# **CHAPTER - 1**

# **INTRODUCTION**

In recent years, RFID technology has been widely used in various sectors, such as in-education, transportation, agriculture, animal husbandry, store sales and other sectors. RFID utilization in education is student attendance monitoring system, by using Internet of Things (IoT) and Cloud technology, it will produce a real time attendance monitoring system that can be accessed by various parties, such as lecturer, campus administration and parents. With this monitoring system if there are students who are not present can be immediately discovered and can be taken immediate action and the learning process can run smoothly.

Key words: Attendance monitoring system, internet of things, RFID

In the rapidly evolving landscape of technology, the integration of Internet of Things (IoT) has revolutionized numerous domains, including attendance management systems. This project presents an IoT-based smart attendance system that leverages RFID (Radio Frequency Identification) technology and Google Sheets for efficient, real-time attendance tracking. The system is designed to streamline the process of recording attendance, eliminating manual errors and ensuring data accuracy. By using RFID tags, individuals can easily check in, and their attendance data is automatically read by an Arduino device. This data is then transmitted to a Google Script, which processes and updates a Google Sheets document instantly. This seamless integration offers a reliable and automated solution for managing attendance, making it particularly beneficial for educational institutions, corporate environments, and other organizations requiring precise attendance monitoring.

## **1.1 OBJECTIVES**

* **Automate Attendance Tracking:** Develop an efficient system to automate the process of attendance tracking in educational institutions or workplaces, minimizing manual intervention and errors.
* **Real-Time Data Recording:** Enable real-time recording and updating of attendance data to a central repository (Google Sheets), ensuring instant access to current attendance records.
* **Integrate RFID Technology:** Utilize RFID technology for the identification and verification of individuals, ensuring quick and accurate attendance marking.
* **Improve Data Accessibility:** Ensure that attendance data is easily accessible to authorized personnel via Google Sheets, providing a user-friendly and familiar interface for data management and analysis.
* **Enhance Security:** Implement security measures to protect attendance data and ensure that only authorized individuals can access and modify the information.
* **Reduce Administrative Burden:** Minimize the workload of administrative staff by automating attendance tracking, allowing them to focus on more critical tasks.

## **1.2 SCOPE**

The project involves developing an IoT-based attendance system using NodeMCU (ESP8266/ESP32) to automate attendance tracking. The system will utilize RFID readers to identify individuals through their RFID tags/cards and record their attendance in real-time on Google Sheets. The NodeMCU will be programmed using the Arduino IDE to read RFID data and send it to Google Sheets via the internet. Google Sheets will serve as the central repository for managing and analyzing attendance data, with fields for date, time, individual ID, name, and attendance status. The system will generate daily, weekly, and monthly attendance reports, and provide analytical insights such as trend analysis and absenteeism reports. Security measures will include data encryption and access control to ensure data privacy.

# **CHAPTER - 2**

# **LITERATURE REVIEW**

This paper introduces a new approach to utilizing RFID (Radio Frequency Identification) to track student attendance that is based on the Internet of Things (IoT). Educational institutions are concerned about student absences. The general academic achievement of a pupil may suffer from truancy. It takes a lot of time and is ineffective to take attendance the old-fashioned way, by calling names or having people sign their names on paper. One of the answers to the issue is an RFID-based attendance system that uses an IoT system. The two most well-liked technological research trends—IoT and RFID—are included in the suggested study [4]. If we look at the state of our educational system right now, we can see that although there are many technologies available, we are still using the conventional system. When it comes to the university and school attendance systems, professors handled that work manually. The database was manually updated by lecturers using the attendance data. When it comes to technology, there are several solutions available that can be used to lighten the load of lectures. One illustration of such is the use of RFID. If RFID and IoT (Internet of Things) are used together, they can be done automatically without the need for lectures. For improved speed, we intend to use the Cloud as storage in this case. We can access it at any time and from any location via IoT and the cloud, giving us greater proficiency and flexibility [5]. Students are required to be present. The lecturer or teacher cannot evaluate a student’s participation without the attendance procedure. However, the current procedure still involves physically taking attendance on paper. The use of excessive paper is the first issue, and it is challenging for the administration to summarize student attendance data that is the second issue. This is due to the administration having to review a large number of attendance records.

# **CHAPTER – 3**

# **3. HARDWARE AND SOFTWARE REQUIREMENTS**

## **3.1 Hardware Requirements**

* **NODEMCU (ESP8266)**

The primary microcontroller for managing RFID reader inputs and handling network communications.

* **RFID Reader (e.g., RC522)**

Device to read RFID tags/cards and identify individuals.

* **RFID Tags/Cards**

Unique identification tags/cards assigned to each individual.

* **Power Supply**

Appropriate power supply for NodeMCU and RFID reader.

* **Connecting Wires and Breadboard**

For connecting the RFID reader to the NodeMCU.

* **USB Cable**

To program and power the NodeMCU.

* **LCD Display with I2C Interface**

For displaying attendance status and other messages.

* **Buzzer**

To provide audio feedback for successful or unsuccessful attendance marking.

* **LED Bulb**

For visual feedback (e.g., green for success, red for failure

## **3.2 Software Requirements**

* **Arduino IDE**

For programming the NodeMCU and writing the firmware code to read RFID data and send it to Google Sheets.

* **Google Sheets**

To store, manage, and analyze attendance data.

* **Google Apps Script**

To create an API endpoint for updating Google Sheets from the NodeMCU.

* **Wi-Fi Network**

To enable NodeMCU to connect to the internet and communicate with Google Sheets.

* **Web Browser**

For accessing Google Sheets and setting up Google Apps Script.

* **Libraries and Dependencies**

ESP8266/ESP32 Wi-Fi Libraries: To manage Wi-Fi connectivity.

MFRC522 Library: For interfacing with the RFID reader.

HTTP Client Library: To send HTTP requests from NodeMCU to Google Sheets API.

LiquidCrystal\_I2C Library: For interfacing with the LCD display via I2C.

# **CHAPTER – 4**

# **SYSTEM REQUIREMENT SPECIFICATIONS**

This document outlines the system requirements for an IoT-based Smart Attendance System using NodeMCU, RFID, and Google Sheets. The system aims to automate attendance tracking, providing real-time updates, and enhancing data accessibility and management.

## **4.1 FUNCTIONAL REQUIREMENTS**

* **RFID Attendance Tracking**

RFID Reading: The system must read RFID tags/cards when presented to the RFID reader.

Data Capture: Capture and store the unique ID from each RFID tag/card along with the current timestamp.

Attendance Marking: Automatically mark attendance for the identified individual and update the attendance record.

* **Data Management and Storage**

Real-Time Data Update: Attendance data must be updated in real-time in Google Sheets.

Data Fields: Store necessary data fields such as date, time, individual ID, name, and attendance status (present/absent).

* **User Feedback and Notification**

LCD Display Integration: Display relevant messages on the LCD screen, such as “Attendance Recorded”, “Invalid Card”, or “Already Marked”.

Audio Feedback: Use a buzzer to provide audio feedback, with distinct sounds for successful and unsuccessful scans.

Visual Feedback: Use an LED bulb to indicate the status of the attendance marking process (e.g., green for success, red for failure).

* **Data Access and Reporting**

Daily Reports: Generate and display daily attendance reports.

Periodic Reports: Generate weekly and monthly summary reports.

Custom Reports: Allow the creation of custom reports based on specific criteria (e.g., department-wise, date range).

* **Administrative Functions**

User Management: Allow administrators to add, remove, and update user information.

Attendance Record Management: Enable administrators to view, edit, and manage attendance records.

