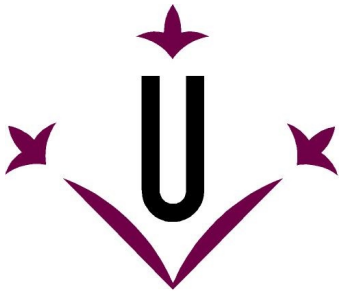

University of Lleida

Master's Degree in Informatics Engineering

Higher Polythecnic School

Data Producer 1

Ubiquitous Computing and Embedded Systems

Francesc Contreras

Albert Pérez

Marc Visa

November 10, 2021

Table of contents

List of Figures

1 Introduction

The purpose of this document is to explain the Data Producer 1 development for the Sprint 1 of the project.

We shall introduce all the steps and software we have used when developing the it and showing images as a real example for detailed explanation.

2 Environment

At first, these are the steps for installing the libraries and setting up the Arduino IDE to get started for the Data Producer 1 development:

1. Install Arduino IDE.
2. Install driver for USB adapter to ESP-01 in PROG mode.
3. Install ESP8266 Board from Boards Manager from Arduino IDE.
4. Install the following libraries from the Manage Libraries of Arduino IDE:
   (a) DHT [1] library by Adafruit.
   (b) Adafruit Unified Sensor by Adafruit.
5. Install the following libraries from GitHub: (These libraries should be added to the Arduino/libraries folder):
   (a) ESPAsynchWebServer
   (b) ESPAsyncTCP
6. Connect the ESP-01[2] along with the USB adapter to your machine.
7. Code the program for the ESP-01.

[1] Temperature and humidity sensor
[2] Low-cost WiFi microchip with built-in TCP/IP networking software, and microcontroller capability

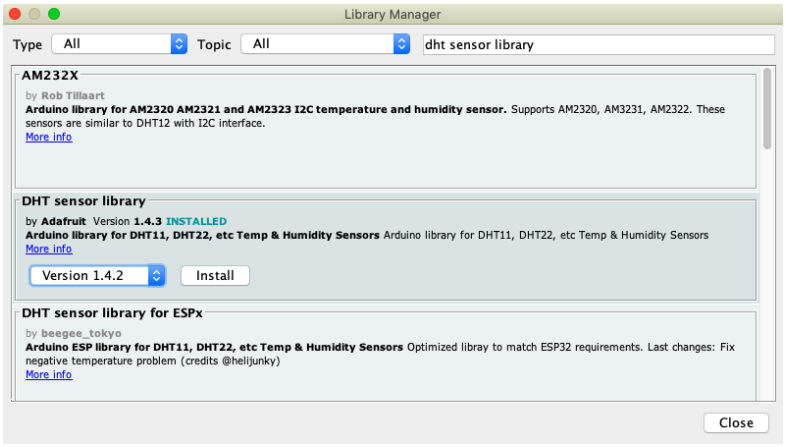

Figure 1: Installing the DHT sensor library for ESPx from the Arduino Manage Libraries.

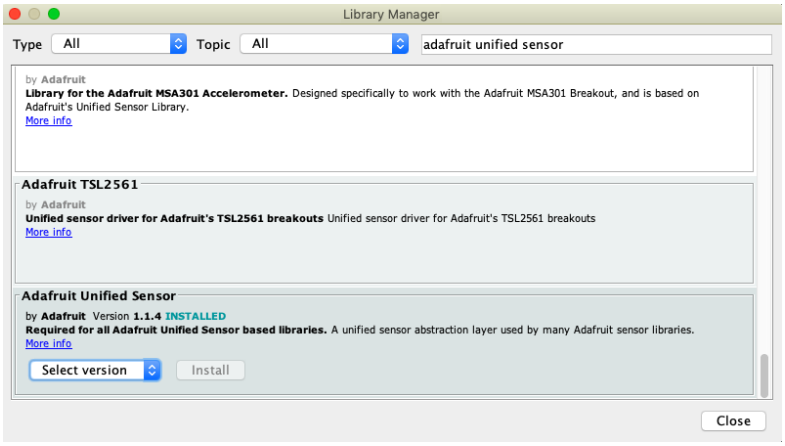

Figure 2: Installing the Adafruit Unified Sensor from the Arduino Manage Libraries.

