19Z604 - EMBEDDED SYSTEMS LABORATORY

NFC-CARD ATTENDANCE SYSTEM

Team - 6

J Dhanvanth- 21z309

KV Muralidhar- 21z312

M Prajeeth- 21z330

RK Ranjith- 21z335

S ShrishGogul-21z339

Shoaib- 21z349

BACHELOR OF ENGINEERING

Branch: COMPUTER SCIENCE AND ENGINEERING

of Anna University

April 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PSG COLLEGE OF TECHNOLOGY

COIMBATORE – 641 004

NFC-Based Attendance System with Real-time Monitoring and Data Logging

1.Problem Statement:

The project aims to develop an embedded system utilizing NFC technology, along with attendance, distance, and count sensors, to create an automated attendance recording system. The system will integrate Arduino, ESP32, and WiFi modules to enable real-time monitoring and data logging. Attendance data will be updated in Google Sheets for easy access and analysis. Key challenges include sensor integration, system calibration, web interface development, and secure data transmission.

2.Here's a list of common problems faced in day-to-day attendance recording and how your project aims to rectify them:

1. Manual Attendance Recording Errors: Traditional methods of manual attendance recording are prone to errors due to illegible handwriting, incorrect entries, or misplaced records. Your project automates the attendance recording process through NFC technology, reducing the chances of human error.

2. Time-consuming Attendance Management: Manual attendance management can be time-consuming for both students/employees and administrative staff. Your system allows for quick and convenient attendance marking with NFC cards, saving time for both parties involved.

3. Limited Real-time Monitoring: Traditional attendance systems often lack real-time monitoring capabilities, making it difficult for administrators to track attendance trends or identify discrepancies promptly. Your project provides real-time monitoring through a web-based interface, enabling administrators to access attendance records and system status remotely.

4. Inefficient Data Logging : Paper-based attendance records can be inefficient to log and manage, leading to delays in data entry and retrieval. Your system automates data logging

by updating attendance records directly in Google Sheets, ensuring timely and accurate data management.

5. Security Concerns : Paper-based attendance records are vulnerable to theft, tampering, or unauthorized access. Your project enhances security by utilizing NFC technology for identification and authentication, reducing the risk of fraudulent attendance marking.

6. Difficulty in Data Analysis : Analyzing attendance data manually from paper records can be challenging and time-consuming. Your system facilitates easy access and analysis of attendance data stored in Google Sheets, allowing administrators to track attendance trends and identify patterns more effectively.

7. Inconvenient Attendance Tracking : Traditional attendance methods may require physical presence for marking attendance, leading to inconvenience for users, especially in remote or large-scale settings. Your system offers a convenient solution with NFC cards for quick and hassle-free attendance marking.

3.Components:

Sure, here's a list of components required for your project along with brief descriptions for each:

1. Arduino Board: The Arduino board serves as the main microcontroller unit responsible for interfacing with various sensors and peripherals, processing data, and controlling system operations.

2. ESP32 Board: The ESP32 board provides WiFi connectivity and serves as the communication interface between the embedded system and the web-based monitoring interface. It enables real-time data transmission and remote access to attendance records.

3. NFC Reader Module: The NFC reader module is used to read NFC cards for identification and authentication purposes. It allows users to quickly scan their NFC cards to mark attendance.

4.Distance Sensor[UltraSonic Sensor]: The distance sensor measures the proximity of objects and is used to detect the presence of individuals within the attendance scanning range. It ensures accurate attendance marking and prevents unauthorized attempts.

5. Count Sensor[IR Sensor] : The count sensor keeps track of the number of times the attendance system is scanned. It helps in monitoring system usage and detecting any anomalies or unusual activity.

6. WiFi Module : The WiFi module enables wireless communication between the embedded system and the web-based monitoring interface. It facilitates real-time data transmission and updates attendance records on Google Sheets.