## **4.2 MODULES DESCRIPTION**

* **NODEMCU 8266**

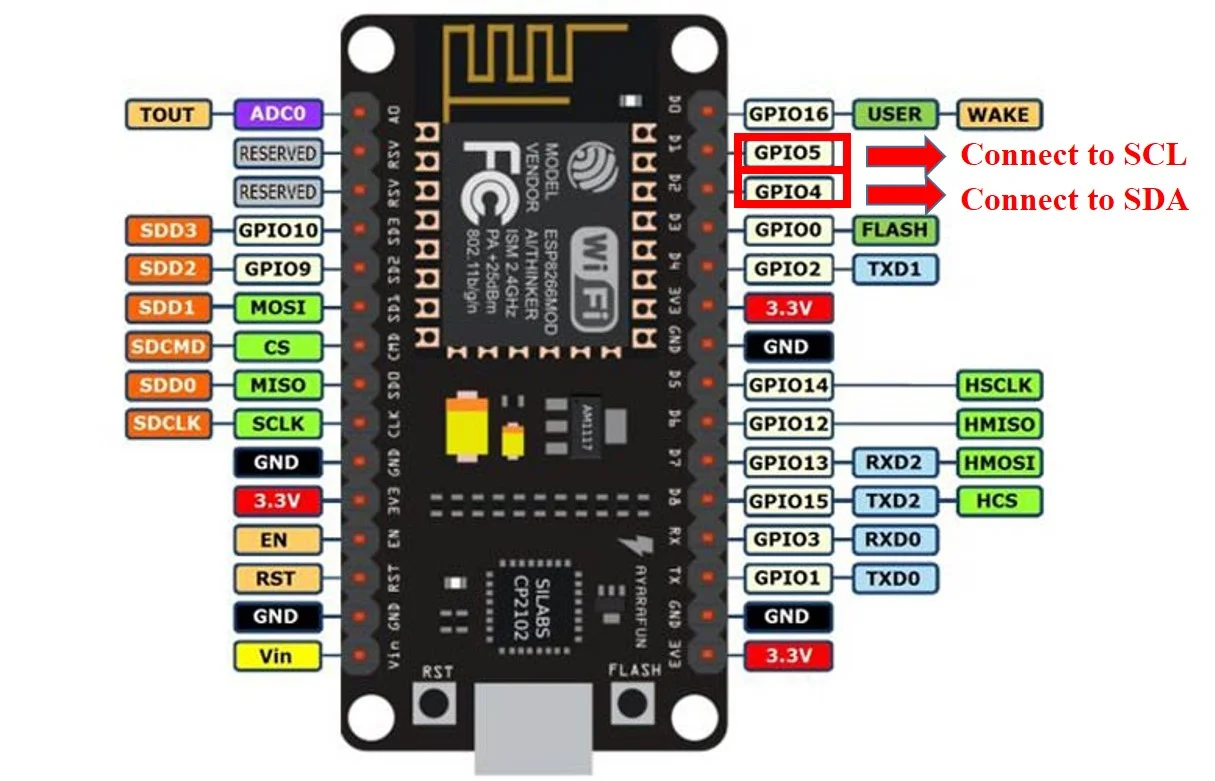
****

Figure 1:**NODEMCU 8266**

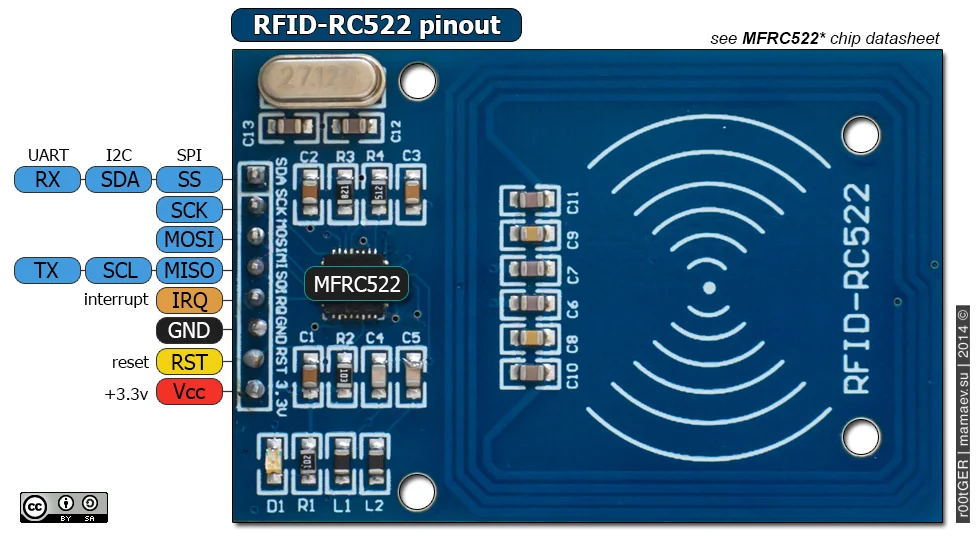
The NodeMCU ESP8266 is a versatile and cost-effective Wi-Fi microcontroller module widely used in IoT (Internet of Things) applications. At its core is the ESP8266EX SoC (System on Chip), featuring a 32-bit Tensilica Xtensa LX106 RISC microprocessor that operates at either 80 MHz or 160 MHz, with 64 KB of instruction RAM and 96 KB of data RAM. The module can also support up to 16 MB of external flash memory via an SPI interface.

One of the standout features of the NodeMCU is its comprehensive Wi-Fi capabilities. It adheres to the 802.11 b/g/n standard, ensuring compatibility with most wireless networks. Additionally, it integrates a TCP/IP protocol stack, which simplifies the process of connecting to Wi-Fi networks. The ESP8266 can function as an access point, a station, or both simultaneously, providing flexibility in various networking scenarios.

The NodeMCU includes 17 GPIO (General-Purpose Input/Output) pins that can be configured for digital input/output, analog input (via a 10-bit ADC on pin A0), PWM, I2C, SPI, and UART functionalities. This makes it highly adaptable for interfacing with a wide range of sensors and peripherals. Communication interfaces such as UART (with two interfaces: UART0 for RX/TX and UART1 for TX only), SPI (Serial Peripheral Interface), and I2C (Inter-Integrated Circuit) expand its connectivity options, allowing for efficient data transfer between devices.

Power management is another crucial aspect, with the NodeMCU being capable of being powered through a standard 5V USB connection. This ease of power supply, coupled with its rich feature set, makes the NodeMCU ESP8266 an excellent choice for developers and hobbyists looking to build connected devices and smart applications.

**RC522 RFID Reader Module**



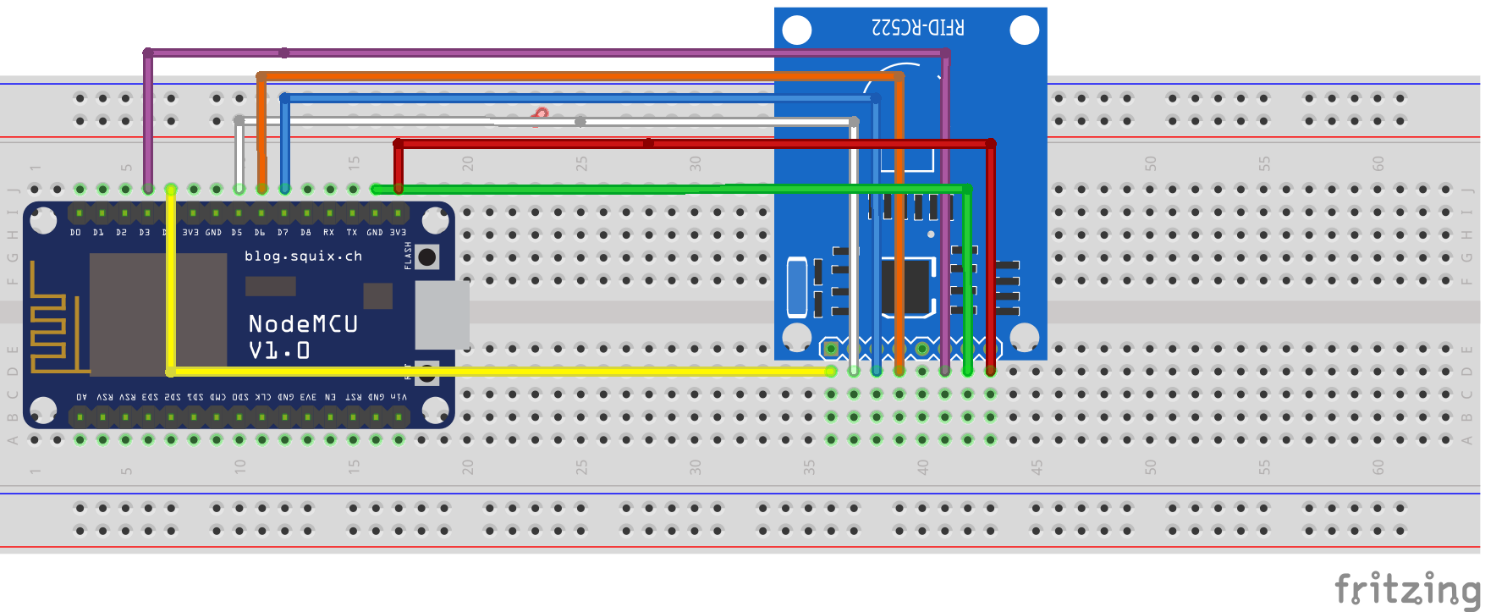
**Figure 2:RC522**

The RC522 RFID Reader Module is a popular, low-cost RFID (Radio Frequency Identification) reader and writer, widely used for various applications such as access control, contactless payment systems, and inventory tracking. This module operates at 13.56 MHz frequency and supports ISO/IEC 14443A/MIFARE and NTAG standards, making it compatible with a broad range of RFID tags, cards.

At the heart of the RC522 module is the MFRC522 IC, which handles all the communication with RFID tags. The module uses SPI (Serial Peripheral Interface) for communication with microcontrollers, although it also supports I2C and UART interfaces. This flexibility allows it to be easily integrated with various microcontroller platforms such as Arduino, Raspberry Pi, and ESP8266.

The module features an onboard antenna, which simplifies the design and setup process. It typically operates at a voltage range of 2.5V to 3.3V, but with appropriate level shifting, it can be used with 5V systems like many Arduino boards. The reading distance for the RC522 module varies depending on the size of the antenna and the tag but is usually around 5 to 7 cm.

In terms of functionality, the RC522 module can perform read and write operations on RFID cards and tags. It supports various authentication schemes to ensure secure communication, which is crucial for applications involving sensitive data. The module also includes several GPIO pins that can be configured for various auxiliary functions, such as triggering other devices or indicating the status of read operations with LEDs.

****

**Figure 3:NODEMCU CONNECTION RC522**

* **Power Connections**

Connect the VCC pin of the RC522 module to the 3.3V pin on the NodeMCU8266.

Connect the GND pin of the RC522 module to any GND pin on the NodeMCU8266.

* **Data Connections**

Connect the SDA pin of the RC522 module to any GPIO pin on the NodeMCU8266

Connect the SCK pin of the RC522 module to any GPIO pin on the NodeMCU8266.

Connect the MOSI pin of the RC522 module to any GPIO pin on the NodeMCU8266.

Connect the MISO pin of the RC522 module to any GPIO pin on the NodeMCU8266.

Connect the IRQ pin of the RC522 module to any GPIO pin on the NodeMCU8266.

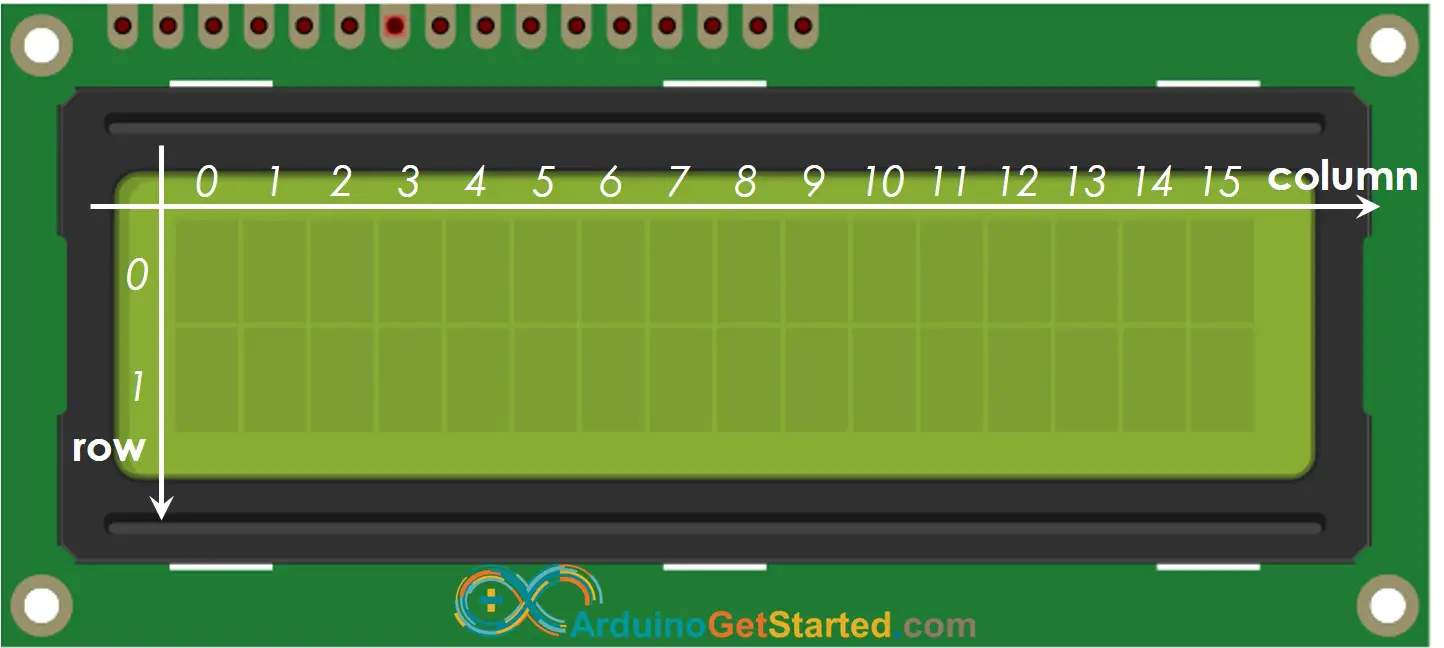
* **Antenna Connection**

Ensure that the antenna is properly connected to the ANT1 pin of the RC522 module.

