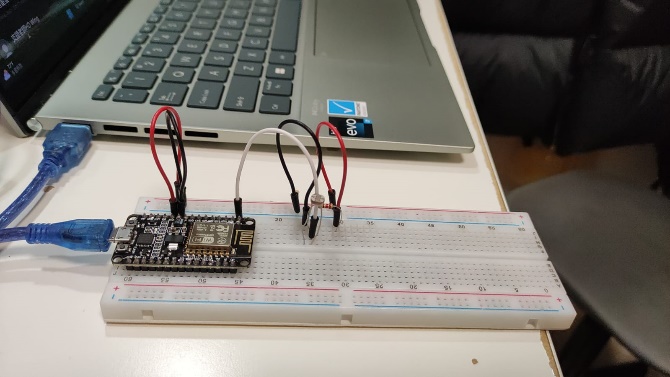
Task:

a) Design a voltage divider to fulfil below requirement. (Total: 15%)

i. Use a resistor and LDR only.



#include <Arduino.h>

void setup() {

pinMode(A0,INPUT);

Serial.begin(115200);

}

void loop() {

Serial.print("LDR: ");

Serial.print(analogRead(A0));

Serial.print('\n');

delay(100);

}

ii. Draw the circuit diagram and label the name and value. (5%)

A white piece of paper with writing on it

Description automatically generated with low confidence

iii. What is the ADC resolution of ESP8266? What is the range of ADC value? Show the calculation step. (5%)

Text, letter

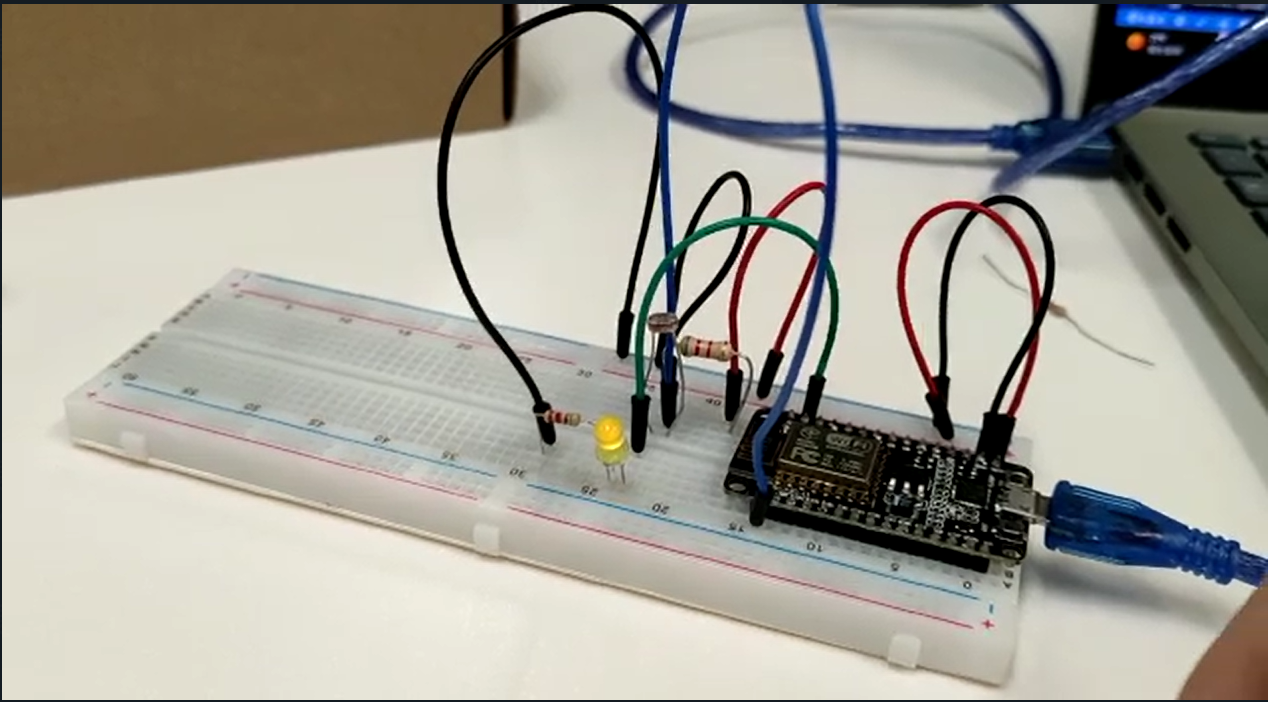
Description automatically generated

iv. What is the value of voltage if the ADC value is 50. Show the calculation step. (5%)