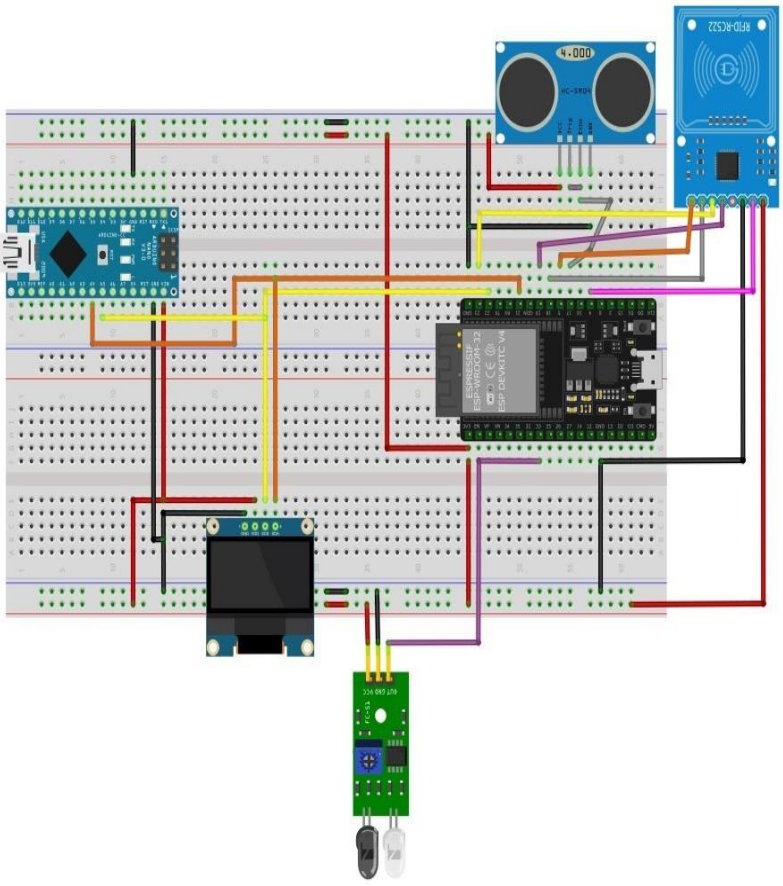
7. LED Indicators[OLED] : LED indicators provide visual feedback to users, indicating successful attendance scans, system status, and any error conditions. They enhance the user experience and improve system usability.

8. Web-Based Monitoring Interface : The web-based monitoring interface is a graphical user interface accessible through a web browser. It allows administrators to remotely monitor attendance records, view distance measurements, and track system usage in real-time.

9. Google Sheets Integration : Google Sheets integration enables automatic updating of attendance records in a cloud-based spreadsheet. It provides a convenient and accessible platform for storing and analyzing attendance data.

These components work together to create a comprehensive NFC-based attendance system with real-time monitoring and data logging capabilities, offering efficient and convenient attendance management for educational institutions and workplaces.

4.Schematic Diagram [Done using Fritzing Software]:



5.Code

```
#include <SPI.h>
#include <MFRC522.h>
#include <WiFi.h>
#include <HTTPClient.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <WebServer.h>

const int IR_Sensor = 33;
volatile uint64_t count = 0;
volatile uint64_t currentCount = -1;
unsigned long currentTime = 0;
unsigned long lastReadTime = 0;
unsigned int intervalDelay = 1000;

const int pingPin = 17;
long duration, inches, cm;

#define SS_PIN 5
#define RST_PIN 4
#define BUZZ_PIN 14
#define BUZZ_CHANNEL 2
#define DOOR_PIN 2
#define TERMINAL_NAME  "basement"
```

Code for linking spreadsheet for updating attendance:

```
const char *mainLinkForSpr =
"https://script.google.com/macros/s/AKfycbwOqbq0JxCyr2idJdhFCj3EqHPTbQJT11xqtLc_BZ
nNgZ5_zNKZw3qJXG5QgfYyE2n-LQ/exec";
const char *ssid = "Embedded";
const char *password = "prajeeth";
#define OLED_SDA 21
#define OLED_SCL 22
#define SCREEN_WIDTH  128
#define SCREEN_HEIGHT  32
```

```
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
MFRC522 mfrc522(SS_PIN, RST_PIN);
WebServer server(80);
MFRC522::StatusCode status;

uint64_t clearDisplayTimer = 0;
bool needDisplayUpdate = true;

void IRAM_ATTR isr()
{
  currentTime = millis();
```

