//Seet Monitor Code ESP32 Dev Module

#include <WiFi.h>

//#include <ESP8266WebServer.h>

#include <LiquidCrystal\_I2C.h>

#include <Wire.h>

#include <SPI.h>

#include "HX711.h"

LiquidCrystal\_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);// 2 x 16 lcd pin outs with driver

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

float Relay11,  Relay22, Relay33, Relay44, Relay55,Relay66, Buzzer1 ;

/\*Put your SSID & Password\*/

const char\* ssid = "22222222";  // Enter SSID here

const char\* password = "22222222";  //Enter Password here

//ESP8266WebServer server(80);

WiFiServer server(80);

String header;

String Relay1State = "off";

String Relay2State = "off";

String Relay3State = "off";

String Relay4State = "off";

String Relay5State = "off";

String Relay6State = "off";

String Relay7State = "off";

String Relay8State = "off";

String Relay9State = "off";

String Relay10State = "off";

String PositionState = "Normal";

String PositionState2 = "Normal";

// Current time

unsigned long currentTime = millis();

// Previous time

unsigned long previousTime = 0;

// Define timeout time in milliseconds (example: 2000ms = 2s)

const long timeoutTime = 2000;

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

byte value1, Value2, value3, value4;

const int Relay1 = 13;

const int Relay2 = 12 ;

const int Relay3 = 14;

const int Relay4 = 27;

const int Relay5 = 26;

const int Relay6 = 32;

const int Relay7 = 15;//15

const int Relay8 = 4;

const int Relay9 = 33;

const int Relay10 = 36;

const int Buzzer = 25;

//const int SmallVibrator = 25;

const int IR1 = 34;

const int IR2 = 35;

// HX711 pinout on ESP32

// HX711 circuit wiring

const int DOUTPIN = 39;// LOADCELL DOUT PIN VN 39

const int LSCKPIN = 15; //LOADCELL SCK PIN

HX711 scale;

int IR1State;

int IR2State;

int count = 0;

float Weght;

float Position;

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/\* 0- General \*/

int decimalPrecision = 2;                   // decimal places for all values shown in LED Display & Serial Monitor

unsigned long previousMillis = 0;

unsigned long interval = 30000;

////////////////////////////////////////////////////////////////////////////////Setup/////////////////////////////////////////////////////

void setup() {

      pinMode(Relay1, OUTPUT);

      pinMode(Relay2, OUTPUT);

      pinMode(Relay3, OUTPUT);

      pinMode(Relay4, OUTPUT);

      pinMode(Relay5, OUTPUT);

      pinMode(Relay6, OUTPUT);

      pinMode(Relay7, OUTPUT);

      pinMode(Relay8, OUTPUT);

      pinMode(Relay9, OUTPUT);

      pinMode(Relay10, OUTPUT);

      pinMode(Buzzer, OUTPUT);

      //pinMode(SmallVibrator, OUTPUT);

      pinMode(IR1, INPUT);

      pinMode(IR2, INPUT);

      digitalWrite(Relay1, HIGH);

      digitalWrite(Relay2, HIGH);

      digitalWrite(Relay3, HIGH);

      digitalWrite(Relay4, HIGH);

      digitalWrite(Relay5, HIGH);

      digitalWrite(Relay6, HIGH);

      digitalWrite(Relay7, HIGH);

      digitalWrite(Relay8, HIGH);

      digitalWrite(Relay9, HIGH);

      digitalWrite(Relay10, HIGH);

      digitalWrite(Buzzer, HIGH);

      ///pinMode(SmallVibrator, LOW);

/////////////////////////////////////////////////////////////////////////////////////

  Serial.begin(115200);

  Serial.println("Connecting to ");

  Serial.println(ssid);

  //connect to your local wi-fi network

  WiFi.begin(ssid, password);

  //check wi-fi is connected to wi-fi network

  while (WiFi.status() != WL\_CONNECTED) {

  delay(1000);

  Serial.print(".");

  }

  Serial.println("");

  Serial.println("WiFi connected..!");

  Serial.print("Go t0 IP:");  Serial.println(WiFi.localIP());

  server.begin();

  Serial.println("HTTP server started");

  lcd.begin(20, 4);

  lcd.clear();

 /// startMillislcd = millis();

  lcd.setCursor(0,0);

  lcd.print("Seet Monitor");

  lcd.setCursor(0,1);

  lcd.print("WiFi Connected");

  lcd.setCursor(0,2);

  lcd.print(WiFi.localIP());

  delay(5000);