A piece of paper with writing on it

Description automatically generated

b) Build a smart lighting system and fulfil below requirement. (Total: 25%)

i. Connect the light sensor and LED to ESP8266. Draw the circuit diagram and label the name, value and pin name. Then, take a clear photo. (No capture or screenshot) (5%)Diagram, schematic

Description automatically generated

ii. Code a program to print the value of brightness by percentage on serial monitor. (Recommend to use “hterm”) The format is “Brightness: xx%”. Capture the output and source code. (10%)

A screenshot of a computer

Description automatically generated

#include <Arduino.h>

int x = 450;

int y = 1023;

int temp =0;

float brightness = 0;

int temp2 =0;

void setup() {

pinMode(A0,INPUT);

pinMode(D1,OUTPUT);

Serial.begin(115200);

}

void loop() {

temp = analogRead(A0);

temp2 = map(temp, x, y, 0, 255);

analogWrite(D1,temp2);

brightness = (100-temp2\*100/255);

Serial.print("Brightness:  ");

Serial.print(brightness);

Serial.print("%");

Serial.print('\n');

delay(100);

}

iii. Control the brightness of LED based on the environment. When the brightness is lower than 50%, turn on the LED. Otherwise, turn off. Capture the source code. (10%)

#include <Arduino.h>

int x = 450;

int y = 1023;

int temp =0;

float brightness = 0;

int temp2 =0;

void setup() {

pinMode(A0,INPUT);

pinMode(D1,OUTPUT);

Serial.begin(115200);

}

void loop() {

temp = analogRead(A0);

temp2 = map(temp, x, y, 0, 255);

brightness = (100-temp2\*100/255);

if(brightness<50)

  {

    digitalWrite(D1,HIGH);//The LED turns ON in Dark.

  }

  else

  {

    digitalWrite(D1,LOW);//The LED turns OFF in Light.

  }

Serial.print("Brightness:  ");

Serial.print(brightness);

Serial.print("%");

Serial.print('\n');

delay(100);

}

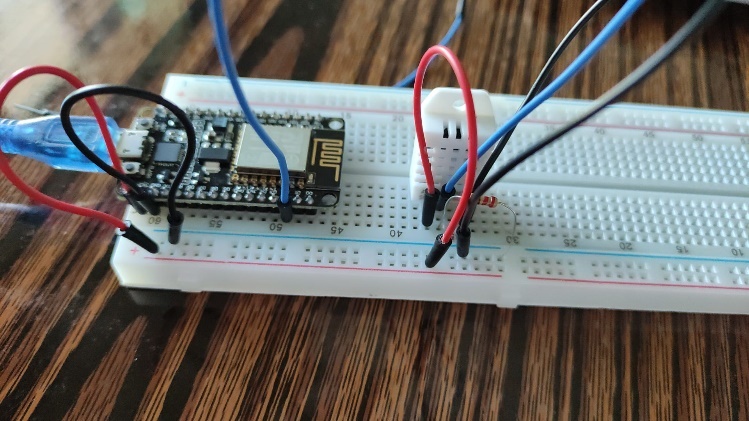
c) Code a program to read the temperature and humidity by DHT22 sensor.

