READ_ME IT ASSESSMENT-James Bollard

General Description

An ESP32 Arduino-based project for a temperature and humidity web server is a practical application of IoT (Internet of Things) technology. It involves using an ESP32 microcontroller, a temperature and humidity sensor (such as the DHT11 or DHT22), and the Arduino framework to create a web server that provides real-time temperature and humidity data to users through a web interface. Here's a general description of such a project:

The project utilises an ESP32 development board, which is a powerful microcontroller with built-in Wi-Fi capabilities. It is connected to a DHT sensor (e.g., DHT11 or DHT22), which is responsible for measuring temperature and humidity. The project is programmed using the Arduino framework, which simplifies the coding process for the ESP32. The Arduino IDE (Integrated Development Environment) or other compatible development environments (like Visual Studio Code with PlatformIO) can be used. The ESP32 reads temperature and humidity data from the DHT sensor at regular intervals. It then hosts a web server on its Wi-Fi network, which allows users to access a web page. This web page dynamically displays the current temperature and humidity data.

Key Features

1. **Real-Time Data:** The web server provides real-time temperature and humidity data, making it easy for users to monitor their environment.

2. **Web Interface:** Users can access the data via a user-friendly web interface using a web browser on a computer, tablet, or smartphone. The web page typically includes simple HTML and JavaScript elements to display the data.

3. **Remote Monitoring:** Users can access the data from anywhere with an internet connection, allowing for remote monitoring and control of the environment.

4. **Historical Data:** Some implementations may store historical data, allowing users to review past temperature and humidity trends.

5. **Alerts and Notifications:** In more advanced versions, the web server can be programmed to send alerts or notifications to users when temperature or humidity levels go beyond specified thresholds.

Code examples

Example of code

```
 <!-- <p>Time: <span id="time2"><%= obj.updateTime %></span><br /></p> -->
   <p>Temperature: <%= obj.temperature %>&deg;<br /></p>
   <p>Humidity: <%= obj.humidity %>&percnt;<br /></p>
  </li>
  <li>
```

This code is in the HTML and it is the part that represents the temperature and humidity and the obj's represent the number from the dht11 coming in and replacing that so it shows you.

```
if (isnan(humidity) || isnan(temperature)) {
  Serial.println("Error: Failed to read valid data from DHT11 sensor.");
 } else {
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.print("%, Temperature: ");
  Serial.print(temperature);
  Serial.println("°C");
```

This code is to show if the dht11 sensor is not giving us the correct temp and humidity or if there was an error it will tell us in the output terminal and the rest is to put the numbers into celsius and a percentage.

Framework used

For me to create a project that involves an ESP32, temperature, and humidity monitoring, along with GitHub for version control, PowerShell for scripting and automation, Arduino for ESP32 programming, and Visual Studio Code for code editing and development, you can use the following technologies and tools:

1. **ESP32**: The ESP32 microcontroller is at the core of your project. You'll use it to read data from the temperature and humidity sensor and create a web server to serve this data.

2. **DHT Sensor Library**: For reading temperature and humidity data, you can use the DHT sensor library. This library is compatible with various DHT sensor models, including DHT11 and DHT22.

3. **Arduino IDE**: To program your ESP32 board, you'll use the Arduino IDE. You'll write code in C/C++ using the Arduino framework.

4. **Visual Studio Code (VS Code)**: While you can use the Arduino IDE for programming, VS Code offers a more versatile and feature-rich development environment. You can use it in combination with platform-specific extensions like "PlatformIO" to write, compile, and upload code to your ESP32.

5. **PowerShell**: PowerShell can be used for various automation tasks, including scripting the deployment and management of your project. You can use PowerShell to automate tasks like compiling and uploading code, managing libraries, or interacting with the ESP32 via a serial connection.

6. **GitHub**: GitHub is a version control platform that allows you to manage your project's source code, collaborate with others, and keep track of changes. You can create a repository for your project and use Git for version control.

Code Styles

The main programming language that I used was HTML and CSS as the main use and programming was already done in the tutorial that was given and the HTML and CSS was what allowed me to stand out from the crowd.

How my Project is unique

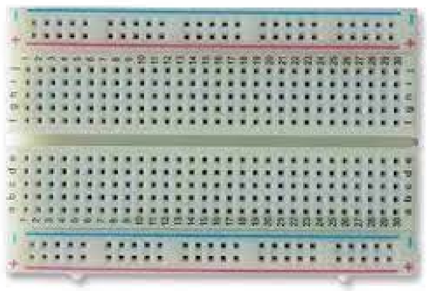
My project is unique as it did not really depend on my teacher's tutorial and it works more effectively as the information shows and updates quicker as the normal delay was 2000 however mine was 1500. Despite this my project is also unique as it requires minimal equipment, other projects took 10’s of wires and complicated code while my code was short, simple and effective, meaning it works faster and there is less room for error and if there are errors it is much quicker to fix.

