data transmission.
- Data Transmission to Google Sheets: Once connected to the Wi-Fi network, the NodeMCU board sends the processed RFID tag data to Google Sheets using the Google Sheets API. This API allows for seamless integration between the NodeMCU board and the Google Sheets application.
- Data Storage and Management: Google Sheets receives the RFID tag data and stores it in the designated spreadsheet. The spreadsheet have been customized with relevant columns to capture information such as date, time, object descriptions and location details.

GOOGLE APPSCRIPT CODE

```
function doGet(e) {
  Logger.log( JSON.stringify(e) );
  var result = 'Ok';
  if (e.parameter == 'undefined') {
    result = 'No Parameters';
  }
  else {
    var sheet_id = '1A4D4pxzIDoms_OTQDwWXduZnHQT7B0mv_FosgOb4D44';  // 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; // Object Name in column C
          result = 'Object 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, "");
}
```

NODEMCU CODE ON ARDUINO IDE

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

// Fingerprint for demo URL, expires on September 11, 2023, needs to be updated well before this date
const uint8_t fingerprint[20] = {0x56, 0xED, 0xC7, 0xDA, 0xBF, 0x51, 0x12, 0xC2, 0x79, 0x43, 0xC6, 0x01, 0xAB, 0xF7, 0x88, 0x98, 0x0F, 0x97, 0xB2, 0xB8};
// 56 ED C7 DA BF 51 12 C2 79 43 C6 01 AB F7 88 98 0F 97 B2 B8

#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 = 20;
byte readBlockData[20];

String data2;
const String data1 = "https://script.google.com/macros/s/AKfycbxwFt2CFAerv2hQxS-_u2CBjW0wuhAqUEB-dnqZMLkfOsi7VGS2fq68cAk1vKls4xtE/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 */
  WiFiMulti.addAP("JioFiber-5Gh9w", "1234");