* **Additional Pins (if required):**

Some modules may have additional pins like RST (Reset) and SDA (Serial Data) that might need connection based on your application. Connect them accordingly.

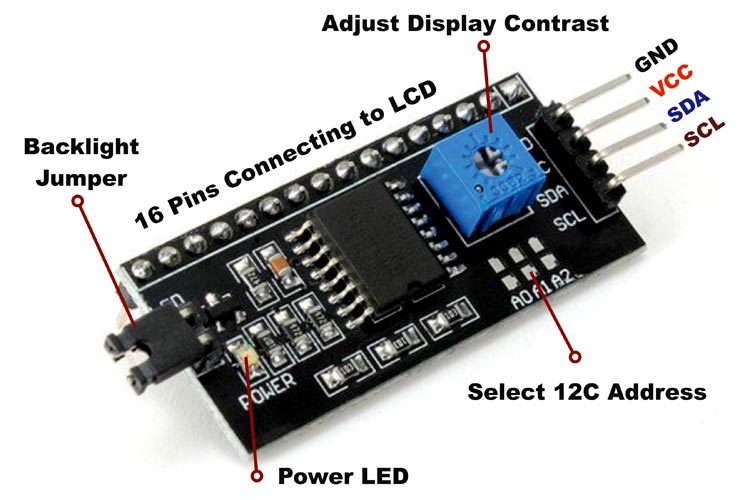
* **LCD I2C MODULE**



**Figure 4:LCD**

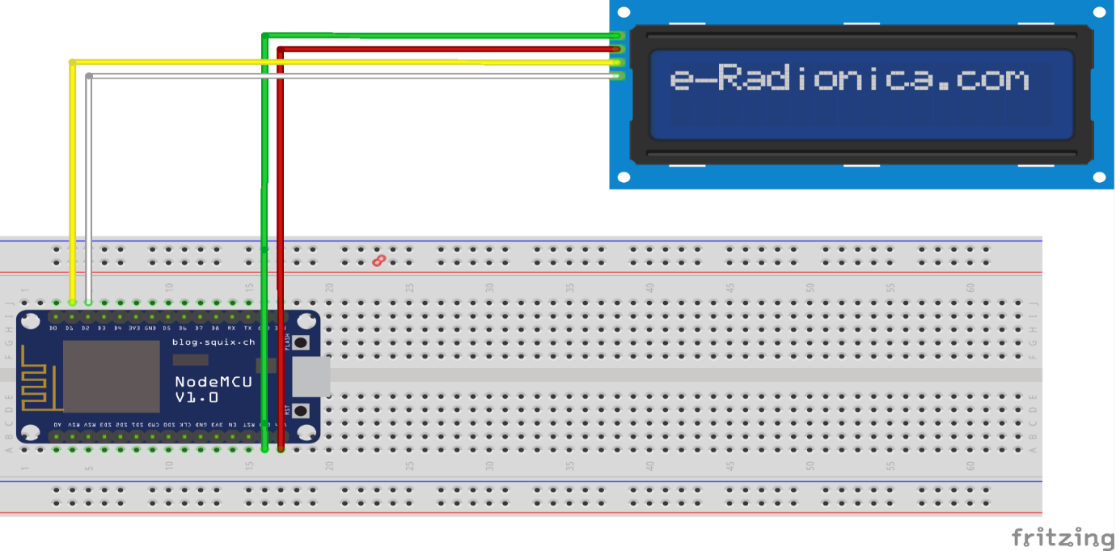
The LCD I2C Module is an efficient and user-friendly interface designed to control LCD displays using the I2C (Inter-Integrated Circuit) protocol. This module, often used with character LCDs like the 16x2 (16 characters per line, 2 lines) or 20x4 (20 characters per line, 4 lines) displays, simplifies the communication between a microcontroller and the LCD.

**I2C Interface:**



**Figure 5:I2C INTERFACE**

The key feature of the LCD I2C module is the I2C communication interface. Unlike traditional parallel LCDs that require multiple GPIO pins for control, the I2C module reduces the connection to just two pins: SDA (Serial Data) and SCL (Serial Clock). This significant reduction in pin usage is especially beneficial for projects with limited GPIO availability, such as those using Arduino, ESP8266.



**Figure 6: NODEMCU CONNECTION LCD I2C INTERFACE**

Connect the I2C LCD to the NodeMCU8266 as follows:

Connect the VCC pin of the I2C LCD to the 5V pin on the NodeMCU8266.

Connect the GND pin of the I2C LCD to any GND pin on the NodeMCU8266.

Connect the SDA (data) pin of the I2C LCD to the D2 pin (GPIO4) on the NodeMCU8266.

Connect the SCL (clock) pin of the I2C LCD to the D1 pin (GPIO5) on the NodeMCU8266.

**BUZZER :**

****

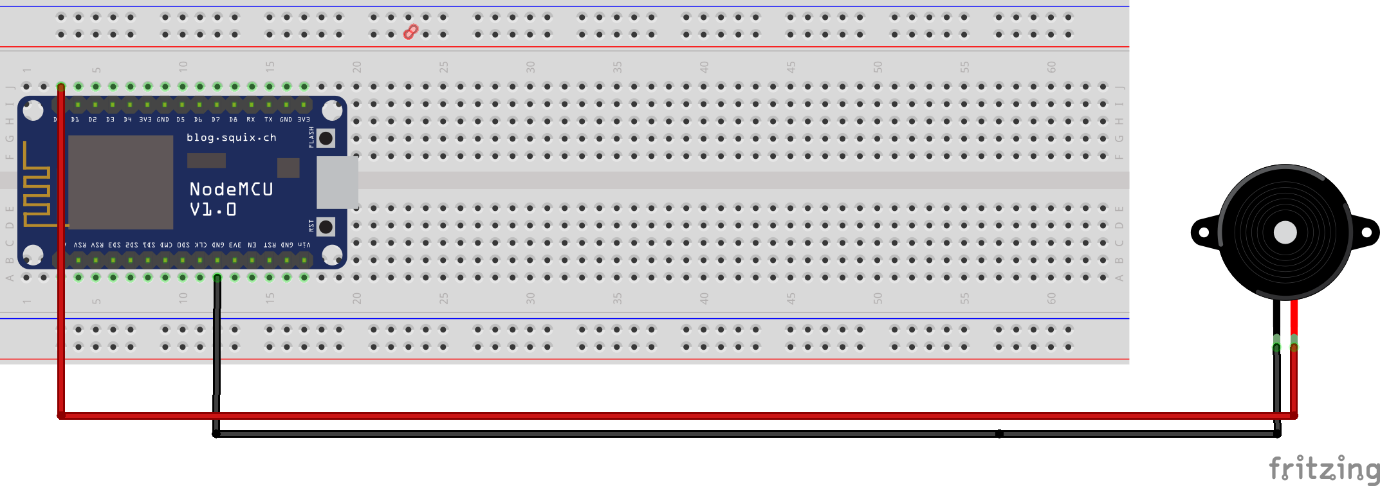
**Figure 7:BUZZER**

The buzzer module is a fundamental electronic component utilized for generating audible alerts or tones in numerous applications. At its core lies a piezoelectric transducer, which converts electrical energy into mechanical vibrations, thereby producing sound. Operating within a voltage range typically spanning from 3V to 12V, these modules emit sound output ranging from continuous tones to intermittent pulses, depending on the application's needs. Available in diverse sizes and forms, from compact units suitable for handheld devices to larger ones for industrial setups, buzzer modules offer flexibility in installation and integration into electronic circuits. Correct polarity connection is crucial, ensuring proper functionality by aligning positive and negative terminals appropriately. Additionally, some buzzer modules boast durability and environmental resistance, making them suitable for outdoor or industrial deployment where conditions might be harsh. From alarm systems and automotive applications to consumer electronics and industrial automation, buzzer modules play a pivotal role in signaling alerts, indicating completion of tasks, or conveying critical information to users. With straightforward control interfaces and reliable performance, buzzer modules remain indispensable across a wide spectrum of electronic devices and systems**.**

Connect the buzzer to the NodeMCU8266 as follows:

For an active buzzer (which generates sound when powered), connect its positive (+) pin to any GPIO pin on the NodeMCU8266 (e.g., D5). Connect its negative (-) pin to a ground (GND) pin on the NodeMCU8266.

For a passive buzzer (which requires an alternating signal to generate sound), connect its positive (+) pin to any GPIO pin on the NodeMCU8266 (e.g., D5). Connect its negative (-) pin to another GPIO pin on the NodeMCU8266 (e.g., D6).