Items used

Items used to make this project were simple yet effective as I only used:

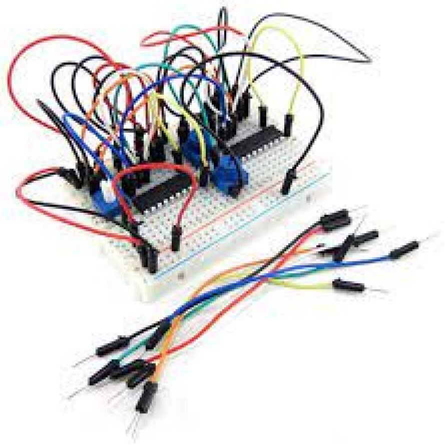
Materials

1 Breadboard

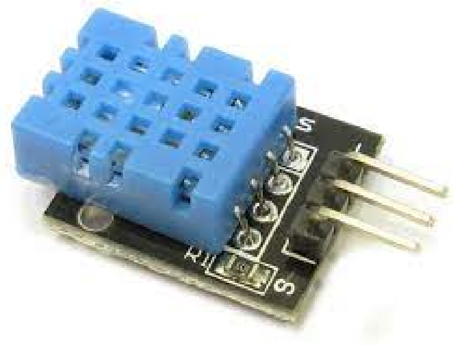


1 ESP32

3x wires (any length)



1 DHT11



Equipment

1 usbc to usb cable (Short length)

1 laptop (Any Microsoft laptop)

What you can use it for

A web server that provides temperature and humidity information for the room you are in can be used for various purposes, both practical and informative. Here are some common use cases for such a web server:

1. **Home Automation and Control**: You can integrate the temperature and humidity data into a home automation system. This allows you to control devices like thermostats, humidifiers, or air conditioning units to maintain a comfortable environment.

2. **Energy Efficiency**: By monitoring room temperature and humidity over time, you can analyse patterns and make adjustments to your heating and cooling systems to save energy and reduce utility costs.

3. **Health and Comfort**: People with specific health conditions or allergies might use this data to ensure that the room's conditions are within a comfortable and healthy range.

4. **Remote Monitoring**: If you're away from home, you can remotely access the web server to check the room's temperature and humidity. This is particularly useful if you're concerned about pets, plants, or sensitive equipment that might be affected by environmental conditions.

5. **Data Logging and Analysis**: You can collect historical data to track trends, analyze changes, and make informed decisions about climate control and other aspects of the room.

6. **Alerts and Notifications**: Set up the web server to send alerts or notifications if temperature or humidity levels go out of a specified range. This can help you take prompt action in case of issues such as leaks, high humidity, or low temperatures.

7. **Research and Experiments**: Researchers, students, or hobbyists might use the data for experiments, data analysis, or academic projects.

8. **Guest Information**: If you have guests, you can share a web link with them so they can check the room's conditions and adjust them to their comfort level.

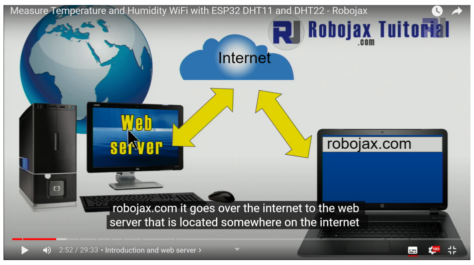
9. **Environmental Monitoring**: The data can be part of a broader environmental monitoring system, providing insights into local conditions over time.

10. **Documentation**: If you're renting a property or using a shared space, the web server can be used to document and verify the room's conditions, potentially for dispute resolution or safety reasons.