IR Sensor is noisy so we add a debounce mechanism:

```
  if (currentTime - lastReadTime > intervalDelay)
  {
    count++;
    lastReadTime = currentTime;
  }
}
void dualPrint(const char* text) {
  display.print(text);
  Serial.println(text);
}
```

Website developed using HTML,CSS,JAVASCRIPT[For distance and count updation]:

```
const char html_page[] PROGMEM = R"RawString(
<!DOCTYPE html>
<html>
  <style>
    body {font-family: sans-serif;}
    h1 {text-align: center; font-size: 30px;}
    p {text-align: center; color: #4CAF50; font-size: 40px;}
  </style>
<body>
  <h1>Distance Measurement</h1><br>
  <p>Distance in CM : <span id="_CM">0</span> CM</p>
  <p>Distance in Inch : <span id="_INCH">0</span> Inch</p>
  <p>IR Sensor Count : <span id="_COUNT">0</span></p>
<script>
```

JavaScript fucntions to update Distance and IR count :

```
function updateIRCount() {
   var xhttp = new XMLHttpRequest();
   xhttp.onreadystatechange = function() {
     if (this.readyState == 4 && this.status == 200) {
       const count = this.responseText;
       document.getElementById("_COUNT").innerHTML = count;
     }
   };
   xhttp.open("GET", "readIRCount", true); // Endpoint to fetch IR sensor count data
   xhttp.send();
 }
 function updateDistance() {
   var xhttp = new XMLHttpRequest();
   xhttp.onreadystatechange = function() {
     if (this.readyState == 4 && this.status == 200) {
       const text = this.responseText;
       const myArr = JSON.parse(text);
       document.getElementById("_CM").innerHTML = myArr[0];
       document.getElementById("_INCH").innerHTML = myArr[1];
     }
   };
   xhttp.open("GET", "readDistance", true);
   xhttp.send();
 }
 // Initial update
 updateDistance();
 updateIRCount();
 setInterval(function() {
   updateDistance();
   updateIRCount();
 }, 50);
</script>
</body>
</html>
)RawString";
void MainPage() {
  String _html_page = html_page;
  server.send(200, "text/html", _html_page);
}
```

```
void Distance() {
  String data = "[\""+String(cm)+"\",\""+String(inches)+"\"]";
  server.send(200, "text/plane", data);
}
```

Buzzer Beep Generator Function:

```
void beep(int count = 1) {
  ledcSetup(BUZZ_CHANNEL, 5000, 10);
  ledcAttachPin(BUZZ_PIN, BUZZ_CHANNEL);
  for (size_t j = 0; j < count; j++) {
    if (j != 0)
      delay(300);
    for (int i = 200; i < 1000; i++) {
      ledcWrite(BUZZ_CHANNEL, i);
      delayMicroseconds(30);
    }
    ledcWrite(BUZZ_CHANNEL, 0);
  }
  ledcDetachPin(BUZZ_PIN);
  pinMode(BUZZ_PIN, INPUT);
}

void openDoor() {
  digitalWrite(DOOR_PIN, HIGH);
  delay(2000);
  digitalWrite(DOOR_PIN, LOW);
}

void setup()
{
  pinMode(IR_Sensor, INPUT);
  attachInterrupt(digitalPinToInterrupt(IR_Sensor), isr, FALLING);

  pinMode(DOOR_PIN, OUTPUT);
  Serial.begin(115200);
  SPI.begin();
```

