**// The RANDoms :   
// Domain : IoT and Android  
// Title : Relentless and Realtime Smart Worker Suit and App**

#include "WiFi.h"

#include "ESPAsyncWebServer.h"

#include <Wire.h>

#include <OneWire.h>

#include <DallasTemperature.h>

#include <Adafruit\_Sensor.h>

#include <DHT.h>

#include <ESPmDNS.h>

const int trigPin = 5;

const int echoPin = 18;

#define SOUND\_SPEED 0.034

long duration;

float distanceCm;

// Replace with your network credentials

const char\* ssid = "Redmi";

const char\* password = "12345678";

#define DHTTYPE DHT11

#define DHTPIN 27

#define ONE\_WIRE\_BUS 4

OneWire oneWire(ONE\_WIRE\_BUS);

DallasTemperature sensors(&oneWire);

String temperatureF = "";

String temperatureC = "";

unsigned long lastTime = 0;

unsigned long timerDelay = 30000;

DHT dht(DHTPIN, DHTTYPE);

// Create AsyncWebServer object on port 80

AsyncWebServer server(80);

String readDHTTemperature() {

float t = dht.readTemperature();

if (isnan(t)) {

Serial.println("Failed to read from DHT sensor!");

return "--";

}

else {

Serial.println(t);

return String(t);

}

}

String readDHTHumidity() {

float h = dht.readHumidity();

if (isnan(h)) {

Serial.println("Failed to read from DHT sensor!");

return "--";

}

else {

Serial.println(h);

return String(h);

}

}

String readDSTemperatureC() {

// Call sensors.requestTemperatures() to issue a global temperature and Requests to all devices on the bus

sensors.requestTemperatures();

float tempC = sensors.getTempCByIndex(0);

if(tempC == -127.00) {

Serial.println("Failed to read from DS18B20 sensor");

return "--";

} else {

Serial.print("Temperature Celsius: ");

Serial.println(tempC);

}

return String(tempC);

}

String readDSTemperatureF() {

// Call sensors.requestTemperatures() to issue a global temperature and Requests to all devices on the bus

sensors.requestTemperatures();

float tempF = sensors.getTempFByIndex(0);

if(int(tempF) == -196){

Serial.println("Failed to read from DS18B20 sensor");

return "--";

} else {

Serial.print("Temperature Fahrenheit: ");

Serial.println(tempF);

}

return String(tempF);

}

String readDistance(){

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

distanceCm = duration \* SOUND\_SPEED/2;

Serial.println("Distance:");

Serial.println(distanceCm);

return String(distanceCm);

}

const char index\_html[] PROGMEM = R"rawliteral(

<!DOCTYPE HTML><html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.7.2/css/all.css" integrity="sha384-fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr" crossorigin="anonymous">

<style>

html {

font-family: Arial;

display: inline-block;

margin: 0px auto;

text-align: center;

}

h2 { font-size: 3.0rem; }

p { font-size: 3.0rem; }

.units { font-size: 1.2rem; }

.dht-labels{

font-size: 1.5rem;

vertical-align:middle;

padding-bottom: 15px;

}

</style>

</head>

<body>

<h2>WORKER HEALTH ANALYSIS</h2>

<p>

<i class="fas fa-thermometer-half" style="color:#059e8a;"></i>

<span class="dht-labels">Temperature</span>

<span id="temperature">%TEMPERATURE%</span>

<sup class="units">&deg;C</sup>

</p>

<p>

<i class="fas fa-tint" style="color:#00add6;"></i>

<span class="dht-labels">Humidity</span>

<span id="humidity">%HUMIDITY%</span>

<sup class="units">&percnt;</sup>

</p>

<p>

<i class="fas fa-thermometer-half" style="color:#059e8a;"></i>

<span class="ds-labels">Temperature Celsius</span>

<span id="temperaturec">%TEMPERATUREC%</span>

<sup class="units">&deg;C</sup>

</p>

<p>

<i class="fas fa-thermometer-half" style="color:#059e8a;"></i>

<span class="ds-labels">Temperature Fahrenheit</span>

<span id="temperaturef">%TEMPERATUREF%</span>

<sup class="units">&deg;F</sup>

</p>

<p>

<i class="fa fa-road" style="color:#00add6;"></i>

<span class="dht-labels">Distance</span>

<span id="Distance">%DISTANCE%</span>

<sup class="units">Cm</sup>

</p>

</body>

<script>

setInterval(function ( ) {

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

document.getElementById("temperature").innerHTML = this.responseText;

}

};

xhttp.open("GET", "/temperature", true);

xhttp.send();

}, 10000 ) ;

setInterval(function ( ) {

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

document.getElementById("humidity").innerHTML = this.responseText;

}

};

xhttp.open("GET", "/humidity", true);

xhttp.send();

}, 10000 ) ;

setInterval(function ( ) {

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

document.getElementById("temperaturec").innerHTML = this.responseText;

}

};

xhttp.open("GET", "/temperaturec", true);

xhttp.send();

}, 10000) ;

setInterval(function ( ) {

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

document.getElementById("temperaturef").innerHTML = this.responseText;

}

};

xhttp.open("GET", "/temperaturef", true);

xhttp.send();

}, 10000) ;

</script>

</html>)rawliteral";

// Replaces placeholder with DHT values

String processor(const String& var){

//Serial.println(var);

if(var == "TEMPERATURE"){

return readDHTTemperature();

}

else if(var == "HUMIDITY"){

return readDHTHumidity();

}

else if(var == "TEMPERATUREC"){

return temperatureC;

}

else if(var == "TEMPERATUREF"){

return temperatureF;

}

else if(var == "DISTANCE"){

return readDistance();

}

return String();

}

void setup(){

// Serial port for debugging purposes

Serial.begin(115200);

dht.begin();

sensors.begin();

temperatureC = readDSTemperatureC();

temperatureF = readDSTemperatureF();

pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output

pinMode(echoPin, INPUT);

// Connect to Wi-Fi

WiFi.begin(ssid, password);

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

delay(1000);

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

}

// Print ESP32 Local IP Address

Serial.println(WiFi.localIP());

if(!MDNS.begin("esp32")) {

Serial.println("Error starting mDNS");

return;

}

Serial.println(WiFi.localIP());

// Route for root / web page

server.on("/", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send\_P(200, "text/html", index\_html, processor);

});

server.on("/temperature", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send\_P(200, "text/plain", readDHTTemperature().c\_str());

});

server.on("/humidity", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send\_P(200, "text/plain", readDHTHumidity().c\_str());

});

server.on("/temperaturec", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send\_P(200, "text/plain", temperatureC.c\_str());

});

server.on("/temperaturef", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send\_P(200, "text/plain", temperatureF.c\_str());

});

server.on("/distanceCm", HTTP\_GET, [](AsyncWebServerRequest \*request){

request->send\_P(200, "text/plain", readDistance().c\_str());

});

// Start server

server.begin();

}

void loop(){

if ((millis() - lastTime) > timerDelay) {

temperatureC = readDSTemperatureC();

temperatureF = readDSTemperatureF();

lastTime = millis();

}

}