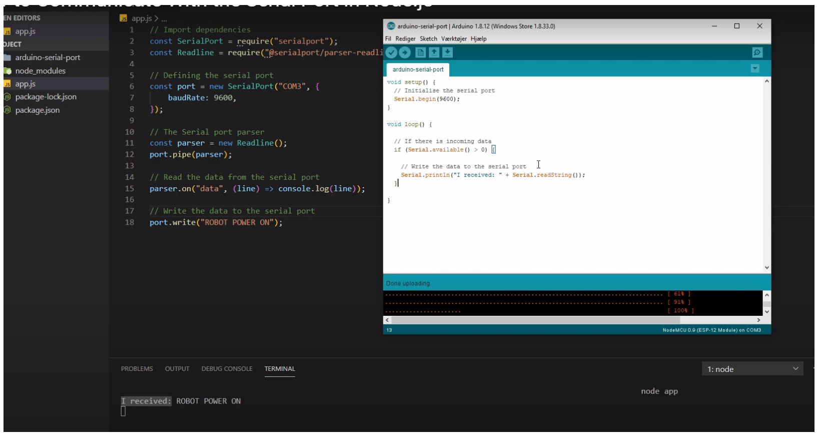
Video

https://clipchamp.com/watch/s7328uj8qBl

Photo’s



Measure Temperature and Humidity WiFi with ESP32 DHT11 and DHT22 - Robojax
Robojax Tuitorial .com
Internet
Web server
robojax.com
robojax.com it goes over the internet to the web server that is located somewhere on the internet
2:52 / 29:33 • Introduction and web server ›



```
// Import dependencies
const SerialPort = require("serialport");
const Readline = require("@serialport/parser-readli

// Defining the serial port
const port = new SerialPort("COM3", {
    baudRate: 9600,
});

// The Serial port parser
const parser = new Readline();
port.pipe(parser);

// Read the data from the serial port
parser.on("data", (line) => console.log(line));

// Write the data to the serial port
port.write("ROBOT POWER ON");
```

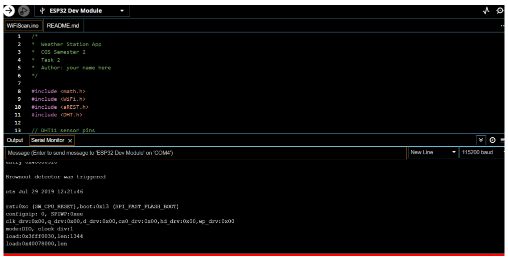
```
void setup() {
  // Initialise the serial port
  Serial.begin(9600);
}

void loop() {

  // If there is incoming data
  if (Serial.available() > 0) {

    // Write the data to the serial port
    Serial.println("I received: " + Serial.readString());
  }

}
```

Done uploading.

```
......................................................................... [ 61% ]
......................................................................... [ 91% ]
....................                                                      [ 100% ]
```

NodeMCU 0.9 (ESP-12 Module) on COM3

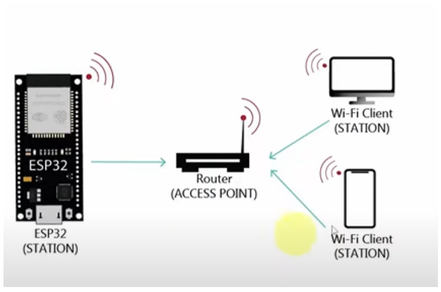
```
node app
I received: ROBOT POWER ON
```

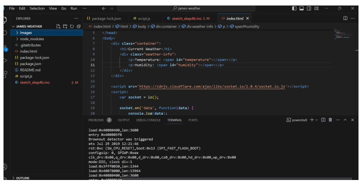


```
/*
 *  Weather Station App
 *  CGS Semester 2
 *  Task 2
 *  Author: your name here
 */

#include <math.h>
#include <WiFi.h>
#include <aREST.h>
#include <DHT.h>

// DHT11 sensor pins
```

Message (Enter to send message to 'ESP32 Dev Module' on 'COM4') — New Line — 115200 baud

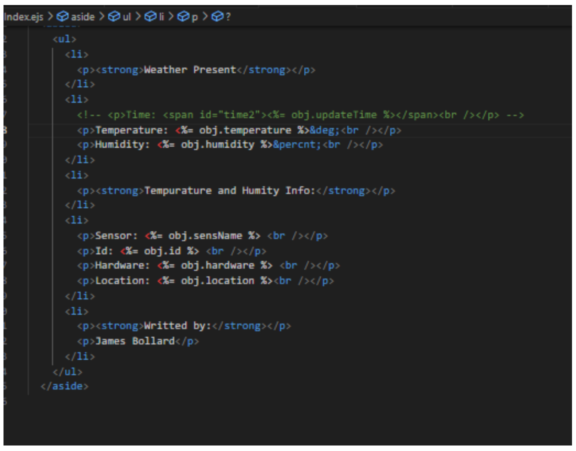
```
Brownout detector was triggered

ets Jul 29 2019 12:21:46

rst:0xc (SW_CPU_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:1344
load:0x40078000,len
```