(Total: 15%)

i. Draw the circuit diagram and label the name, value and pin number. (5%)A piece of paper with writing on it

Description automatically generated with medium confidence

ii. Print the value of temperature and humidity to serial monitor. The format is “Temp: xxx°C, Humi: yyy%”. Capture the output and source code. (10%)



Shape, rectangle

Description automatically generated

#include <Adafruit\_Sensor.h>

#include <DHT.h>

#include <DHT\_U.h>

#define DHTPIN 4     // Digital pin connected to the DHT sensor

// Feather HUZZAH ESP8266 note: use pins 3, 4, 5, 12, 13 or 14 --

// Pin 15 can work but DHT must be disconnected during program upload.

// Uncomment the type of sensor in use:

//#define DHTTYPE    DHT11     // DHT 11

#define DHTTYPE    DHT22     // DHT 22 (AM2302)

//#define DHTTYPE    DHT21     // DHT 21 (AM2301)

// See guide for details on sensor wiring and usage:

//   https://learn.adafruit.com/dht/overview

DHT\_Unified dht(DHTPIN, DHTTYPE);

uint32\_t delayMS;

void setup() {

  Serial.begin(9600);

  // Initialize device.

  dht.begin();

  Serial.println(F("DHTxx Unified Sensor Example"));

  // Print temperature sensor details.

  sensor\_t sensor;

  dht.temperature().getSensor(&sensor);

  Serial.println(F("------------------------------------"));

  Serial.println(F("Temperature Sensor"));

  Serial.print  (F("Sensor Type: ")); Serial.println(sensor.name);

  Serial.print  (F("Driver Ver:  ")); Serial.println(sensor.version);

  Serial.print  (F("Unique ID:   ")); Serial.println(sensor.sensor\_id);

  Serial.print  (F("Max Value:   ")); Serial.print(sensor.max\_value); Serial.println(F("°C"));

  Serial.print  (F("Min Value:   ")); Serial.print(sensor.min\_value); Serial.println(F("°C"));

  Serial.print  (F("Resolution:  ")); Serial.print(sensor.resolution); Serial.println(F("°C"));

  Serial.println(F("------------------------------------"));

  // Print humidity sensor details.

  dht.humidity().getSensor(&sensor);

  Serial.println(F("Humidity Sensor"));

  Serial.print  (F("Sensor Type: ")); Serial.println(sensor.name);

  Serial.print  (F("Driver Ver:  ")); Serial.println(sensor.version);

  Serial.print  (F("Unique ID:   ")); Serial.println(sensor.sensor\_id);

  Serial.print  (F("Max Value:   ")); Serial.print(sensor.max\_value); Serial.println(F("%"));

  Serial.print  (F("Min Value:   ")); Serial.print(sensor.min\_value); Serial.println(F("%"));

  Serial.print  (F("Resolution:  ")); Serial.print(sensor.resolution); Serial.println(F("%"));

  Serial.println(F("------------------------------------"));

  // Set delay between sensor readings based on sensor details.

  delayMS = sensor.min\_delay / 1000;

}

void loop() {

  // Delay between measurements.

  delay(delayMS);

  // Get temperature event and print its value.

  sensors\_event\_t event;

  dht.temperature().getEvent(&event);

  if (isnan(event.temperature)) {

    Serial.println(F("Error reading temperature!"));

  }

  else {

    Serial.print(F("Temp: "));

    Serial.print(event.temperature);

    Serial.print(F("°C"));

    Serial.print(", ");

  }

  // Get humidity event and print its value.

  dht.humidity().getEvent(&event);

  if (isnan(event.relative\_humidity)) {

    Serial.println(F("Error reading humidity!"));

  }

  else {

    Serial.print(F("Humi: "));

    Serial.print(event.relative\_humidity);

    Serial.println(F("%"));

  }

}

d) Create a web server on ESP8266. It includes below functions. (Total: 45%)

