19Z604 - EMBEDDED SYSTEMS LABORATORY

NFC-CARD ATTENDANCE SYSTEM

Team - 6

J Dhanvanth- 21z309
KV Muralidhar- 21z312
M Prajeeth- 21z330
RK Ranjith- 21z335
S Shrish Gogul-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

(Autonomous Institution)

COIMBATORE – 641 004

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

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.

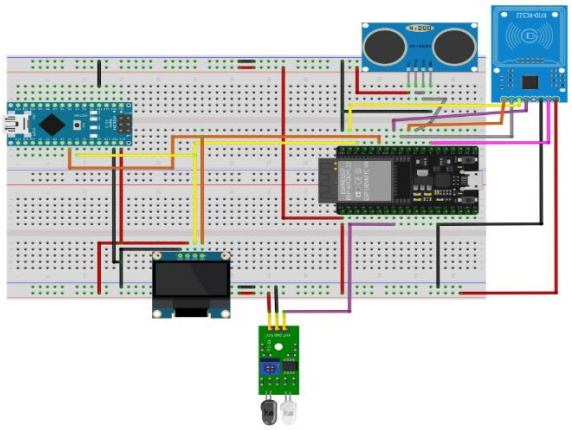
COMPONENTS USED

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

1. Arduino Board
2. ESP32 Board
3. NFC Reader Module
4. Distance Sensor
5. Count Sensor
6. WIFI Module
7. LED Indicators
8. Web-Based Monitoring Interface
9. Google Sheets Integration

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

SCHEMATIC DIAGRAM



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_BZnNg
Z5_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]);
  }
```

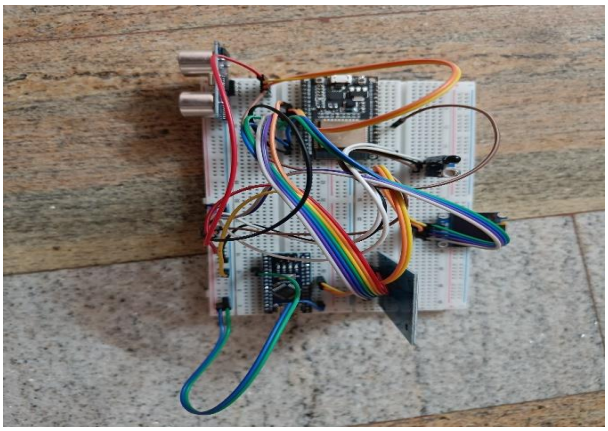
```
#ifdef TERMINAL_NAME
  pointerShift += sprintf(url + pointerShift, "&terminal=%s", TERMINAL_NAME);
#endif

  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();
```

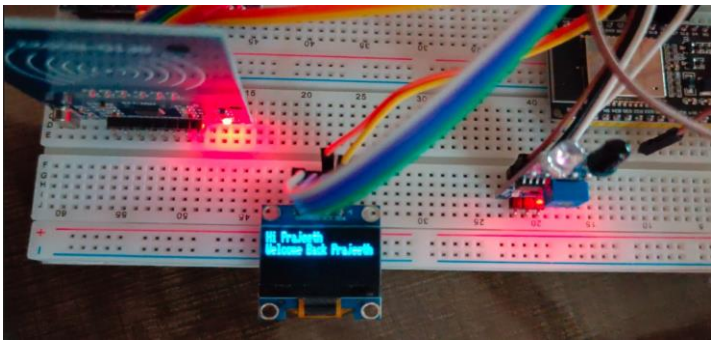
```
  Serial.print(inches);
  Serial.print("in, ");
  Serial.print(cm);
  Serial.print("cm");
  Serial.println();


  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;}
```

OUTPUT:

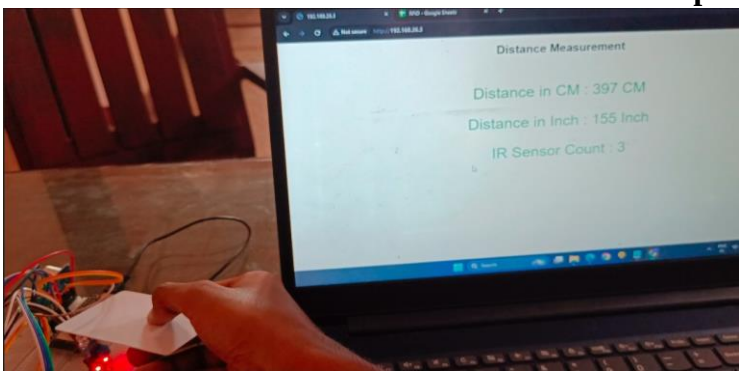
1.Hardware Interface



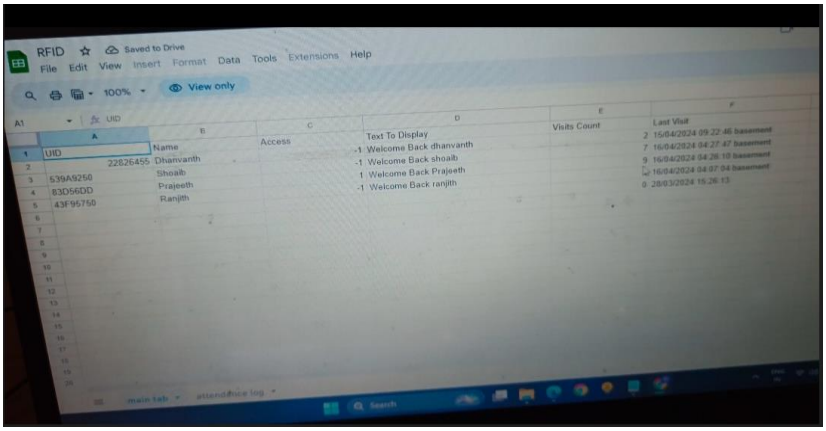
2.Nfc-Card Scanned



3.Website Distance Measurement And IR Count Updation



4.Attendance Updation in Excel



Result:

The implementation of the NFC attendance system resulted in a functional solution for tracking attendance using RFID technology. By integrating hardware components like an RFID scanner, microcontroller, and sensors, along with software programming, the system offers a convenient and efficient way to record attendance in various settings. The result is a practical tool that enhances attendance management processes with real-time feedback and easy user interaction.