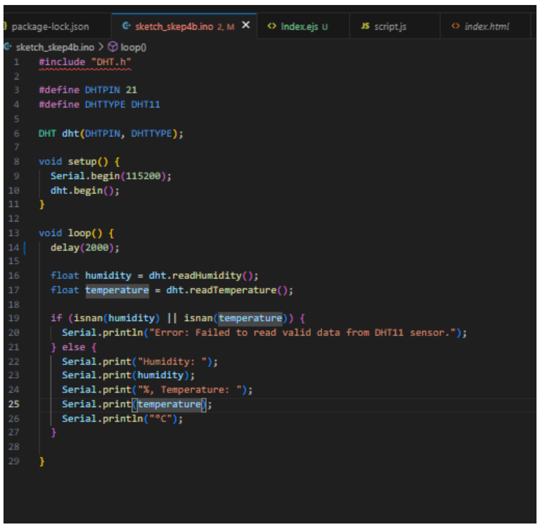


```
</head>
<body>
    <div class="container">
        <h1>Current Weather</h1>
        <div class="weather-info">
            <p>Temperature: <span id="temperature"></span></p>
            <p>Humidity: <span id="humidity"></span></p>
        </div>
    </div>

    <script src='https://cdnjs.cloudflare.com/ajax/libs/socket.io/2.0.4/socket.io.js'></script>
    <script>
        var socket = io();

        socket.on('data', function(data) {
            console.log(data);
```

```
load:0x40080400,len:3600
entry 0x400805f0
Brownout detector was triggered
ets Jul 29 2019 12:21:46
rst:0xc (SW_CPU_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:1344
load:0x40078000,len:13964
load:0x40080400,len:3600
```

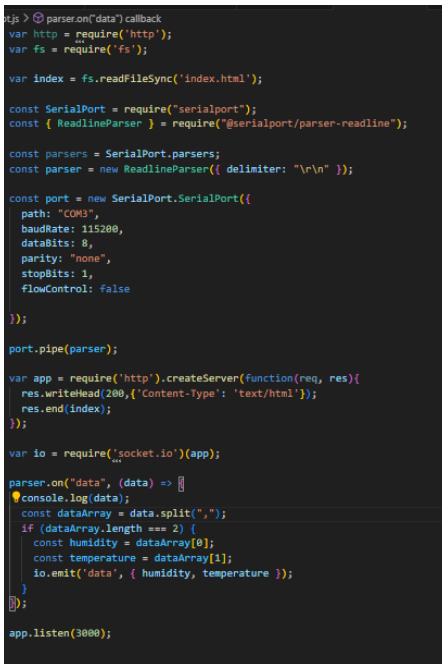

Index.ejs > aside > ul > li > p > ?

```html
    <ul>
      <li>
        <p><strong>Weather Present</strong></p>
      </li>
      <li>
        <!-- <p>Time: <span id="time2"><%= obj.updateTime %></span><br /></p> -->
        <p>Temperature: <%= obj.temperature %>&deg;<br /></p>
        <p>Humidity: <%= obj.humidity %>&percnt;<br /></p>
      </li>
      <li>
        <p><strong>Tempurature and Humity Info:</strong></p>
      </li>
      <li>
        <p>Sensor: <%= obj.sensName %> <br /></p>
        <p>Id: <%= obj.id %> <br /></p>
        <p>Hardware: <%= obj.hardware %> <br /></p>
        <p>Location: <%= obj.location %><br /></p>
      </li>
      <li>
        <p><strong>Writted by:</strong></p>
        <p>James Bollard</p>
      </li>
    </ul>
  </aside>
```



sketch_skep4b.ino > loop()

```cpp
#include "DHT.h"

#define DHTPIN 21
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(115200);
  dht.begin();
}

void loop() {
  delay(2000);

  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();

  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Error: Failed to read valid data from DHT11 sensor.");
  } else {
    Serial.print("Humidity: ");
    Serial.print(humidity);
    Serial.print("%, Temperature: ");
    Serial.print(temperature);
    Serial.println("°C");
  }

}
```


t.js > parser.on("data") callback

```javascript
var http = require('http');
var fs = require('fs');

var index = fs.readFileSync('index.html');

const SerialPort = require("serialport");
const { ReadlineParser } = require("@serialport/parser-readline");

const parsers = SerialPort.parsers;
const parser = new ReadlineParser({ delimiter: "\r\n" });

const port = new SerialPort.SerialPort({
  path: "COM3",
  baudRate: 115200,
  dataBits: 8,
  parity: "none",
  stopBits: 1,
  flowControl: false

});

port.pipe(parser);

var app = require('http').createServer(function(req, res){
  res.writeHead(200,{'Content-Type': 'text/html'});
  res.end(index);
});

var io = require('socket.io')(app);

parser.on("data", (data) => {
  console.log(data);
  const dataArray = data.split(",");
  if (dataArray.length === 2) {
    const humidity = dataArray[0];
    const temperature = dataArray[1];
    io.emit('data', { humidity, temperature });
  }
});

app.listen(3000);
```