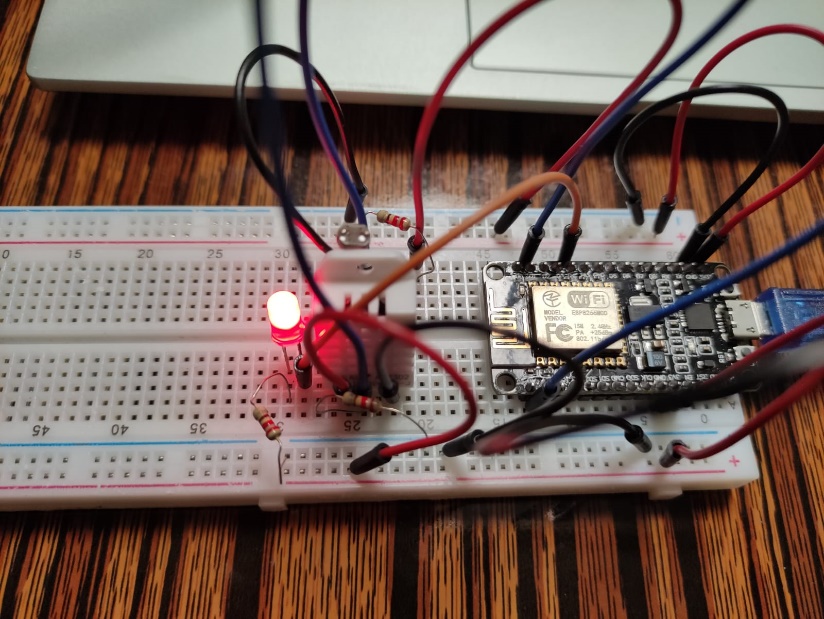
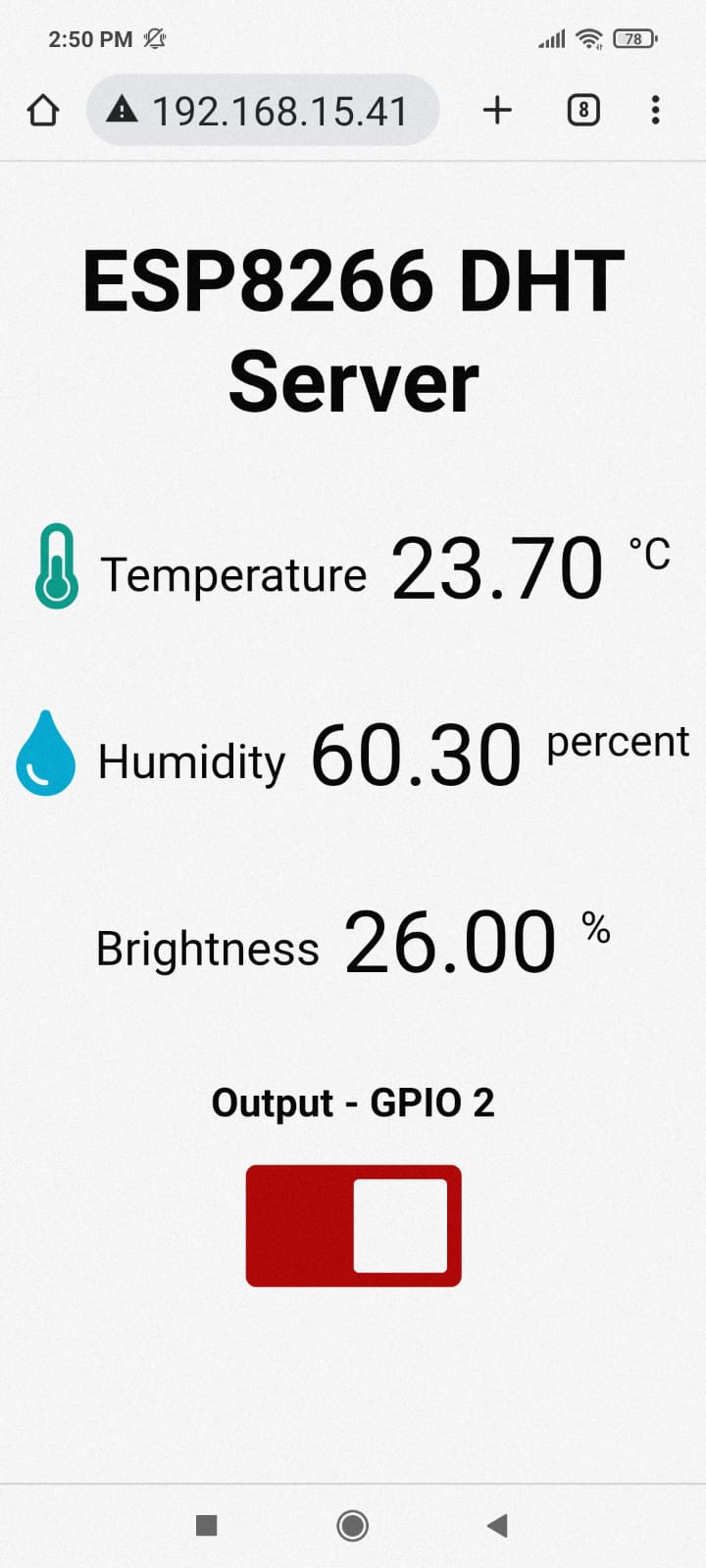
i. Show the real time temperature and humidity. (15%)