Connecting WIFI Module:

```
  mfrc522.PCD_Init();
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
```

```
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println(WiFi.localIP());
  Wire.begin(OLED_SDA, OLED_SCL);
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 allocation failed"));
    for (;;) {}
  }
  server.on("/", MainPage);
  server.on("/readDistance", Distance);
  server.on("/readIRCount", IRCount);
  server.begin();
  delay(2000);
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 0);
  display.print("Ready");
  display.display();
    beep(2);
}
void clearDisplayIn(int mSec = 5000) {
  clearDisplayTimer = millis() + mSec;
  needDisplayUpdate = true;
}
```

Handling data in Excel Sheet:

```
void handleDataFromGoogle(String data) {
  int colonIndex = data.indexOf(":");
  String accessType = data.substring(0, colonIndex);
  int nextColonIndex = data.indexOf(":", colonIndex + 1);
  String name = data.substring(colonIndex + 1, nextColonIndex);
  String text = data.substring(nextColonIndex + 1, data.length());

  display.clearDisplay();
  display.setCursor(0, 0);
  display.print("Hi ");
  display.print(name);
  display.setCursor(0, 8);
  display.print(text);
```

```
  display.display();

  if (accessType.equalsIgnoreCase("beep")) {
    beep(5);
  } else if (accessType.equalsIgnoreCase("door")) {
    openDoor();
  }
}
void getGoogleData()
{
  HTTPClient http;
  String data;

  display.clearDisplay();
  uint64_t time = esp_timer_get_time();
  char url[150];
  int pointerShift = sprintf(url, "%s?uid=", mainLinkForSpr);

  for (size_t i = 0; i < mfrc522.uid.size; i++)
  {
    pointerShift += sprintf(url + pointerShift, "%X", mfrc522.uid.uidByte[i]);
  }

#ifdef TERMINAL_NAME
  pointerShift += sprintf(url + pointerShift, "&terminal=%s", TERMINAL_NAME);
#endif

  Serial.println(url);
  Serial.println(F("Connecting to google"));

  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 0);
  display.print("Connecting to");
  display.setCursor(0, 10);
  display.print("Google");
  display.display();
  http.begin(url, root_ca);

  const char *location = "Location";
  const char *headerKeys[] = {location};
```

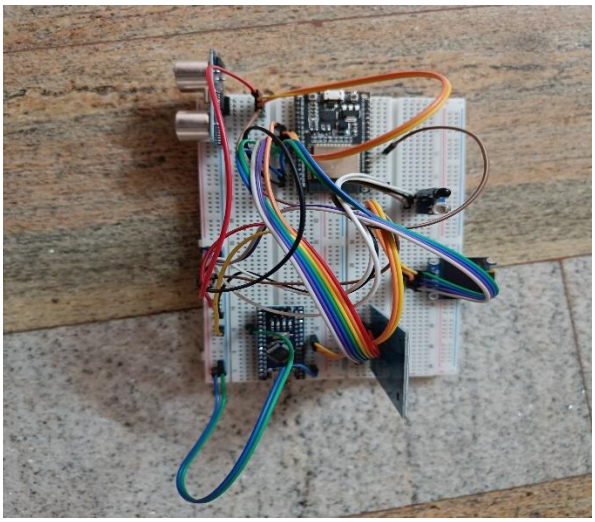
```
http.collectHeaders(headerKeys, 1);
int code = http.GET();
Serial.printf("code %d\n", code);
// 302 code means redirect
if (code == 302)
{
  String newUrl = http.header(location);
  http.end();

  Serial.println(newUrl);
  http.begin(newUrl, root_ca);
  code = http.GET();
  Serial.printf("status code %d\n", code);

  data = http.getString();
  Serial.println(data);

  display.clearDisplay();
  display.setCursor(0, 0);
  display.print(data);
  display.display();
}
else
{
  display.clearDisplay();
  display.setCursor(0, 0);
  display.print(code);
  if (code == 403 || code == -1)
  {
    display.setCursor(0, 10);
    display.print("Err open terminal");
    display.setCursor(0, 20);
    display.print("for help");
    if (code == -1)
    {
      Serial.println(F("If it says something like start_ssl_client error"));
      Serial.print(F("try to update the SSL certificate"));
    }
    else
    {
      Serial.print(F("Open this link in any browser: "));
```

```
        Serial.println(url);
        Serial.println(F("If it says Authorization is ..."));
        Serial.println(F("Open the Google script and republish it"));
      }
    }
    else
    {
      display.setCursor(0, 10);
      display.print(F("Something went wrong"));
    }
  }

  if (!data.isEmpty() && data.length() > 1)
  {
    handleDataFromGoogle(data);
  }

  Serial.printf("time=%d\n", esp_timer_get_time() - time);
  clearDisplayIn();
}

void loop() {

    pinMode(pingPin, OUTPUT);
  digitalWrite(pingPin, LOW);
  delayMicroseconds(2);
  digitalWrite(pingPin, HIGH);
  delayMicroseconds(5);
  digitalWrite(pingPin, LOW);
  pinMode(pingPin, INPUT);
  duration = pulseIn(pingPin, HIGH);

  cm = microsecondsToCentimeters(duration);
  inches = microsecondsToInches(duration);

  server.handleClient();
  Serial.print(inches);
  Serial.print("in, ");
  Serial.print(cm);
  Serial.print("cm");
  Serial.println();
```