****

**Figure 8: NODEMCU CONNECTION BUZZER**

* **LED**

****

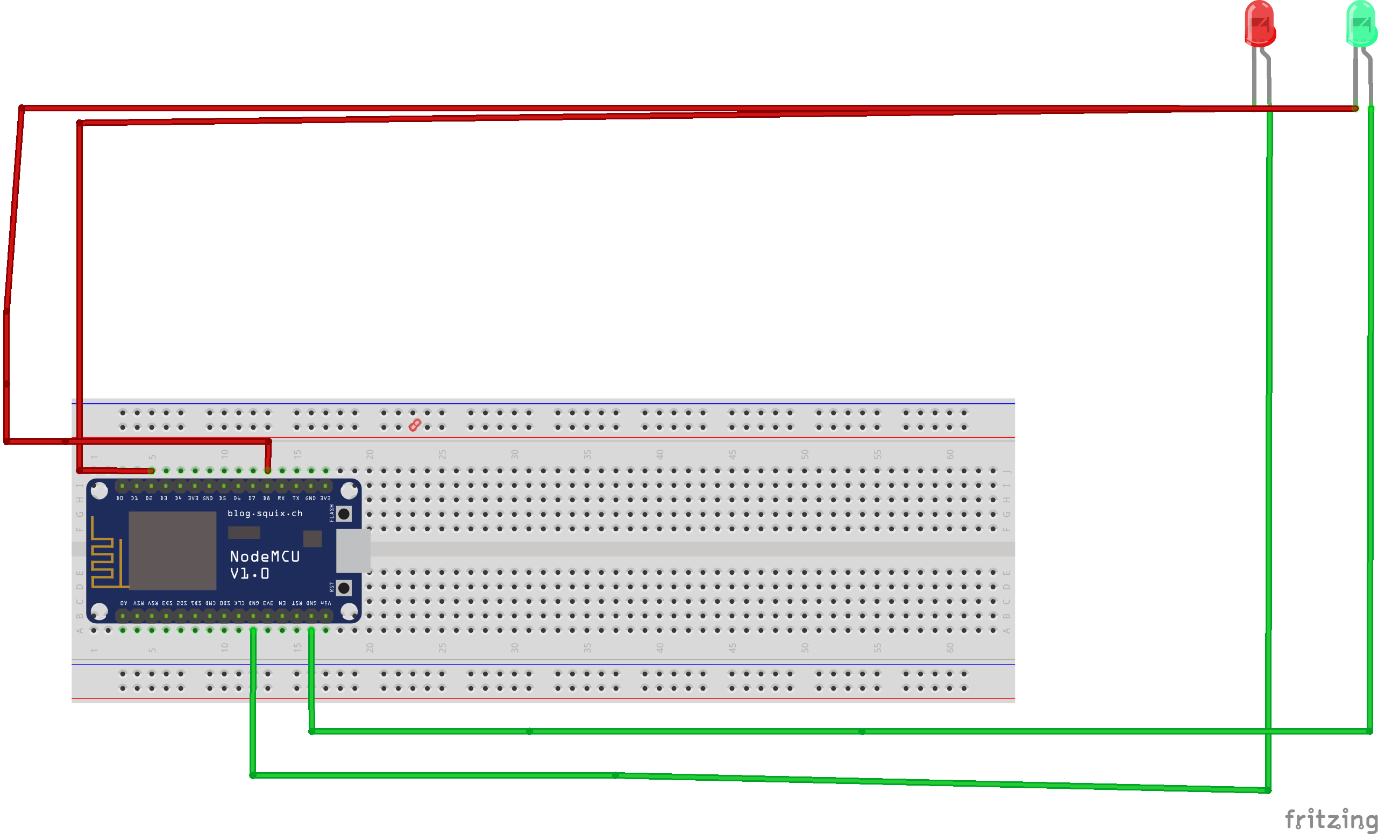
**Figure 9: LED**

In RFID projects, LEDs play a vital role as visual indicators, offering immediate feedback on the status and operations of the system. These LEDs, available in various colors, brightness levels, and viewing angles, are strategically integrated into RFID tags, readers, or associated control circuitry. When an RFID tag is successfully detected by a reader, the LED may illuminate to signal the recognition. Similarly, during data transfer between the tag and the reader, LEDs may flash or change color to indicate communication activity. The choice of LED specifications, including forward voltage, current requirements, and packaging type (such as through-hole or surface-mount), is carefully made to ensure compatibility with the RFID system's power supply and integration needs. Additionally, LEDs with high reliability and long lifespans are preferred to ensure uninterrupted operation in RFID applications where continuous functionality is essential. Whether mounted directly on RFID devices or on external indicator panels, these LEDs provide users with clear, immediate visual cues, enhancing the overall usability and effectiveness of RFID systems in various environments and applications.

Connect the longer leg (anode) of the LED to a GPIO pin on the NodeMCU8266 (e.g., D5).

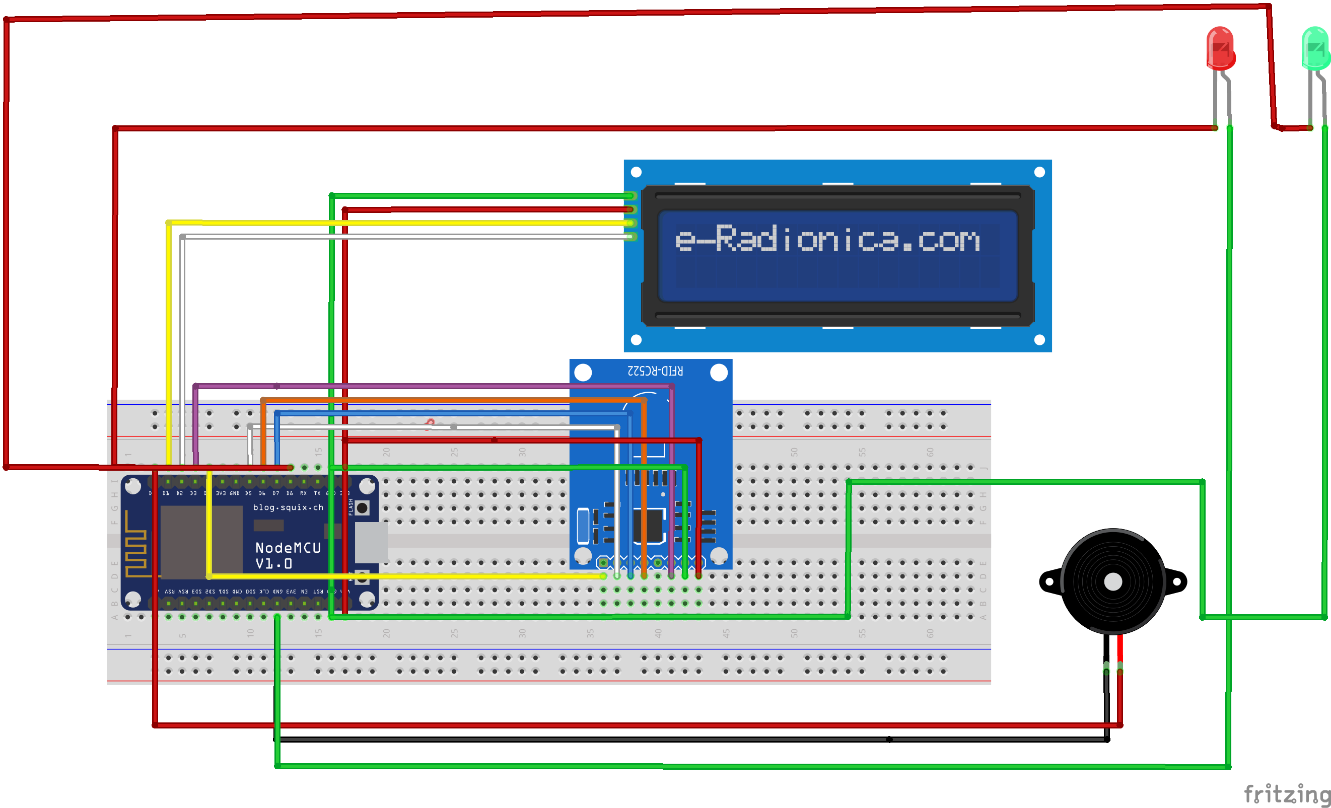
Connect the shorter leg (cathode) of the LED to a resistor.

Connect the other end of the resistor to a ground (GND) pin on the NodeMCU8266.