  /////////////////////////////////////////////// HX711 ////////////////////////////////////////////////

  scale.begin(DOUTPIN, LSCKPIN);

  Serial.println("Before setting up the scale:");

  Serial.print("read: \t\t");

  Serial.println(scale.read());     // print a raw reading from the ADC

  /\*

  Serial.print("read average: \t\t");

  Serial.println(scale.read\_average(20));   // print the average of 20 readings from the ADC

  Serial.print("get value: \t\t");

  Serial.println(scale.get\_value(5));   // print the average of 5 readings from the ADC minus the tare weight (not set yet)

  Serial.print("get units: \t\t");

  Serial.println(scale.get\_units(5), 1);  // print the average of 5 readings from the ADC minus tare weight (not set) divided

  // by the SCALE parameter (not set yet)

  /\*

  scale.set\_scale(2280.f);                      // this value is obtained by calibrating the scale with known weights; see the README for details

  scale.tare();               // reset the scale to 0

  Serial.println("After setting up the scale:");

  Serial.print("read: \t\t");

  Serial.println(scale.read());                 // print a raw reading from the ADC

  Serial.print("read average: \t\t");

  Serial.println(scale.read\_average(20));       // print the average of 20 readings from the ADC

  Serial.print("get value: \t\t");

  Serial.println(scale.get\_value(5));   // print the average of 5 readings from the ADC minus the tare weight, set with tare()

  Serial.print("get units: \t\t");

  Serial.println(scale.get\_units(5), 1);        // print the average of 5 readings from the ADC minus tare weight, divided

  // by the SCALE parameter set with set\_scale

\*/

  //Serial.println("Readings:");

}

////////////////////////////////////////////////////////////////////////////////End Setup///////////////////////////////////////////////////

void loop() {

 // Get the current Beats Per Minute (BPM)

IR1State = digitalRead(IR1);

IR2State = digitalRead(IR2);

  Serial.print("one reading:\t");

  Serial.print(scale.get\_units(), 1);

  Serial.print("\t| average:\t");

  Serial.println(scale.get\_units(10), 1);

  ///scale.power\_down();              // put the ADC in sleep mode

 /// delay(5000);

  ///scale.power\_up();

 // int Weght = 6 ;

int Weght=(scale.get\_units(10), 1);

Serial.print(Weght);

  lcd.clear();

  lcd.setCursor(0,0);

  lcd.print("Seet Monitor");

  lcd.setCursor(0, 1);

  lcd.print("kg1: ");

  lcd.setCursor(8, 1);

  lcd.print(Weght);

   Serial.print("Position: ");

   Serial.println(PositionState);

   delay(2000);

    ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

if (Weght >20) {

  lcd.clear();

  lcd.setCursor(0,0);

  lcd.print("Seet Monitor");

  lcd.setCursor(0,1);

  lcd.print("Over Weight");

   lcd.setCursor(0,2);

  lcd.print("Weight kg:");

  lcd.setCursor(10, 2);

  lcd.print(Weght);

   Serial.println("Over Weight");

   digitalWrite(Buzzer, LOW);

   delay(2000);

}

if (Weght <10) {

PositionState2="Person Not On Position";

  lcd.clear();

  lcd.setCursor(0,0);

  lcd.print("Seet Monitor");

  lcd.setCursor(0,1);

  lcd.print("Preson NOT ");

   lcd.setCursor(0,2);

  lcd.print("On Position");

   Serial.println("Person not On Position");

   digitalWrite(Buzzer, LOW);

   delay(2000);

}

else {

  lcd.clear();

  lcd.setCursor(0,0);

  lcd.print("Seet Monitor");

  digitalWrite(Buzzer, HIGH);

   lcd.setCursor(0, 1);

  lcd.print("Position:");

  lcd.setCursor(10, 2);

  lcd.print("Normal");

  PositionState2= ("Normal");

}

if (IR1State == HIGH) {

  lcd.clear();

  lcd.setCursor(0,0);

  lcd.print("Seet Monitor");

  lcd.setCursor(0, 1);

  lcd.print("Position Wrong");

  lcd.setCursor(0,2);

  lcd.print("Seating Position");

  lcd.setCursor(0,3);

  lcd.print("Take Rest");

PositionState="Wrong Seating Position";

  Serial.println("Wrong Seating Position");

  Serial.println("Take Rest");

   digitalWrite(Buzzer, LOW);

