

ESP32: Mesure de temperature

Liens Web

[ESP32 : DHT11/22](#)

[ESP32 : ServeurWeb1 DHT11/22](#)

[ESP32 : ServeurWeb2 DHT11/22](#)

[ESP32 : Objet connecté DGT11/22](#)

[ESP32 PicoKit : DHT22](#)

[ESP32 : Librairie DHT22](#)

[DHT11/22](#)

[ARduino : DHT11/22](#)

Test en réel sur carte ESP32

Matériel

Branchement

Programmes

[ESP32_Temperature.ino](#)

```
#include <DHT.h>

// Replace with your network credentials
const char* ssid = "Livebox-5ac0";
const char* password = "chanterie19282008";

#define DHTPIN 13    // 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);

// Create AsyncWebServer object on port 80
```

```
AsyncWebServer server(80);

String readDHTTemperature() {
    // Sensor readings may also be up to 2 seconds 'old' (its a very slow
    // sensor)
    // Read temperature as Celsius (the default)
    float t = dht.readTemperature();
    // Read temperature as Fahrenheit (isFahrenheit = true)
    //float t = dht.readTemperature(true);
    // Check if any reads failed and exit early (to try again).
    if (isnan(t)) {
        Serial.println("Failed to read from DHT sensor!");
        return "--";
    }
    else {
        Serial.println(t);
        return String(t);
    }
}

String readDHTHumidity() {
    // Sensor readings may also be up to 2 seconds 'old' (its a very slow
    // sensor)
    float h = dht.readHumidity();
    if (isnan(h)) {
        Serial.println("Failed to read from DHT sensor!");
        return "--";
    }
    else {
        Serial.println(h);
        return String(h);
    }
}

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-
fnm0CqbTlWIlj8LyTjo7m0UStjsKC4p0pQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr"
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>ESP32 Serveur DHT 22</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">Humidite</span>
  <span id="humidity">%HUMIDITY%</span>
  <sup class="units">&percnt;</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 ) ;
</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();
  }
  return String();
}

void setup(){
  // Serial port for debugging purposes
  Serial.begin(115200);

  dht.begin();

  // 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());

  // 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());
  });

  // Start server
  server.begin();
}

void loop(){

}
```

Broches ESP32

ESP32 TTGO LORA 32 V2.0