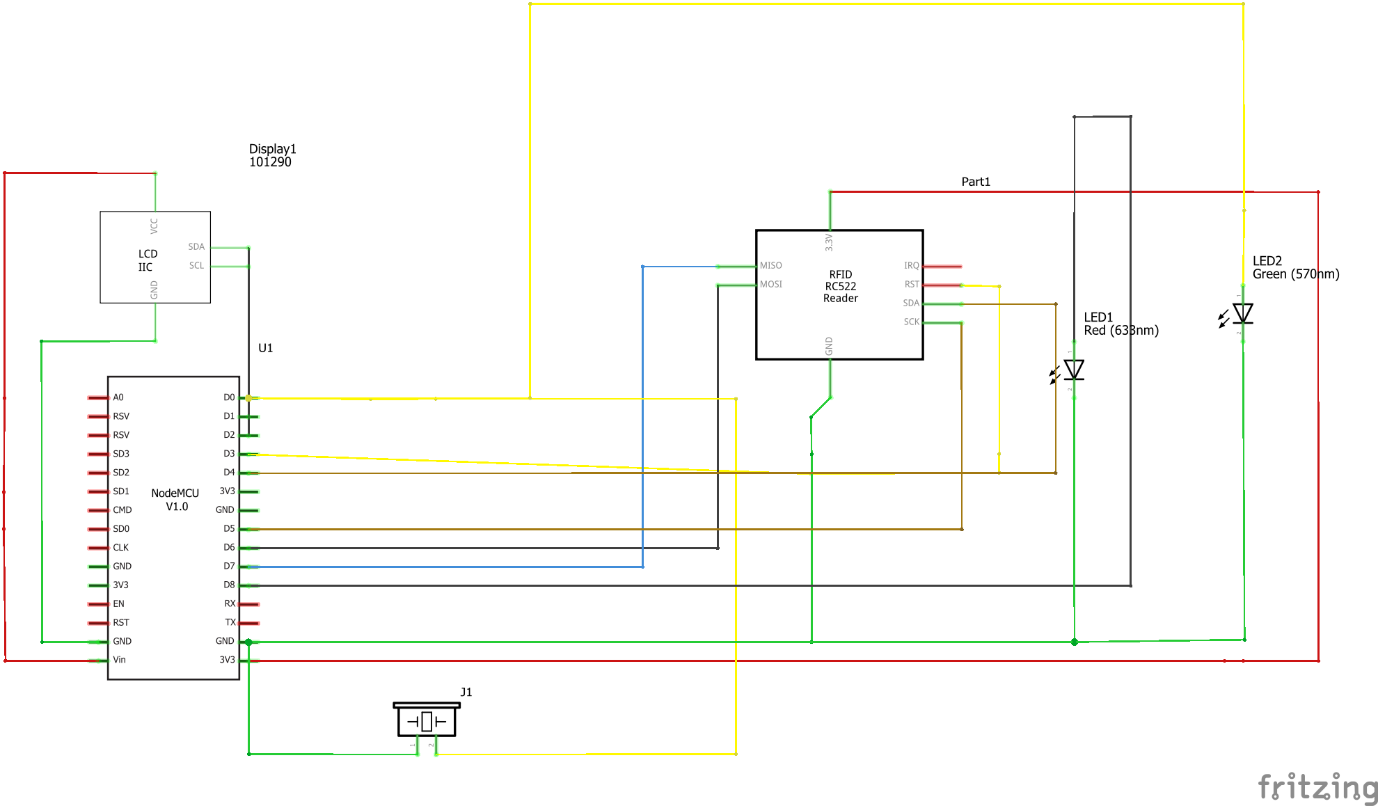


**Figure 10: NODEMCU CONNECTION OF LED**

**CIRCUIT AND SCHEMATIC DIAGRAM:**

****

**Figure 11:CRICUIT DIAGRAM**

****

**Figure 12:DIAGRAM SCHEMATIC**

**RFID Module:**

* Connect the RFID module's VCC pin to the 3.3V pin on the NodeMCU.
* Connect the RFID module's GND pin to any GND pin on the NodeMCU.
* Connect the RFID module's RST pin to any digital GPIO pin on the NodeMCU (e.g., D3).
* Connect the RFID module's SDA pin to any digital GPIO pin on the NodeMCU (e.g., D4).
* Connect the RFID module's SCK pin to the NodeMCU's SCK pin if it uses SPI communication.

**LED:**

* Connect the other end of the resistor to any digital GPIO pin on the NodeMCU (e.g., D0-D8).
* Connect the LED's cathode (shorter leg) to any GND pin on the NodeMCU.
* LCD (16x2 or similar):
* Connect the LCD's VCC pin to the 5V pin on the NodeMCU (some LCDs can work with 3.3V).
* Connect the LCD's GND pin to any GND pin on the NodeMCU.
* Connect the LCD's SDA (I2C data) pin to the NodeMCU's SDA pin (D2).
* Connect the LCD's SCL (I2C clock) pin to the NodeMCU's SCL pin (D1).

**Buzzer:**

* Connect the buzzer's positive terminal (usually red) to any digital GPIO pin on the NodeMCU (e.g., D2).
* Connect the buzzer's negative terminal (usually black) to any GND pin on the NodeMCU.

**LCDI2C:**

Connect the I2C LCD to the NodeMCU8266 as follows:

Connect the VCC pin of the I2C LCD to the 5V pin on the NodeMCU8266.

Connect the GND pin of the I2C LCD to any GND pin on the NodeMCU8266.

Connect the SDA (data) pin of the I2C LCD to the D2 pin (GPIO4) on the NodeMCU8266.

Connect the SCL (clock) pin of the I2C LCD to the D1 pin (GPIO5) on the NodeMCU8266.

## **4.3 NON-FUNCTIONAL REQUIREMENTS**

* **Performance**

Response Time: The system should respond to RFID scans and update Google Sheets within 1-2 seconds.

Feedback Latency: LCD, buzzer, and LED should provide immediate feedback (less than 1 second) after an RFID scan.

* **Reliability**

System Uptime: The system should have a high uptime, with minimal downtime to ensure consistent operation.

Error Handling: Implement robust error handling for network failures, power interruptions, and invalid RFID reads to ensure system stability.

* **Scalability**

Expandability: The system should be designed to easily accommodate more RFID readers and NodeMCU units without significant changes to the infrastructure.

User Capacity: Support an increasing number of users and data entries without degradation in performance.

* **Security**

Data Encryption: All data transmitted between the NodeMCU and Google Sheets should be encrypted using secure protocols (e.g., HTTPS).

Access Control: Implement strict access control measures to ensure only authorized personnel can access or modify attendance data.

Data Integrity: Ensure the integrity of attendance data by implementing validation checks and secure storage practices.

# **CHAPTER – 5**

# **SYSTEM ANALYSIS**

## **5.1 EXISTING SYSTEM**

Many educational institutions and workplaces currently use traditional attendance systems that are either manual or semi-automated. These systems have several limitations, including inefficiency, error-proneness, and lack of real-time data management.

* **Manual Attendance Systems**

**Paper-Based Records:** Attendance is recorded manually on paper registers.

**Human Error:** High likelihood of errors in recording and calculating attendance.

**Time-Consuming:** Requires significant time for recording and consolidating attendance data.

**No Real-Time Data:** Data needs to be manually entered into digital formats for further processing.

* **Semi-Automated Attendance Systems**

**Biometric Systems:** Use fingerprints or facial recognition for attendance but often have high costs and maintenance issues.

**Card Swipe Systems:** Employees or students swipe ID cards to mark attendance, but these systems are prone to hardware malfunctions and require regular maintenance.

**Bar Code/QR Code Systems:** Involves scanning bar codes or QR codes, which can be cumbersome and slow.

### **5.1.1 LIMITATIONS OF EXISTING SYSTEM**

* **Inefficiency**

Time-Consuming Processes: Manual and semi-automated systems often require significant time for recording, consolidating, and processing attendance data. For example, manually marking attendance in paper registers is labor-intensive and slow.

Batch Processing: Semi-automated systems like card swipe or barcode scanners often update attendance data in batches rather than real-time, leading to delays in data availability.

* **Error-Prone**

Human Errors: Manual systems are highly susceptible to human errors such as incorrect entries, omissions, and duplication of records. For instance, a teacher or manager might mistakenly mark a student or employee present when they are absent.

Data Manipulation: Manual and some semi-automated systems can be manipulated, leading to inaccurate attendance records.

* **Lack of Real-Time Data Management**

Delayed Updates: Existing systems often do not provide real-time updates, making it difficult to get an accurate and up-to-date view of attendance. This is particularly problematic in semi-automated systems that process data at the end of the day.

Manual Data Entry: In many cases, data from manual systems need to be manually entered into digital formats, which is both time-consuming and prone to errors.

* **Limited Integration and Reporting**

Poor Integration: Many existing systems do not integrate well with other data management tools or systems, leading to fragmented and siloed data.

Limited Reporting Capabilities: Generating attendance reports often requires manual compilation and analysis, lacking automation and real-time insights. This makes it difficult to track attendance trends or quickly generate reports for decision-making.

* **User Experience Issues**

Inconvenience for Users: Manual attendance taking can be inconvenient for both users and administrators, leading to dissatisfaction and non-compliance. For example, students might find it cumbersome to sign paper registers.

Lack of Immediate Feedback: Existing systems often do not provide immediate feedback to users about their attendance status, which can lead to uncertainty and confusion.

## **5.2 PROPOSED SYSTEM**

The proposed system leverages IoT technology to create an efficient, real-time, and user-friendly attendance tracking solution. By integrating RFID technology with NodeMCU for hardware control and Google Sheets for data management, this system aims to overcome the limitations of existing attendance systems.

### **5.2.1 ADVANTAGES OF PROPOSED SYSTEM**

* **Efficiency**

Automated Attendance Tracking: The system automates the entire attendance tracking process, significantly reducing the time and effort required compared to manual and semi-automated systems.

Real-Time Data Updates: Attendance data is updated in real-time in Google Sheets, providing instant access to up-to-date records.

* **Accuracy and Reliability**

Reduced Human Errors: By automating data capture and processing, the system minimizes human errors that are common in manual attendance tracking.

Consistent Data Entry: The use of RFID technology ensures that each scan is accurately recorded, maintaining consistent and reliable attendance records.

* **User Feedback**

Immediate Feedback: Users receive immediate feedback on their attendance status via the LCD display, buzzer, and LED, reducing uncertainty and confusion.

Clear Communication: The LCD screen displays clear messages, such as “Attendance Recorded”, “Invalid Card”, or “Already Marked”, ensuring users are well-informed.

* **Data Management and Reporting**

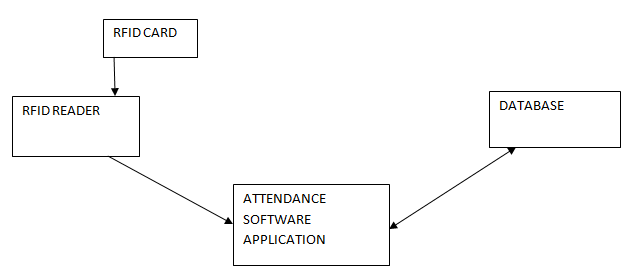
Centralized Data Storage: All attendance data is stored centrally in Google Sheets, making it easy to manage, access, and analyze.

Automated Reporting: The system can generate daily, weekly, and custom reports automatically, saving time and improving data analysis capabilities.

# **CHAPTER – 6**

# **SYSTEM DESIGN**

## **6.1 USE-CASE DIAGRAM**



**Figure 13:USE CASE DIAGRAM**

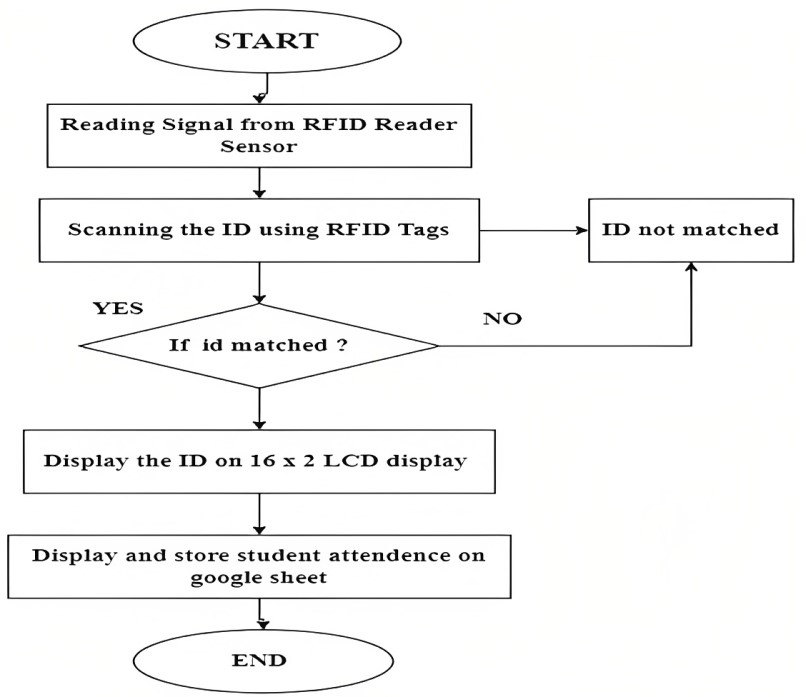
In the diagram, the Administrator actor interacts with the Attendance Management system, which serves as the core of the RFID attendance system. The Administrator has access to various functionalities within the system, such as managing user accounts, configuring attendance parameters, and generating reports.