    //digitalWrite(SmallVibrator, HIGH);

 delay(2000);

}

else {

  lcd.clear();

  lcd.setCursor(0,0);

  lcd.print("Seet Monitor");

  digitalWrite(Buzzer, HIGH);

  //digitalWrite(SmallVibrator, LOW);

   lcd.setCursor(0, 1);

  lcd.print("Seating Position:");

  lcd.setCursor(10, 2);

  lcd.print("Normal");

  PositionState= ("Normal");

  delay(3000);

}

if (IR2State == HIGH) {

  lcd.clear();

  lcd.setCursor(0,0);

  lcd.print("Seet Monitor");

  lcd.setCursor(0, 1);

  lcd.print("Position Wrong");

  lcd.setCursor(0,2);

  lcd.print("Seating Position");

  lcd.setCursor(0,3);

  lcd.print("Take Rest");

PositionState="Wrong Seating Position";

  Serial.println("Wrong Seating Position");

  Serial.println("Take Rest");

   digitalWrite(Buzzer, LOW);

    ///digitalWrite(SmallVibrator, HIGH);

 delay(2000);

}

else {

  lcd.clear();

  lcd.setCursor(0,0);

  lcd.print("Seet Monitor");

  digitalWrite(Buzzer, HIGH);

  ///digitalWrite(SmallVibrator, LOW);

  lcd.setCursor(0, 1);

  lcd.print("Position:");

  lcd.setCursor(10, 2);

  lcd.print("Normal");

  PositionState= ("Normal");

  delay(3000);

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

WiFiClient client = server.available();   // Listen for incoming clients

if (client) {                             // If a new client connects,

    Serial.println("New Client.");          // print a message out in the serial port

    String currentLine = "";                // make a String to hold incoming data from the client

    currentTime = millis();

    previousTime = currentTime;

    while (client.connected() && currentTime - previousTime <= timeoutTime) { // loop while the client's connected

      currentTime = millis();

      if (client.available()) {             // if there's bytes to read from the client,

        char c = client.read();             // read a byte, then

        Serial.write(c);                    // print it out the serial monitor

        header += c;

        if (c == '\n') {                    // if the byte is a newline character

          // if the current line is blank, you got two newline characters in a row.

          // that's the end of the client HTTP request, so send a response:

          if (currentLine.length() == 0) {

            // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)

            // and a content-type so the client knows what's coming, then a blank line:

            client.println("HTTP/1.1 200 OK");

            client.println("Content-type:text/html");

            client.println("Connection: close");

            client.println();

            if (header.indexOf("GET /1/on") >= 0) {

              Serial.println("Relay1 On");

              Relay1State = "on";

              digitalWrite(Relay1, LOW);

            } else if (header.indexOf("GET /1/off") >= 0) {

              Serial.println("Relay1 Off");

              Relay1State = "off";

              digitalWrite(Relay1, HIGH);

            }

            if (header.indexOf("GET /2/on") >= 0) {

              Serial.println("Relay2 On");

              Relay2State = "on";

              digitalWrite(Relay2, LOW);

            } else if (header.indexOf("GET /2/off") >= 0) {

              Serial.println("Relay2 Off");

              Relay2State = "off";

              digitalWrite(Relay2, HIGH);

            }

            if (header.indexOf("GET /3/on") >= 0) {

              Serial.println("Relay3 On");

              Relay3State = "on";

              digitalWrite(Relay3, LOW);

            } else if (header.indexOf("GET /3/off") >= 0) {

              Serial.println("Relay3 Off");

              Relay3State = "off";

              digitalWrite(Relay3, HIGH);

            }

            if (header.indexOf("GET /4/on") >= 0) {

              Serial.println("Relay4 On");

              Relay4State = "on";

              digitalWrite(Relay4, LOW);

            } else if (header.indexOf("GET /4/off") >= 0) {

              Serial.println("Relay4 Off");

              Relay4State = "off";

              digitalWrite(Relay4, HIGH);

            }

////////////////////////////////////////////////////////////////////////////////////////

            if (header.indexOf("GET /5/on") >= 0) {

              Serial.println("Relay5 On");

              Relay5State = "on";

              digitalWrite(Relay5, LOW);

              delay(2000);

              digitalWrite(Buzzer, HIGH);

              delay(2000);

              digitalWrite(Relay6, LOW);

              delay(2000);

              digitalWrite(Relay5, HIGH);

              delay(2000);

              digitalWrite(Relay6, LOW);

              delay(2000);

              digitalWrite(Relay5, HIGH);