  /* 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());
    }
```

```
135       https.end();
136       delay(1000);
137     }
138     else
139     {
140       Serial.printf("[HTTPS} Unable to connect\n");
141     }
142   }
143 }
144
145 void ReadDataFromBlock(int blockNum, byte readBlockData[])
146 {
147   /* Prepare the ksy for authentication */
148   /* All keys are set to FFFFFFFFFFFFh at chip delivery from the factory */
149   for (byte i = 0; i < 6; i++)
150   {
151     key.keyByte[i] = 0xFF;
152   }
153   /* Authenticating the desired data block for Read access using Key A */
154   status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, blockNum, &key, &(mfrc522.uid));
155
156   if (status != MFRC522::STATUS_OK)
157   {
158     Serial.print("Authentication failed for Read: ");
159     Serial.println(mfrc522.GetStatusCodeName(status));
160     return;
161   }
162   else
163   {
164     Serial.println("Authentication success");
165   }
166
167   /* Reading data from the Block */
168   status = mfrc522.MIFARE_Read(blockNum, readBlockData, &bufferLen);
```

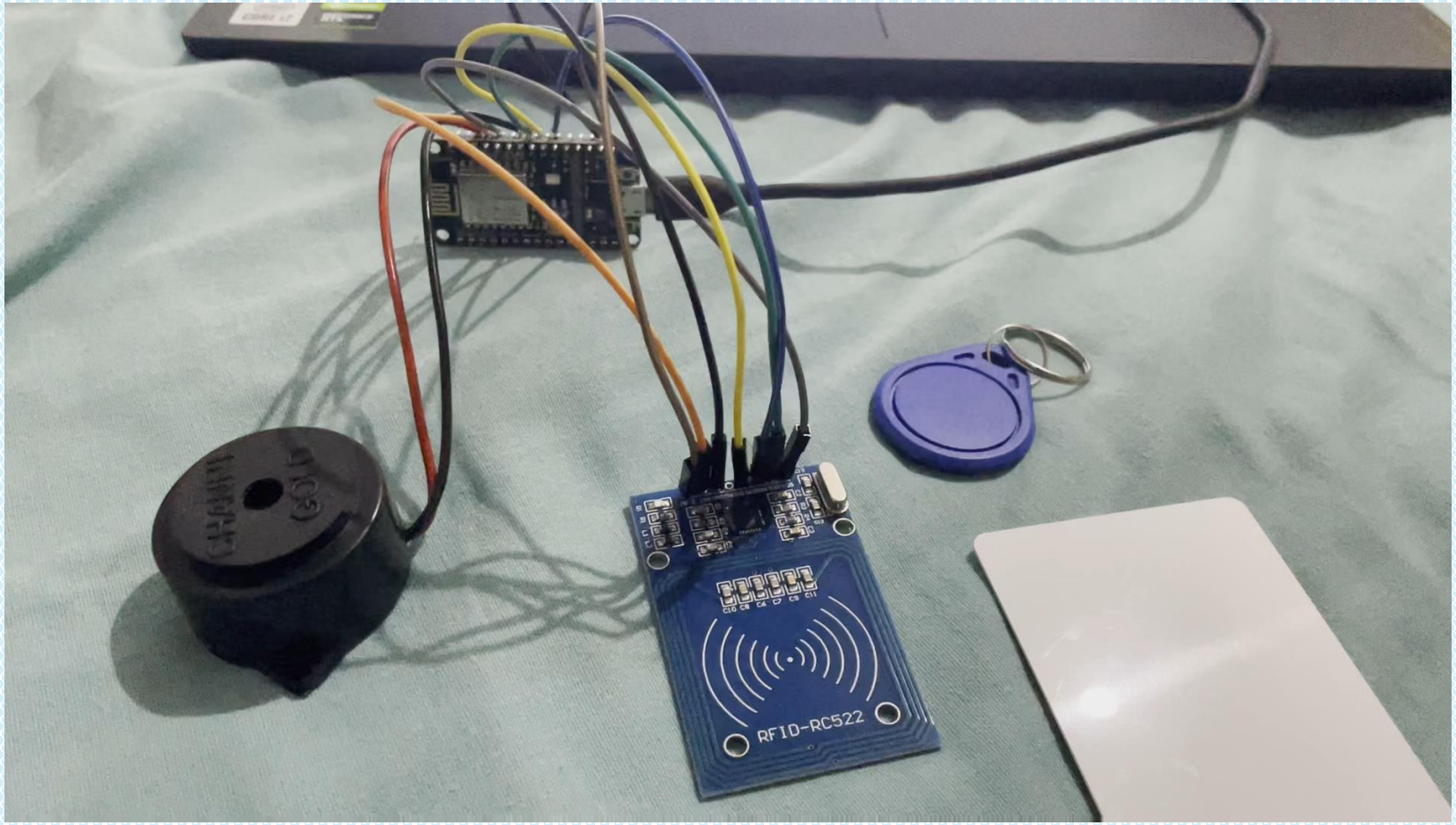
```
  /* Prepare the ksy for authentication */
  /* All keys are set to FFFFFFFFFFFFh at chip delivery from the factory */
  for (byte i = 0; i < 6; i++)
  {
    key.keyByte[i] = 0xFF;
  }
  /* Authenticating the desired data block for Read access using Key A */
  status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A,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

```

OUTPUT VIDEO



BENEFITS AND APPLICATIONS

a. Inventory Management: Real-time tracking of objects helps streamline inventory management processes, reducing errors and improving efficiency.

b. Supply Chain Optimization: The system enables accurate tracking of objects throughout the supply chain, enhancing visibility and reducing the risk of loss or theft.

c. Asset Tracking: Valuable assets can be monitored and tracked, ensuring their proper utilization and minimizing the chances of misplacement.

d. Research and Data Collection: The system can be used in research studies or data collection projects where tracking and analysis of objects are required.

REFERENCES

https://ieeexplore.ieee.org/document/9616967

https://www.researchgate.net/publication/315808980_RFID-based_Object_Detection_System_in_IoT

https://www.irjet.net/archives/V6/i2/IRJET-V6I2180.pdf