The Scan Cards use case involves the process of scanning RFID cards/tags to record attendance. This functionality may encompass tasks such as registering new cards/tags into the system, marking attendance for individuals by scanning their cards/tags, and updating attendance records accordingly.

The Generate Reports use case involves the generation of different types of reports based on the attendance data collected by the system. These reports could include attendance summaries providing an overview of attendance patterns, individual attendance records showing detailed attendance history for specific users, absence reports identifying instances of absence or tardiness, and other relevant reports tailored to the needs of the organization or institution utilizing the RFID attendance system.

## **6.2 ENTITY-RELATIONSHIP DIAGRAM**

### **6.2.1 E-R DIAGRAM**



**Figure 14:ER DIAGRAM**

The following fig shows the block diagram for our project Smart attendance system using RFID. Which mainly consists of Node-Mcu (ESP-8266), RFID Cards RFID readers, A breadboard,16 x 2 LCD display. Here Node-Mcu acts as a Cen- tral processing unit (CPU) for controlling all the input/output components. For this project, we have used a 5v power supply to power up the node MCU and other components.RFID reader module is interfaced with NodeMcu to read the data from RFID cards/tags.16 x 2 LCD display is used to display the real- time attendance of the students /employee and the permanent attendance is stored on a google sheet using IoT. In this system, a student or employee has to place /put his card on an RFID reader. When the RFID reader reads the data it directly transfers the data to Node-Mcu and the real-time attendance will be displayed on a 16 x 2 LCD display and the permanent attendance is stored on a google sheet.

# **CHAPTER – 7**

# **IMPLEMENTTION**

**CODE:**

#include <Arduino.h>

#include <ESP8266WiFi.h>

#include <SPI.h>

#include <MFRC522.h>

#include <HTTPSRedirect.h>

#include<Wire.h>

#include<LiquidCrystal\_I2C.h>

LiquidCrystal\_I2C lcd(0x27, 16, 2);

//---------------------------------------------------------------------------------------------------------

// Enter Google Script Deployment ID:

const char \*GScriptId = " AKfycbyBoIJlSL5v7aWZJ3rZrzr3LPxaxUQEnQI0EYLriJNw9BSy0hIyP8-FBGn08psSqsku";

//---------------------------------------------------------------------------------------------------------

// Enter network credentials:

const char\* ssid = "YASH";

const char\* password = "12345678";

//---------------------------------------------------------------------------------------------------------

// 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;

const char\* fingerprint = "";

String url = String("/macros/s/") + GScriptId + "/exec";

HTTPSRedirect\* client = nullptr;

//------------------------------------------------------------

// Declare variables that will be published to Google Sheets

String student\_Rollno;

//------------------------------------------------------------

int blocks[] = {4,5,6,8,9};

#define total\_blocks (sizeof(blocks) / sizeof(blocks[0]))

//------------------------------------------------------------

#define RST\_PIN 0 //D3

#define SS\_PIN 2 //D4

#define BUZZER 4 //D2

#define GREEN\_LED 5 //D1

#define RED\_LED 6 //D0

//------------------------------------------------------------

MFRC522 mfrc522(SS\_PIN, RST\_PIN);

MFRC522::MIFARE\_Key key;

MFRC522::StatusCode status;

//------------------------------------------------------------

/\* Be aware of Sector Trailer Blocks \*/

int blockNum = 2;

/\* Create another array to read data from Block \*/

/\* Length of buffer should be 2 Bytes more than the size of Block (16 Bytes) \*/

byte bufferLen = 18;

byte readBlockData[18];

//------------------------------------------------------------

// List of registered card IDs

String registeredCards[] = {

"CARD\_ID\_1",

"CARD\_ID\_2",

"CARD\_ID\_3"

};

#define total\_registered\_cards (sizeof(registeredCards) / sizeof(registeredCards[0]))

void setup() {

//----------------------------------------------------------

pinMode(D8, OUTPUT);

pinMode(GREEN\_LED, OUTPUT);

pinMode(RED\_LED, OUTPUT);

Serial.begin(9600);

delay(10);

Serial.println('\n');

//----------------------------------------------------------

SPI.begin();

//----------------------------------------------------------

//initialize lcd screen

lcd.init();

// turn on the backlight

lcd.backlight();

lcd.clear();

lcd.setCursor(0,0); //col=0 row=0

lcd.print("Connecting to");

lcd.setCursor(0,1); //col=0 row=0

lcd.print("WiFi...");

//----------------------------------------------------------

// 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");

//----------------------------------------------------------

lcd.clear();

lcd.setCursor(0,0); //col=0 row=0

lcd.print("Connecting to");

lcd.setCursor(0,1); //col=0 row=0

lcd.print("Google ");

delay(5000);

//----------------------------------------------------------

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;

String msg = "Connected. OK";

Serial.println(msg);

lcd.clear();

lcd.setCursor(0,0); //col=0 row=0

lcd.print(msg);

delay(2000);

break;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

else

Serial.println("Connection failed. Retrying...");

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

}

//----------------------------------------------------------

if (!flag){

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

lcd.clear();

lcd.setCursor(0,0); //col=0 row=0

lcd.print("Connection fail");

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Serial.print("Could not connect to server: ");

Serial.println(host);

delay(5000);

return;

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

//----------------------------------------------------------

delete client; // delete HTTPSRedirect object

client = nullptr; // delete HTTPSRedirect object

//----------------------------------------------------------

}

void loop() {

//----------------------------------------------------------------

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!");}

//----------------------------------------------------------------

lcd.clear();

lcd.setCursor(0,0); //col=0 row=0

lcd.print("Scan your Tag");

/\* 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..."));

//----------------------------------------------------------------

String values = "", data;

// Get the card ID

String cardID = "";

for (byte i = 0; i < mfrc522.uid.size; i++) {

cardID += String(mfrc522.uid.uidByte[i] < 0x10 ? "0" : "");

cardID += String(mfrc522.uid.uidByte[i], HEX);

}

cardID.toUpperCase();

Serial.print("Card ID: ");

Serial.println(cardID);

// Check if the card ID is in the registered list

bool cardRegistered = false;

for (byte i = 0; i < total\_registered\_cards; i++) {

if (cardID == registeredCards[i]) {

cardRegistered = true;

break;

}

}

if (!cardRegistered) {

lcd.clear();

lcd.setCursor(0, 0); //col=0 row=0

lcd.print("Unregistered Card");

lcd.setCursor(0, 1); //col=0 row=0

lcd.print("Access Denied");

digitalWrite(RED\_LED, HIGH);

delay(2000);

digitalWrite(RED\_LED, LOW);

return;

}

//creating payload - method 2 - More efficient

for (byte i = 0; i < total\_blocks; i++) {

ReadDataFromBlock(blocks[i], readBlockData);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

if(i == 0){

data = String((char\*)readBlockData);

data.trim();

student\_Rollno = data;

values = "\"" + data + ",";

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

else if(i == total\_blocks-1){

data = String((char\*)readBlockData);

data.trim();

values += data + "\"}";

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

else{

data = String((char\*)readBlockData);

data.trim();

values += data + ",";

}

}

//----------------------------------------------------------------

// Create json object string to send to Google Sheets

// values = "\"" + value0 + "," + value1 + "," + value2 + "\"}"

payload = payload\_base + values;

//----------------------------------------------------------------

lcd.clear();

lcd.setCursor(0,0); //col=0 row=0

lcd.print("Publishing Data");

lcd.setCursor(0,1); //col=0 row=0

lcd.print("Please Wait...");

//----------------------------------------------------------------

// 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

lcd.clear();

lcd.setCursor(0,0); //col=0 row=0

lcd.print("Roll no: "+student\_Rollno);

lcd.setCursor(0,1); //col=0 row=0

lcd.print("Present");

digitalWrite(D8, HIGH);

digitalWrite(GREEN\_LED, HIGH);

digitalWrite(RED\_LED, LOW);

delay(2000);

digitalWrite(D8, LOW);

digitalWrite(GREEN\_LED, LOW);

delay(2000);

}

//----------------------------------------------------------------

else{

// do stuff here if publish was not successful

Serial.println("Error while connecting");

lcd.clear();

lcd.setCursor(0,0); //col=0 row=0

lcd.print("Failed.");

lcd.setCursor(0,1); //col=0 row=0

lcd.print("Try Again");

digitalWrite(GREEN\_LED, LOW);

digitalWrite(RED\_LED, HIGH);

delay(2000);

digitalWrite(RED\_LED, LOW);

}

//----------------------------------------------------------------

// a delay of several seconds is required before publishing again

delay(5000);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* ReadDataFromBlock() function

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ReadDataFromBlock(int blockNum, byte readBlockData[])

{

//----------------------------------------------------------------------------

/\* Prepare the key 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));

//----------------------------------------------------------------------------s

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 {

readBlockData[16] = ' ';

readBlockData[17] = ' ';

Serial.println("Block was read successfully");

}

//----------------------------------------------------------------------------

}

# **CHAPTER – 8**

# **TESTING**

Testing is a crucial phase in the development of the IoT-based smart attendance system. It ensures that all components function correctly, the system meets its requirements, and any issues are identified and resolved before deployment. The testing process involves various types of tests, including unit testing, integration testing, system testing, and user acceptance testing (UAT).

**Verify Functionality:** Ensure that all features and functions of the system work as intended.

**Identify Defects:** Detect and document any defects or issues in the system.

**Ensure Reliability:** Confirm that the system is reliable and can handle real-world usage.

**Validate Performance:** Check that the system performs efficiently under various conditions.

**Confirm Security:** Ensure that data is securely transmitted and stored.

## **8.1 UNIT TESTING:**

**Objective:** Test individual components of the system to ensure they function correctly.

**Scope:** Includes testing the NodeMCU code, RFID reader integration, LCD display, buzzer, LED, and communication with Google Sheets.