TimeLine

19/9/2023- Set up code and materials and equipment

21/9/2023- fix bugs and connect esp32

22/9/2023- fix problem of dht11 not working properly

26/9/2023- make sure it works

1/10/2023- add features to html and css

15/10/2023- start READ_ME

19/10/2023- Finish READ_ME

20/10/2023-Finalise READ_ME and submit

Instructions To Use

To access and view data from a web server running on an Arduino ESP32 board that provides temperature and humidity information, you can follow these instructions:

Step 1: Set Up Your Hardware
1. Ensure you have an Arduino ESP32 development board.
2. The board should already have a web server program running on it, serving temperature and humidity data.
3. If not, make sure that your ESP32 board is equipped with a DHT sensor (e.g., DHT11 or DHT22) and that it is properly connected to measure temperature and humidity.

Step 2: Connect to the ESP32's Wi-Fi
1. Power on your ESP32 board.
2. On your computer or mobile device, go to the Wi-Fi settings.
3. Look for a Wi-Fi network with a name that matches your ESP32's network SSID (Service Set Identifier). It is typically something like "ESP32_XXXXXX" where "XXXXXX" is a unique identifier.
4. Connect to the ESP32's Wi-Fi network.

Step 3: Access the Web Server
1. Open a web browser on your computer or mobile device.
2. In the browser's address bar, enter the IP address of your ESP32. This IP address can usually be found in the Serial Monitor when you upload your code to the ESP32.

Step 4: View Temperature and Humidity Data
1. You should see a web page displayed in your browser that shows the temperature and humidity data, as served by the ESP32's web server.
2. The temperature and humidity data will be shown on the webpage, along with any other information or controls provided by the web server.

That's it! You've successfully accessed and viewed the temperature and humidity data from a web server running on your Arduino ESP32 board. You don't need to modify or create any code on the ESP32; you are simply connecting to the web server and viewing the data it provides.

Reference list

Anon, (2019). *ESP32 with DHT11/DHT22 Temperature and Humidity Sensor using Arduino IDE | Random Nerd Tutorials*. [online] Available at: https://randomnerdtutorials.com/esp32-dht11-dht22-temperature-humidity-sensor-arduino-ide/.

Arduino (2018). *Arduino* . [online] Arduino.cc. Available at: https://www.arduino.cc/.

ChatGPT (2023). *ChatGPT*. [online] chat.openai.com. Available at: https://chat.openai.com/.

Create And Play Electronics. (n.d.). *Create And Play Electronics*. [online] Available at: https://createandplayelectronics.com/ [Accessed 20 Oct. 2023].

docs.arduino.cc. (n.d.). *Nano ESP32 | Arduino Documentation*. [online] Available at: https://docs.arduino.cc/hardware/nano-esp32 [Accessed 20 Oct. 2023].

GitHub (2023). *GitHub*. [online] GitHub. Available at: https://github.com/.

Microsoft (2019). *Visual Studio*. [online] Visual Studio. Available at: https://visualstudio.microsoft.com/.

upesy (2023). *DHT11 ESP32 in Arduino Code: Measuring humidity & temperature*. [online] uPesy. Available at: https://www.upesy.com/blogs/tutorials/dht11-humidity-temperature-sensor-with-arduino-code-on-esp32-board [Accessed 20 Oct. 2023].

www.youtube.com. (n.d.). *ESP32 DHT Web Server Project*. [online] Available at: https://www.youtube.com/watch?v=_GohoygyCNE&t=56s [Accessed 20 Oct. 2023].

www.youtube.com. (n.d.). *ESP32 DHT11/DHT22 Asynchronous Web Server (auto updates Temperature and Humidity)*. [online] Available at: https://www.youtube.com/watch?v=tDdL5urWvH4&t=77s [Accessed 20 Oct. 2023].

www.youtube.com. (n.d.). *How to Communicate With the Serial Port in Node.js*. [online] Available at: https://www.youtube.com/watch?v=__FSpGHx9Ow [Accessed 9 Oct. 2022].

www.youtube.com. (n.d.). *Measure Temperature and Humidity WiFi with ESP32 DHT11 and DHT22 - Robojax*. [online] Available at: https://www.youtube.com/watch?v=JXCcmZUmzy8&t=278s [Accessed 20 Oct. 2023].