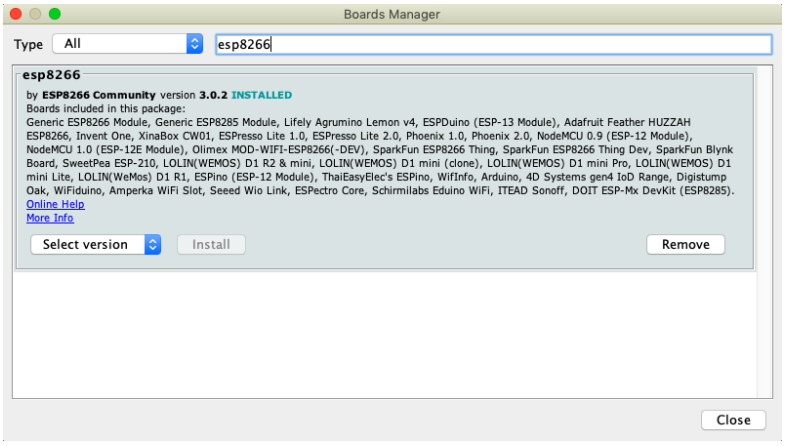

Figure 3: Installing the ESP8266 Board from the Arduino Boards Manager.

2.1 Material required

Subsequently, you will find all the required components for the Data Procuer 1, and a short explanatory description for those not so common.

- **ESP-01 (3.3 V)**: is a Wi-Fi module that allows microcontrollers access to a Wi-Fi network.
- **Protoboard**: A board, having a matrix of small holes to which components may be attached without solder.
- **LCD Screen (5V)**: is a flat-panel display or other electronically modulated optical device that uses the light-modulating properties of liquid crystals combined with polarizers.
- **DHT11 (3.3V)**: is a basic, ultra low-cost digital temperature and humidity sensor.
- **VCC Protoboard adapter (3.3V, 5V)**
- **ESP Programmer module**
- **Wires**

3 Development

Regards the development, this is based on two parts, the first one realated to work directly with the electronic components, and the second by programming the arduino sketch.

The following sections shows you how the electronic schema is in real, as well as how finally we got the visualization of the temperature and the humidity by two different ways.

Moreover, there is the code shown below, but also you may consult the github repository of the project, the one is specified subsequently.