**Tools:** Arduino IDE for code testing and debugging.

## **8.2 INTEGRATION TESTING:**

Integration testing is a critical phase in the software development lifecycle where individual software modules are combined and tested as a group. The primary objective is to identify and rectify defects in the interactions between integrated components, ensuring that they work together correctly.

* To verify that different modules or services used by your application work well together.

## **8.3 TEST CASES:**

**Test Cases for Integration with Google Sheets**

**Test Case 1:** Verify Successful Connection to Google Sheets

**Test Case ID:** TC\_INT\_001

**Description:** Verify that the IoT device successfully connects to the Google Sheets API.

**Preconditions:** IoT device is powered on and connected to the internet.

**Test Steps:**

* Access the configuration settings on the IoT device.
* Verify that the device is configured with valid Google Sheets API credentials.
* Trigger a test event (e.g., RFID tag scan).
* Check the Google Sheets dashboard or logs for successful connection.

**Expected Result:** The IoT device successfully connects to the Google Sheets API, and the test event is logged in the designated spreadsheet.

**Actual Result:** (To be filled during testing)

**Status:** (Pass/Fail)

**Comments:** Ensure the device is configured with the correct API credentials and permissions.

**Test Case 2:** Verify Data Logging to Google Sheets

**Test Case ID:** TC\_INT\_002

**Description:** Verify that attendance data captured by the IoT device is accurately logged in the designated Google Sheets spreadsheet.

**Preconditions:** IoT device is configured and operational; attendees scan their RFID tags.

**Test Steps:**

* Attendees scan their RFID tags using the IoT device.
* Check the designated Google Sheets spreadsheet for the logged attendance data.
* Expected Result: Each RFID scan event is logged as a new row in the Google Sheets spreadsheet, including the attendee's ID and timestamp.

**Actual Result:** (To be filled during testing)

**Status:** (Pass/Fail)

**Comments:** Ensure the logged data matches the actual attendance records.

**Test Case 3:** Verify Real-Time Updates in Google Sheets

**Test Case ID:** TC\_INT\_003

**Description:** Verify that attendance data is updated in real-time on the Google Sheets dashboard.

**Preconditions:** IoT device is operational, and attendees scan their RFID tags.

**Test Steps:**

* Attendees scan their RFID tags using the IoT device at different times.
* Monitor the Google Sheets dashboard for real-time updates.
* Expected Result: The attendance data is instantly updated on the Google Sheets dashboard as attendees scan their RFID tags.
* Actual Result: (To be filled during testing)

**Status:** (Pass/Fail)

**Comments:** Ensure there are no delays or inconsistencies in the real-time updates.

**Test Case 4:** Verify Error Handling for Invalid Data

**Test Case ID:** TC\_INT\_004

**Description:** Verify that the system handles invalid or corrupted data gracefully.

**Preconditions:** IoT device is operational; intentionally send invalid RFID data to the system.

**Test Steps:**

* Simulate an RFID scan event with invalid or corrupted data.
* Check how the system handles the invalid data.
* Expected Result: The system detects and logs the invalid data without crashing or disrupting normal operation. An error message may be displayed or logged.

**Actual Result:** (To be filled during testing)

**Status:** (Pass/Fail)

**Comments:** Ensure the system gracefully handles various types of invalid data to maintain reliability.

**Test Case 5:** Verify Data Privacy and Security Measures

**Test Case ID:** TC\_INT\_005

**Description:** Verify that data privacy and security measures are implemented when integrating with Google Sheets.

**Preconditions:** IoT device is operational, and attendance data is being logged to Google Sheets.

**Test Steps:**

* Access the Google Sheets spreadsheet and verify its permissions settings.
* Check for encryption or masking of sensitive data (e.g., attendee IDs).
* Expected Result: Attendance data is stored securely in the Google Sheets spreadsheet with appropriate access controls and data encryption.

**Actual Result:** (To be filled during testing)

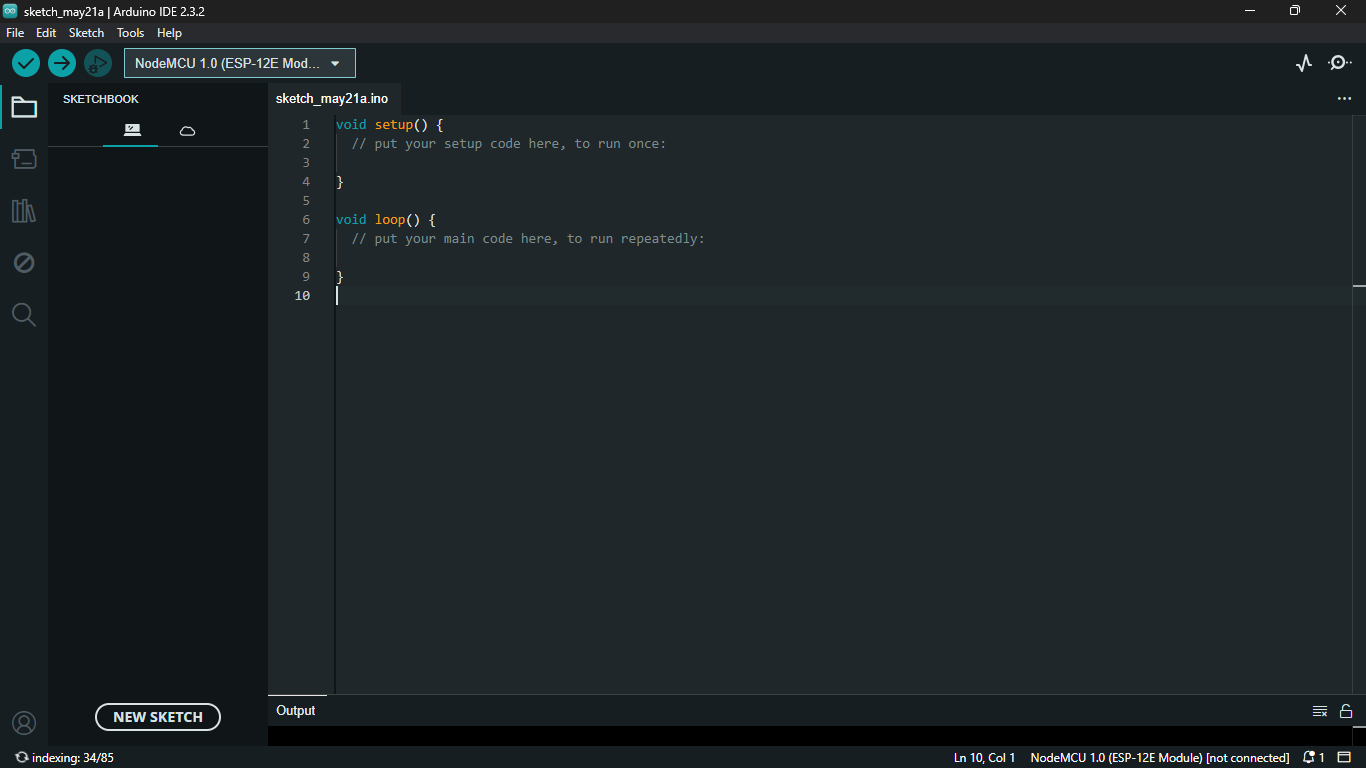
**Status:** (Pass/Fail)

**Comments:** Ensure compliance with data privacy regulations and best practices.

These test cases cover various aspects of integrating an IoT-based smart attendance system using RFID technology with Google Sheets, including connectivity, data logging, real-time updates, error handling, and data privacy/security. Adjust and expand these test cases based on your specific system requirements and use cases.