ii. Show the real time brightness value. (10%)

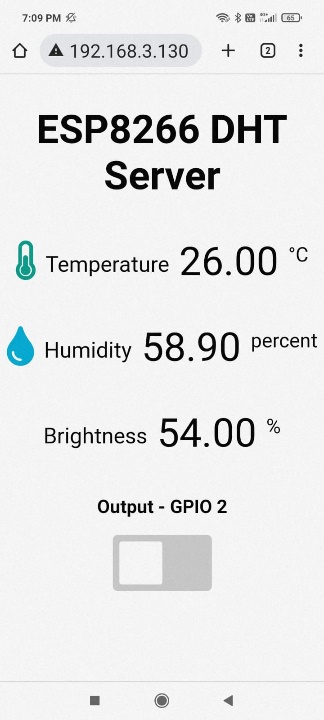
iii. Have a switch to control the LED. (10%)

iv. Good coding style. (5%)

v. Nice outlook of website. (5%)



When light turns on



A picture containing text

Description automatically generated

When light turns off

// Import required libraries

#include <Arduino.h>

#include <ESP8266WiFi.h>

#include <Hash.h>

#include <ESPAsyncTCP.h>

#include <ESPAsyncWebServer.h>

#include <Adafruit\_Sensor.h>

#include <DHT.h>

int x = 450;

int y = 1023;

int temp =0;

const char\* PARAM\_INPUT\_1 = "output";

const char\* PARAM\_INPUT\_2 = "state";

// Replace with your network credentials

const char\* ssid = "noddy wifi";

const char\* password = "55434050";

#define DHTPIN 5     // Digital pin connected to the DHT sensor

// Uncomment the type of sensor in use:

//#define DHTTYPE    DHT11     // DHT 11

#define DHTTYPE    DHT22     // DHT 22 (AM2302)

//#define DHTTYPE    DHT21     // DHT 21 (AM2301)

DHT dht(DHTPIN, DHTTYPE);

// current temperature & humidity, updated in loop()

float t = 0.0;

float h = 0.0;

float b = 0.0;

// Create AsyncWebServer object on port 80

AsyncWebServer server(80);

// Generally, you should use "unsigned long" for variables that hold time

// The value will quickly become too large for an int to store

unsigned long previousMillis = 0;    // will store last time DHT was updated

// Updates DHT readings every 10 seconds

const long interval = 10000;

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;

    }

    .switch {position: relative; display: inline-block; width: 120px; height: 68px}

    .switch input {display: none}

    .slider {position: absolute; top: 0; left: 0; right: 0; bottom: 0; background-color: #ccc; border-radius: 6px}

    .slider:before {position: absolute; content: ""; height: 52px; width: 52px; left: 8px; bottom: 8px; background-color: #fff; -webkit-transition: .4s; transition: .4s; border-radius: 3px}

    input:checked+.slider {background-color: #b30000}

    input:checked+.slider:before {-webkit-transform: translateX(52px); -ms-transform: translateX(52px); transform: translateX(52px)}

  </style>

</head>

<body>

  <h2>ESP8266 DHT Server</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">percent</sup>