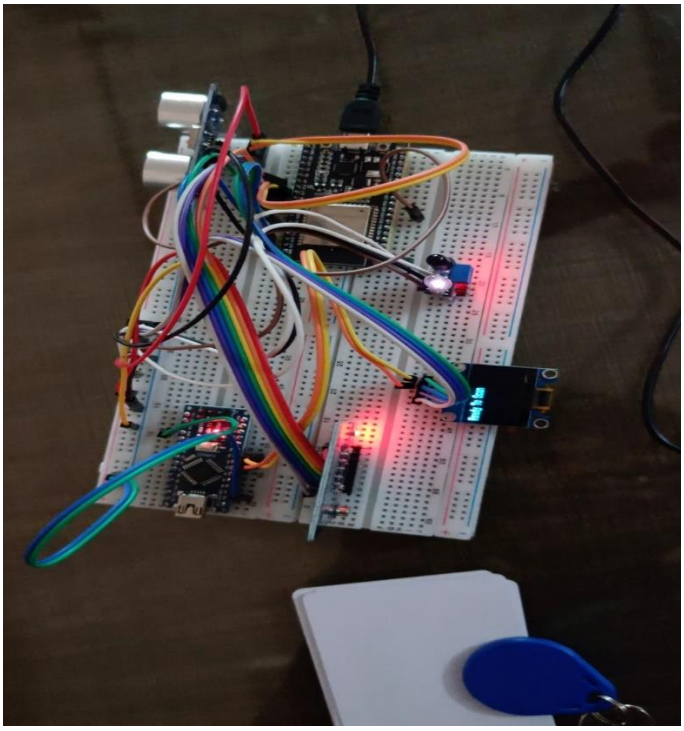
```
  if (needDisplayUpdate && millis() > clearDisplayTimer) {
    display.clearDisplay();
    display.setCursor(1, 1);
    display.print("Ready To Scan");
    display.display();
    needDisplayUpdate = false;
    //clearDisplayIn(1000);
  }
  if (!mfrc522.PICC_IsNewCardPresent()) {
    return;
  }

  if (!mfrc522.PICC_ReadCardSerial()) {
    return;
  }

  beep();
  display.clearDisplay();
  display.setCursor(0, 0);
  display.print("Scanning Card...");
  display.display();

  getGoogleData();

  mfrc522.PICC_HaltA();
  mfrc522.PCD_StopCrypto1();
  beep();

}
void IRCount() {
  server.send(200, "text/plain", String(count));
}
long microsecondsToInches(long microseconds) {return microseconds / 74 / 2;}
long microsecondsToCentimeters(long microseconds) {return microseconds / 29 / 2;}
```