            } else if (header.indexOf("GET /5/off") >= 0) {

              Serial.println("Relay5 Off");

              Relay5State = "off";

              digitalWrite(Relay5, HIGH);

            }

            if (header.indexOf("GET /6/on") >= 0) {

              Serial.println("Relay6 On");

              Relay6State = "on";

              digitalWrite(Relay6, LOW);

              //lcd.clear();

              //lcd.setCursor(0,0);

              //lcd.print("Person Back Normal");

             // delay(10000);

              //Relay6State = "off";

              //digitalWrite(Relay6, HIGH);

            } else if (header.indexOf("GET /6/off") >= 0) {

              Serial.println("Relay6 Off");

              Relay6State = "off";

              digitalWrite(Relay6, HIGH);

            }

////////////////////////////////////////////////////////////////////////////////////////////////

            if (header.indexOf("GET /7/on") >= 0) {

              Serial.println("Relay7 On");

              Relay7State = "on";

              digitalWrite(Relay7, LOW);

            } else if (header.indexOf("GET /7/off") >= 0) {

              Serial.println("Relay7 Off");

              Relay7State = "off";

              digitalWrite(Relay7, HIGH);

            }

            if (header.indexOf("GET /8/on") >= 0) {

              Serial.println("Relay8 On");

              Relay8State = "on";

              digitalWrite(Relay8, LOW);

            } else if (header.indexOf("GET /8/off") >= 0) {

              Serial.println("Relay8 Off");

              Relay8State = "off";

              digitalWrite(Relay8, HIGH);

            }

          if (header.indexOf("GET /9/on") >= 0) {

              Serial.println("Relay9 On");

              Relay9State = "on";

              digitalWrite(Relay9, LOW);

            } else if (header.indexOf("GET /9/off") >= 0) {

              Serial.println("Relay9 Off");

              Relay9State = "off";

              digitalWrite(Relay9, HIGH);

            }

      if (header.indexOf("GET /10/on") >= 0) {

              Serial.println("Relay10 On");

              Relay10State = "on";

              digitalWrite(Relay10, LOW);

            } else if (header.indexOf("GET /10/off") >= 0) {

              Serial.println("Relay10 Off");

              Relay10State = "off";

              digitalWrite(Relay10, HIGH);

            }

// Display the HTML web page

            client.println("<!DOCTYPE html><html>");

            client.println("<head><meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");

            //client.println("<meta http-equiv=\"refresh\" content=\"5\">");  // Refresh page every 1 second

            client.println("<link rel=\"icon\" href=\"data:,\">");

            // CSS to style the on/off buttons

            // Feel free to change the background-color and font-size attributes to fit your preferences

            client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px auto; text-align: center;}");

            client.println(".button { background-color: #195B6A; border: none; color: white; padding: 16px 40px;");

            client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;}");

            client.println(".button2 {background-color: #77878A;}</style></head>");

            // Web Page Heading

              //client.println("<body><h1>Seet Monitor</h1>");

              client.println("<body><h1>Seet Monitor</h1><br><br>");

              //client.println();///space

             // client.println();///space

//////////////////////////////////////////////////////////////////////////////////////////////////////////

client.println ("<table border='0' cellpadding='2' cellspacing='10'>");

      client.println ("<tr>");// Used as space between list

    //  String TTes = connect();

        client.println ("<td>KG : ");

          client.println(Weght);

        client.println ("</td>");

       client.println ("<tr>");// Used as space between list

       client.println ("<td>Person:");

        client.println(PositionState2);

        client.println("</td>");

      client.println ("</tr>"); // used as end of first space

        client.println ("<td>Position:");

        client.println(PositionState);

        client.println("</td>");

      client.println ("</tr>"); // used as end of first space

      client.println ("<tr style='text-align:center;'>");

            if (Relay1State=="off") {

              client.println("<td>Leg Up: "+Relay1State +" <br><a href=\"/1/on\"><button class=\"button\">ON</button></a></td>");

            } else {

              client.println("<td> Leg Down: "+Relay1State +"<br><a href=\"/1/off\"><button class=\"button button2\">OFF</button></a></td>");

            }

            if (Relay2State=="off") {

              client.println("<td>Leg Down: "+Relay2State +" <br><a href=\"/2/on\"><button class=\"button\">ON</button></a></td>");