  </p>

  <p>

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

    <span id="brightness">%BRIGHTNESS%</span>

    <sup class="units">%</sup>

  </p>

  <h4>Output - GPIO 2</h4>

  <label class="switch"><input type="checkbox" onchange="toggleCheckbox(this)" id="2" " + outputState(2) + "><span class="slider"></span></label>

</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("brightness").innerHTML = this.responseText;

    }

  };

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

  xhttp.send();

}, 10000 ) ;

function toggleCheckbox(element) {

  var xhr = new XMLHttpRequest();

  if(element.checked){

    xhr.open("GET", "/update?output="+element.id+"&state=1", true);

  }

  else {

    xhr.open("GET", "/update?output="+element.id+"&state=0", true);

  }

  xhr.send();

}

</script>

</html>)rawliteral";

String outputState(int output){

  if(digitalRead(output)){

    return "checked";

  }

  else {

    return "";

  }

}

String processor(const String& var){

  //Serial.println(var);

  if(var == "TEMPERATURE"){

    return String(t);

  }

  if(var == "HUMIDITY"){

    return String(h);

  }

  if(var == "BRIGHTNESS"){

    return String(b);

  }

  if(var == "BUTTONPLACEHOLDER"){

    String buttons = "";

    buttons += "<h4>Output - GPIO 5</h4><label class=\"switch\"><input type=\"checkbox\" onchange=\"toggleCheckbox(this)\" id=\"5\" " + outputState(2) + "><span class=\"slider\"></span></label>";

    return buttons;

  }

  return String();

}

void setup(){

  // Serial port for debugging purposes

  Serial.begin(115200);

  dht.begin();

  pinMode(2, OUTPUT);

  digitalWrite(2, LOW);

  pinMode(A0,INPUT);

  // Connect to Wi-Fi

  WiFi.begin(ssid, password);

  Serial.println("Connecting to WiFi");

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

    delay(1000);

    Serial.println(".");

  }

  // Print ESP8266 Local IP Address

  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", String(t).c\_str());

  });

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

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

  });

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

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

  });

// Send a GET request to <ESP\_IP>/update?output=<inputMessage1>&state=<inputMessage2>

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

    String inputMessage1;

    String inputMessage2;

    // GET input1 value on <ESP\_IP>/update?output=<inputMessage1>&state=<inputMessage2>

    if (request->hasParam(PARAM\_INPUT\_1) && request->hasParam(PARAM\_INPUT\_2)) {

      inputMessage1 = request->getParam(PARAM\_INPUT\_1)->value();

      inputMessage2 = request->getParam(PARAM\_INPUT\_2)->value();

      digitalWrite(inputMessage1.toInt(), inputMessage2.toInt());

    }

    else {

      inputMessage1 = "No message sent";

      inputMessage2 = "No message sent";

    }

    Serial.print("GPIO: ");

    Serial.print(inputMessage1);

    Serial.print(" - Set to: ");

    Serial.println(inputMessage2);

    request->send(200, "text/plain", "OK");

  });

  // Start server

  server.begin();

}

void loop(){

  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval) {

    // save the last time you updated the DHT values

    previousMillis = currentMillis;

    // Read temperature as Celsius (the default)

    float newT = dht.readTemperature();

    // Read temperature as Fahrenheit (isFahrenheit = true)

    //float newT = dht.readTemperature(true);

    // if temperature read failed, don't change t value

    if (isnan(newT)) {

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

    }

    else {

      t = newT;

      Serial.println(t);

    }

    // Read Humidity

    float newH = dht.readHumidity();

    // if humidity read failed, don't change h value

    if (isnan(newH)) {

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

    }

    else {

      h = newH;

      Serial.println(h);

    }

    float newB;

    // Read brightness

    temp = analogRead(A0);

    newB = map(temp, x, y, 0, 100);

    newB = (100 - newB);

    // if LDR read failed, don't change b value

    if (isnan(newB)) {

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

    }

    else {

      b = newB;

      Serial.println(b);

    }

  }

}