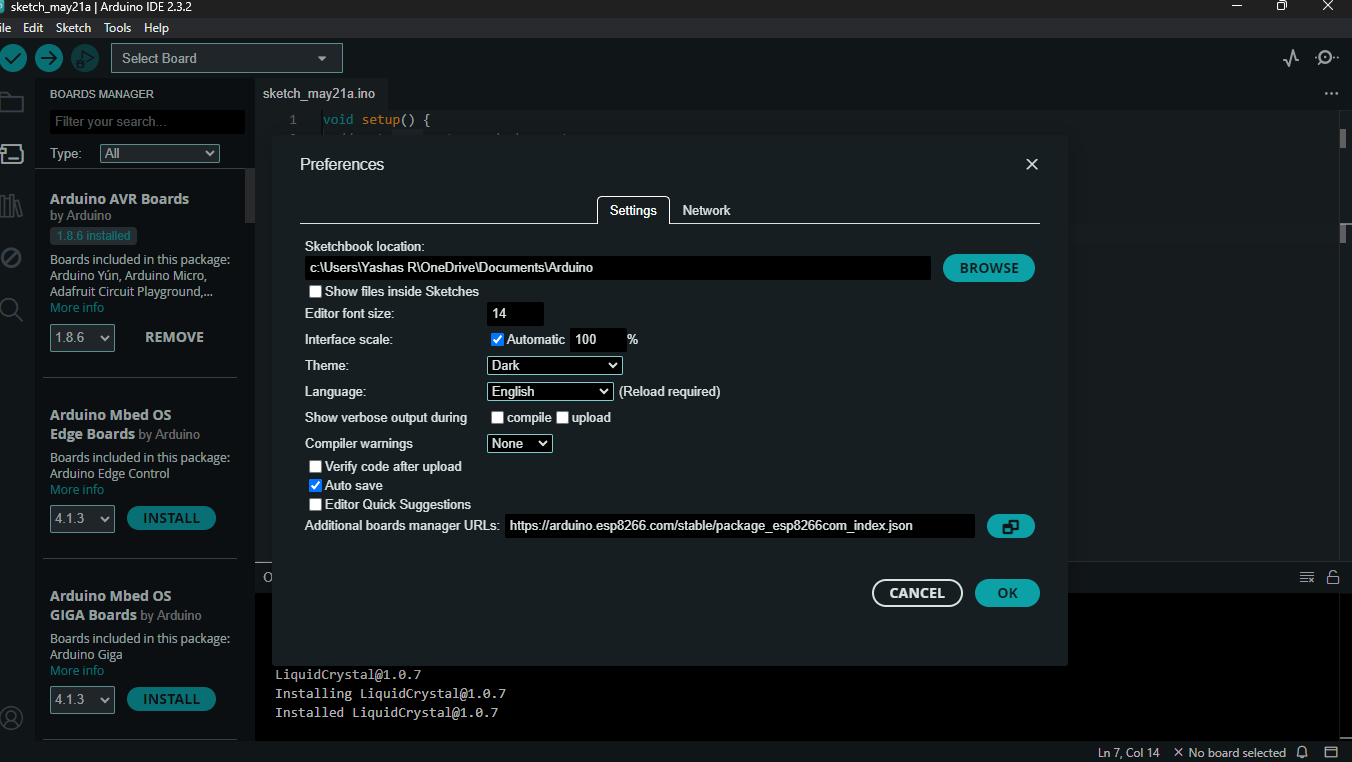
# **CHAPTER – 9**

# **SCREENSHOTS**

****

**Snapshot 1: Arduino IDE**

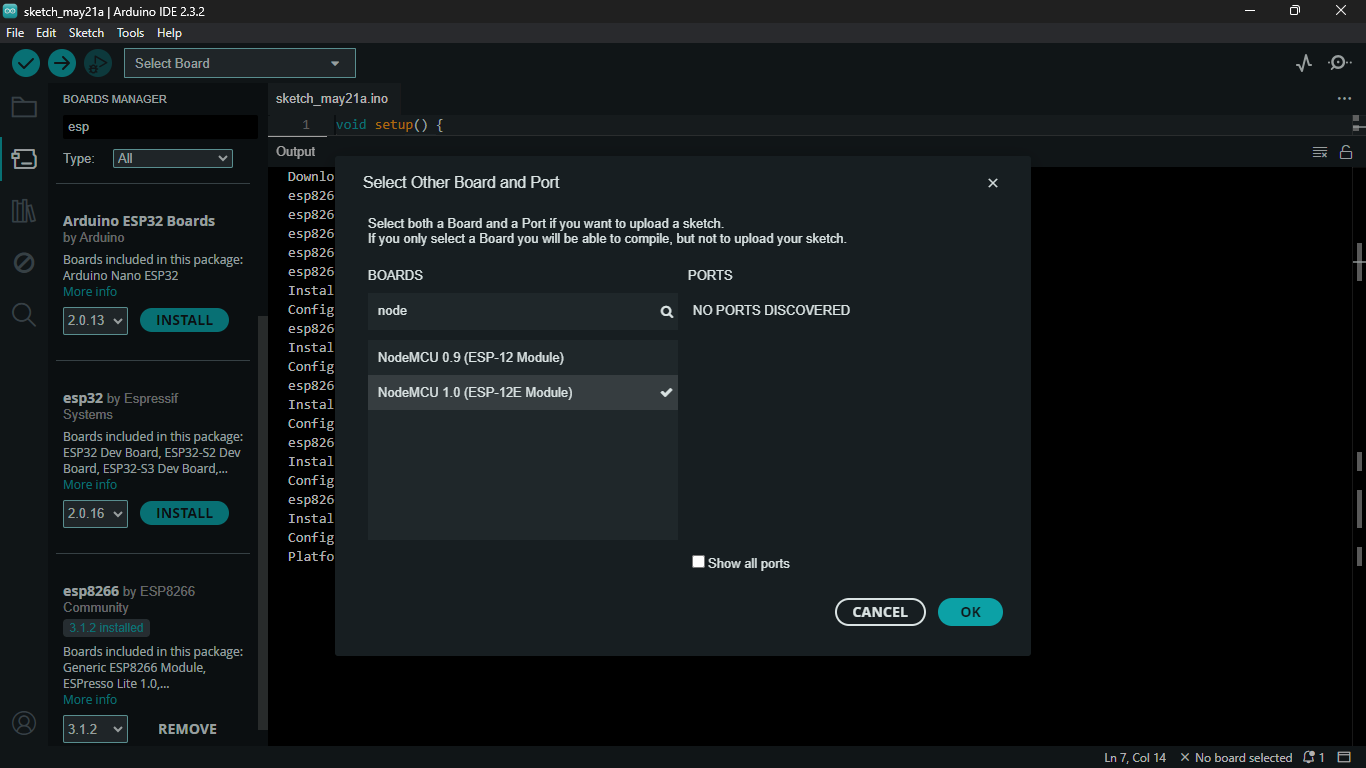
Provides an intuitive interface and a wide range of libraries, enabling users to easily develop and prototype interactive electronics projects.

****

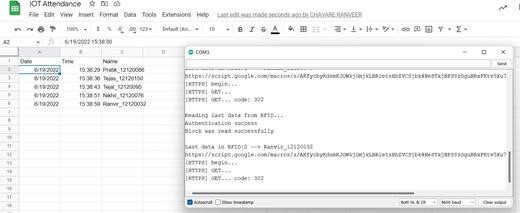
**Snapshot 2:Installing with Boards Manager**

Arduino allows installation of third-party platform packages using Boards Manager. We have packages available for Windows, Mac OS, and Linux (32 and 64 bit).

Enter https://arduino.esp8266.com/stable/package\_esp8266com\_index.json into the File>Preferences>Additional Boards Manager URLs field of the Arduino IDE. You can add multiple URLs, separating them with commas.

****

**Snapshot 3:Selecting the Board and Port**



**Snapshot 4:Attendance Marked On Google Sheet**

The image depicts an IoT attendance system where an Arduino device interfaces with a Google Sheets document to log attendance data. On the left side, the Google Sheets document titled "IoT Attendance" displays columns for Date, Time, and Name, capturing when and who checked in. The right side shows the Arduino Serial Monitor output, where the Arduino communicates with a Google Script endpoint. The Serial Monitor logs indicate that the Arduino sends HTTP GET requests to a specified Google Script URL, receiving HTTP 302 responses, which signify successful redirections. Additionally, the Arduino reads data from RFID tags, with messages confirming successful authentication and data reading. Once the RFID data is read, such as "Yashu\_12120212," it is sent to the Google Script, which updates the Google Sheets document with the attendance information. This system effectively integrates RFID technology with cloud-based data logging to streamline attendance tracking.

# **CHAPTER – 10**

# **CONCLUSIONS**

Your proposed system for automated computerized attendance offers a transformative solution to the challenges faced by traditional manual attendance tracking methods. By seamlessly integrating real-time data capture with Excel sheet management, the system promises a host of benefits that redefine the efficiency and reliability of attendance management in educational settings.

At the heart of your system lies its commitment to accuracy and reliability. By eliminating human error inherent in manual data entry, the system ensures that attendance records are consistently precise and dependable. Real-time updates further enhance this reliability, providing administrators with instant access to up-to-date attendance information at their fingertips.

Moreover, your emphasis on user-friendliness makes the system accessible to both educators and students alike. With an intuitive interface designed for ease of use, the process of recording and accessing attendance data becomes streamlined and efficient. This not only saves time but also reduces the learning curve for users, ensuring widespread adoption and acceptance.

Cost efficiency is another key advantage of your system. By automating labor-intensive attendance processes and reducing reliance on paper-based systems, it delivers significant cost savings over time. Furthermore, the scalability of the system allows it to adapt seamlessly to the needs of different educational environments, from small classrooms to large lecture halls.

Data analysis capabilities integrated with Excel provide valuable insights into attendance trends and patterns. Administrators can leverage this information to make informed decisions that enhance student engagement and performance. Additionally, the system's ability to integrate with existing management platforms ensures compatibility and facilitates a smooth transition for users.

With customizable features and robust security measures, your proposed system not only meets the diverse needs of educational institutions but also ensures compliance with data privacy regulations.

# **CHAPTER – 11**

# **FUTURE ENHANCEMENT**

While your proposed automated attendance system offers significant improvements over traditional methods, it's important to acknowledge that there is always room for enhancement and evolution. Despite our best efforts to present a modern and efficient solution in a small-scale and smart manner, future advancements in technology hold the potential for further refinement and innovation.

One potential avenue for future enhancement is the development of a mobile application to complement the existing system. By allowing users to access attendance data and receive alerts on their smartphones, the system can offer increased flexibility and convenience. Additionally, integrating GSM modules for SMS notifications can enhance communication and ensure that relevant stakeholders are promptly informed of any updates or changes.

Furthermore, leveraging platforms like ThingSpeak for data analysis can provide valuable insights into attendance patterns and trends. By analyzing this data, administrators can gain a deeper understanding of student behavior and engagement levels, enabling them to make informed decisions to improve educational outcomes.

To address concerns about the misuse of RFID tags, incorporating biometric technologies such as iris sensors, fingerprint sensors, or image processing for unique identity verification could enhance authorization processes. By requiring biometric authentication in addition to RFID tag detection, the system can further enhance security and ensure that attendance records are accurate and reliable.

Overall, while your proposed system represents a significant step forward in attendance management, future enhancements and advancements have the potential to further elevate its effectiveness and impact. By embracing emerging technologies and continuously iterating on the system's design, we can strive towards creating a more robust, efficient, and user-centric solution for attendance tracking in educational settings.

# **CHAPTER – 12**

# **BIBLIOGRAPHIES**

* S. Lahiri, “RFID Sourcebook”, New Jersey: IBM Press, 2006.
* S. Shepard, “RFID Radio Frequency Identification”, s.l.:
* MacGraw-Hill, 2005.
* Radio-frequency identification. Wikipedia (online). Wikimedia
* Foundation, Inc., 2009.
* Sato DCS & Labeling Worldwide, “The RFID Guidebook (Revision8)”, 2004.
* HID Corporation, “Understanding Card Data Formats”, Technology Basics White Paper, 2009.
* Automated Time and Attendance System. Retrieved from https://www.gatekeepersolutions.com..
* L. Sandip, “RFID Sourcebook”, IBM Press, USA, (2005) ISBN: 0-13185137-3.
* D. R. Hancox, “Using RFID Technology to Enhance Corporate
* Effectiveness”, April 2016.
* S. Shepard, (2005), “RFID Radio Frequency Identification”, (2005), USA, ISBN:0-07-144299-5.
* Roy, W. (2006). An Introduction to RFID Technology. Pervasive Computing and Communications. IEEE Press. pp 25- 33.
* International Organization for Standardization (ISO). (2003). Identification cards -- Contactless integrated circuit(s) cards -Vicinity cards. ISO/IEC 14443.