3.1 Hardware

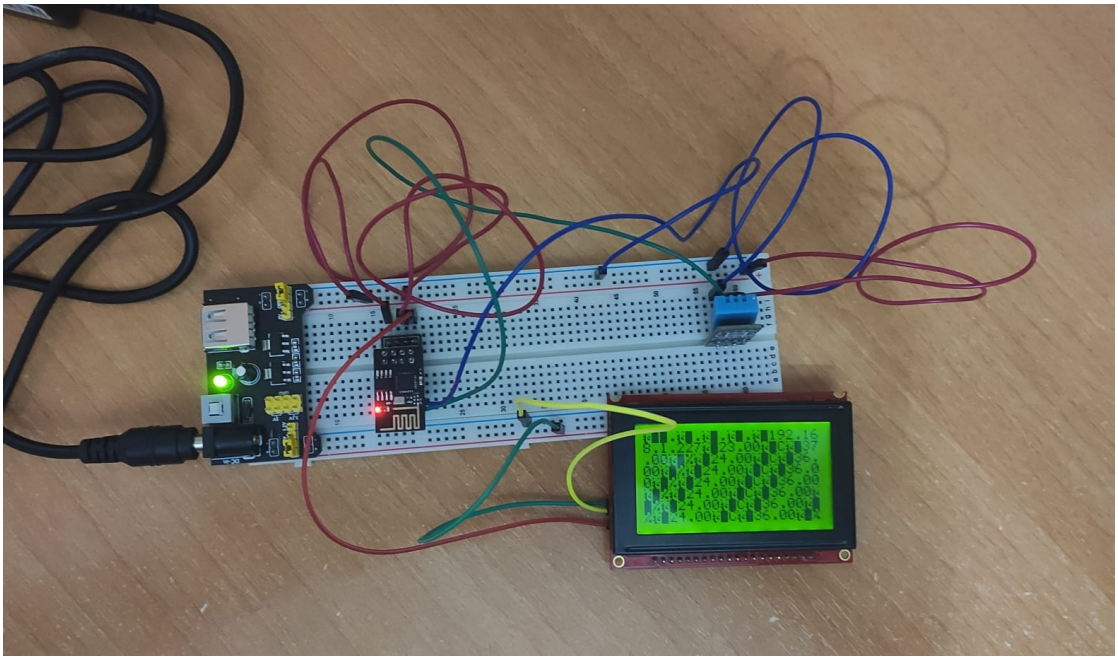

Figure 4: Data Producer 1 assembly

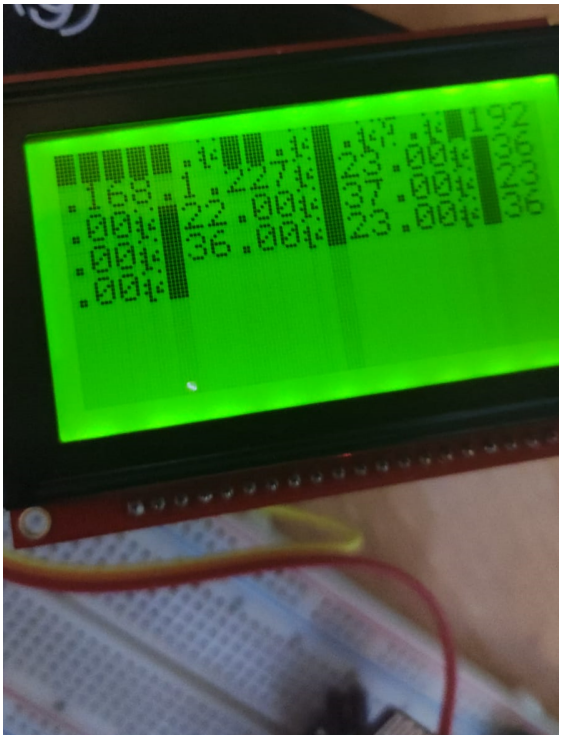

Figure 5: LCD Screen showing visualization

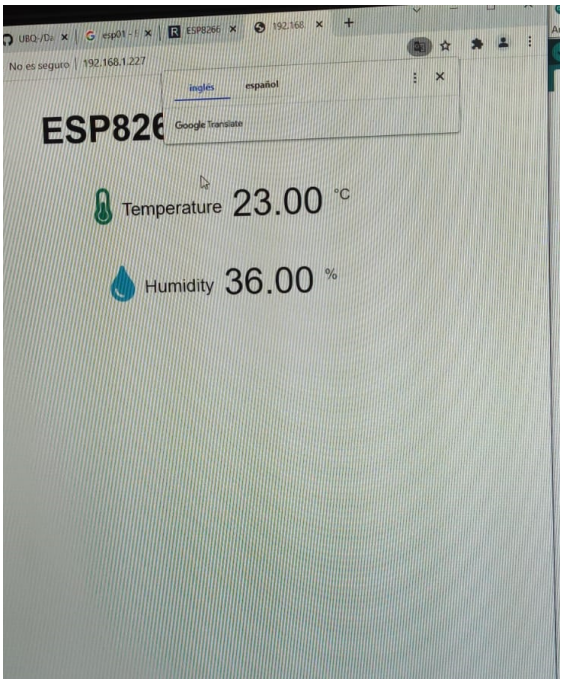

Figure 6: Web client visualization

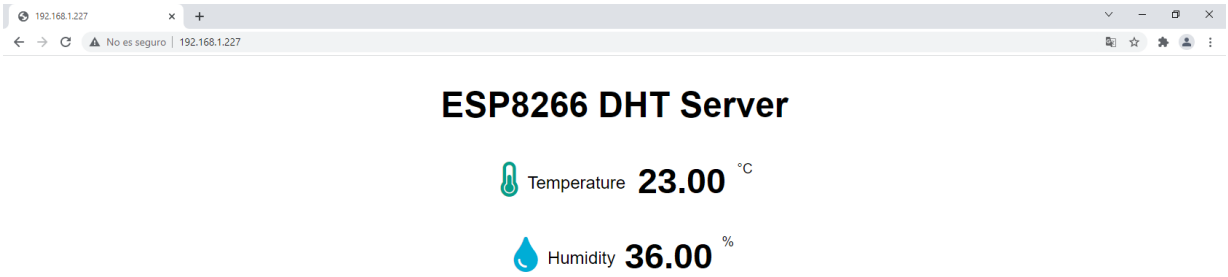

Figure 7: Web Client to visualize the humidity and temperature from DHT11 sensor.

3.2 Software

```
 // 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>

#include <SoftwareSerial.h>

// Replace with your network credentials
const char* ssid = "ADAMO-C6CA";
const char* password = "JA54W6HGFCV7NC";

#define DHTPIN 2   // Digital pin connected to the DHT sensor

// Uncomment the type of sensor in use:
#define DHTTYPE  DHT11    // DHT 11

DHT dht(DHTPIN, DHTTYPE);

// current temperature & humidity, updated in loop()
float t = 0.0;
float h = 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;
    }
  </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">%</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 String(t);
  }
  else if(var == "HUMIDITY"){
    return String(h);
  }
  return String();
}

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

  // 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());
  });

  // Start server
  server.begin();

  delay(10);
}

void loop(){
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    // save the last time you updated the DHT values
    previousMillis = currentMillis;

    float newT = dht.readTemperature();

    if (isnan(newT)) {
      Serial.println("Failed to read from DHT sensor!");
    }
```

```
    else {
      t = newT;
      String outStringT = (String) t+"C";
      Serial.println(outStringT);
    }

    // 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;
      String outStringH = (String) h+"%";
      Serial.println(outStringH);
    }
  }
}
```

4 Code repository

Wind Turbine Generator Github repository

References

[1] ESP8266 DHT11/DHT22 Temperature and Humidity Web Server with Arduino IDE