6. OUTPUT SCREENSHOTS

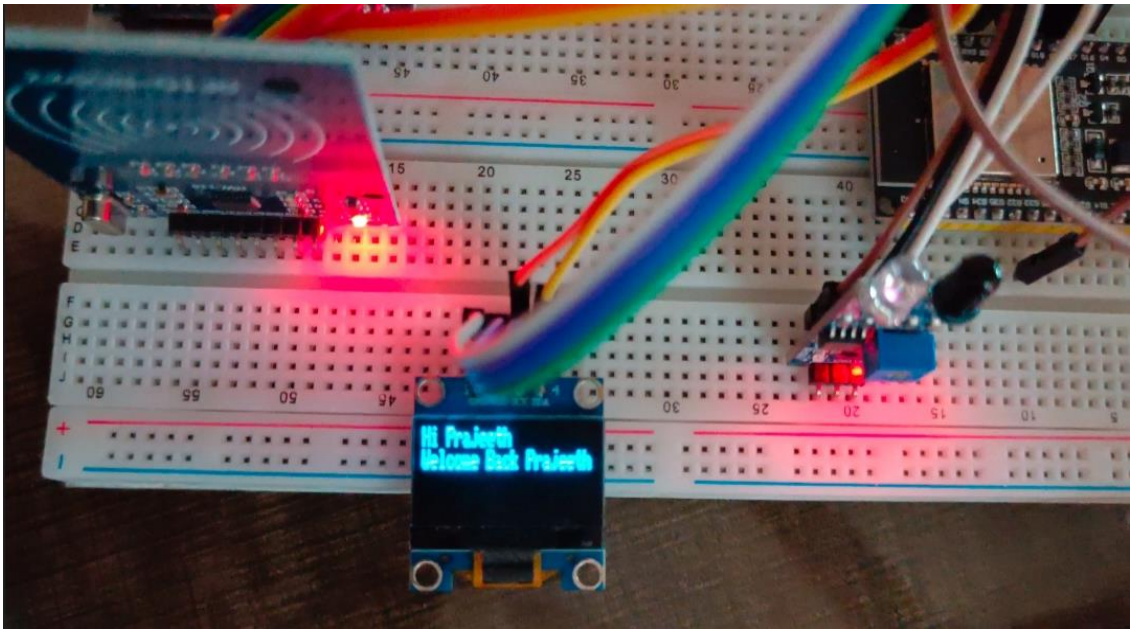
1.Hardware Interface



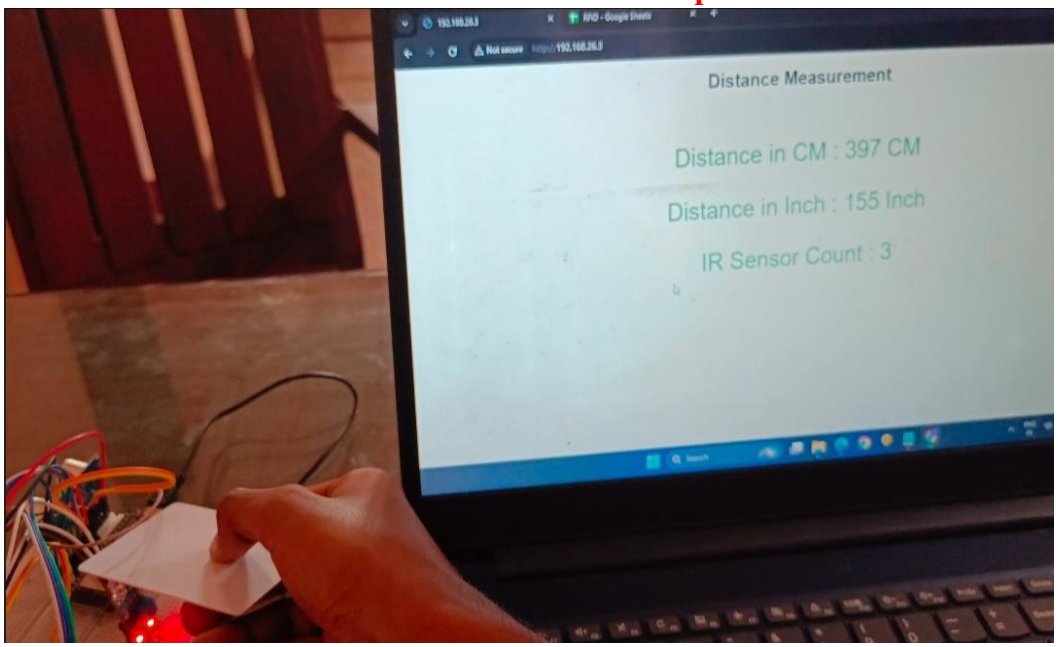
2.Ready To Scan



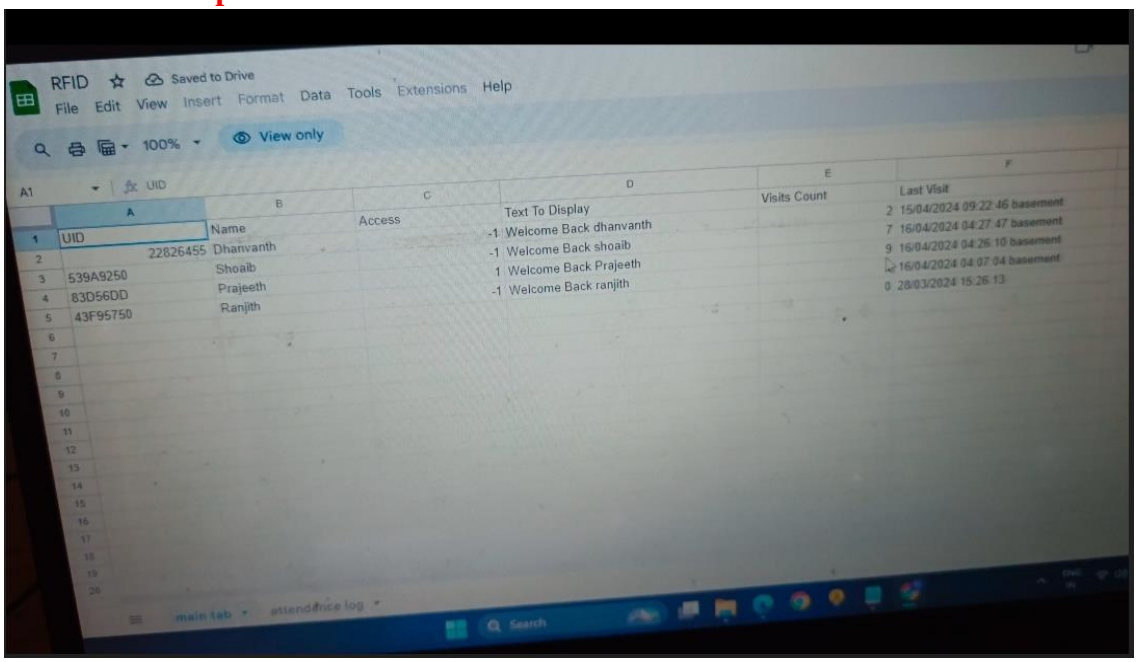
3.Nfc-Card Scanned



4.Website Distance Measurement And IR Count Updation:



5.Attendance Updation In Excel:



7.Conclusion:

The NFC card attendance project offers a convenient and efficient solution for tracking attendance in various settings such as schools, offices, or events. By utilizing an RFID scanner like the RC522, an ESP32 microcontroller, and accompanying sensors like ultrasonic and IR, the system can accurately record attendance by scanning NFC cards. The OLED display provides real-time feedback, enhancing user interaction. Arduino Nano can be used as an alternative microcontroller for smaller-scale implementations. This project demonstrates the integration of hardware components and software programming to create a practical attendance management system.

8.References:

- MFRC522 library: https://github.com/miguelbalboa/rfid
- ESP32 Arduino Core: https://github.com/espressif/arduino-esp32
- Adafruit SSD1306 library: https://github.com/adafruit/Adafruit_SSD1306
- Ultrasonic sensor library: https://github.com/JRodrigoTech/Ultrasonic-HC-SR04
- Arduino IRremote library: https://github.com/Arduino-IRremote/Arduino-IRremote