            } else {

              client.println("<td> Leg Up: "+Relay2State +"<br><a href=\"/2/off\"><button class=\"button button2\">OFF</button></a></td>");

            }

            client.println ("<tr>"); // used as separtion space between buttones to be despaled as two of them

              if (Relay3State=="off") {

              client.println("<td>Back Down: "+Relay3State +" <br><a href=\"/3/on\"><button class=\"button\">ON</button></a></td>");

            } else {

              client.println("<td> Back Up: "+Relay3State +"<br><a href=\"/3/off\"><button class=\"button button2\">OFF</button></a></td>");

            }

              if (Relay4State=="off") {

              client.println("<td>Back Up: "+Relay4State +" <br><a href=\"/4/on\"><button class=\"button\">ON</button></a></td>");

            } else {

              client.println("<td> Back Down: "+Relay4State +"<br><a href=\"/4/off\"><button class=\"button button2\">OFF</button></a></td>");

            }

            client.println ("</tr>");// used as end of the second space

              if (Relay5State=="off") {

              client.println("<td>Stretching On: "+Relay5State +" <br><a href=\"/5/on\"><button class=\"button\">ON</button></a></td>");

            } else {

              client.println("<td> Stretching Off: "+Relay5State +"<br><a href=\"/5/off\"><button class=\"button button2\">OFF</button></a></td>");

            }

             if (Relay6State=="off") {

              client.println("<td> Stretching Off: "+Relay6State +" <br><a href=\"/6/on\"><button class=\"button\">ON</button></a></td>");

            } else {

              client.println("<td> Stretching On: "+Relay6State +"<br><a href=\"/6/off\"><button class=\"button button2\">OFF</button></a></td>");

            }

            client.println ("</tr>");// used as end of the second space

              if (Relay7State=="off") {

              client.println("<td>Leg Massag On: "+Relay7State +" <br><a href=\"/7/on\"><button class=\"button\">ON</button></a></td>");

            } else {

              client.println("<td> Leg Massag Off: "+Relay7State +"<br><a href=\"/7/off\"><button class=\"button button2\">OFF</button></a></td>");

            }

              if (Relay8State=="off") {

              client.println("<td>Heater  On: "+Relay8State +" <br><a href=\"/8/on\"><button class=\"button\">ON</button></a></td>");

            } else {

              client.println("<td> Heater Off: "+Relay8State +"<br><a href=\"/8/off\"><button class=\"button button2\">OFF</button></a></td>");

            }

            client.println ("</tr>");// used as end of the second space

            if (Relay9State=="off") {

              client.println("<td>Vibrator On: "+Relay9State +" <br><a href=\"/9/on\"><button class=\"button\">ON</button></a></td>");

            } else {

              client.println("<td> Vibrator Off: "+Relay9State +"<br><a href=\"/9/off\"><button class=\"button button2\">OFF</button></a></td>");

            }

             client.println ("</tr>");// used as end of the second space

            if (Relay10State=="off") {

              client.println("<td>Sitting Time : "+Relay10State +" <br><a href=\"/10/on\"><button class=\"button\">ON</button></a></td>");

            } else {

              client.println("<td> Sitting Time  : "+Relay10State +"<br><a href=\"/10/off\"><button class=\"button button2\">OFF</button></a></td>");

            }

          client.println ("</tr>");// used as end of the second space

        //client.println ("<td>Relay1 : On <br>Butt</td>");

        //client.println ("<td>Relay2 : On <br>Butt</td>");

      client.println ("</tr>  ");

    client.println ("</table>");

 ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

            client.println("</body></html>");

            // The HTTP response ends with another blank line

            client.println();

            // Break out of the while loop

            break;

          } else { // if you got a newline, then clear currentLine

            currentLine = "";

          }

        } else if (c != '\r') {  // if you got anything else but a carriage return character,

          currentLine += c;      // add it to the end of the currentLine

        }

      }

    }

    // Clear the header variable

    header = "";

    // Close the connection

    ///client.stop();

    /////Serial.println("Client disconnected.");

    /////Serial.println("");

  }

unsigned long currentMillis = millis();

  // if WiFi is down, try reconnecting every CHECK\_WIFI\_TIME seconds

  if ((WiFi.status() != WL\_CONNECTED) && (currentMillis - previousMillis >=interval)) {

    Serial.print(millis());

    Serial.println("Reconnecting to WiFi...");

    WiFi.disconnect();

    WiFi.reconnect();

    previousMillis = currentMillis;

  }

}// End of Loop