



University of West Attica

Faculty of Engineering
Department of Electrical
and Electronics Engineering

Internet of Things



Semester Assignment

Name: Alexandros Demirtzoglou

Student ID: 50107150

Email: ee07150@uniwa.gr

Εξάμηνο: 14th

Submission Date: Mon, Jul 1st, 2024

Submitted to: Dimitris Pyromalis

The purpose of this assignment is to construct an electronic framework that measures humidity and temperature from the environment using a DHT11 sensor. These readings are then sent to an ESP9266 module (NodeMCU), which uploads the data to a cloud accessible to users who are connected.

Theoretical Background

NodeMCU is an open-source programmable board increasingly used in Internet of Things applications. Its operating system runs on an ESP8266 processor, which, through an antenna, can connect to the internet and execute tasks assigned via code. Powering and programming it can be done through a micro USB cable.



The ESP8266, designed and manufactured by Espressif Systems, includes all critical components of a modern computer: CPU, RAM, networking (WiFi), and is quite economical. These features make it an excellent choice for various IoT projects.

It also has WiFi capabilities, allowing us to control and operate installations remotely easily. We can set the board's behavior by sending a set of commands to its microcontroller. For this, we use the Arduino Software (IDE).

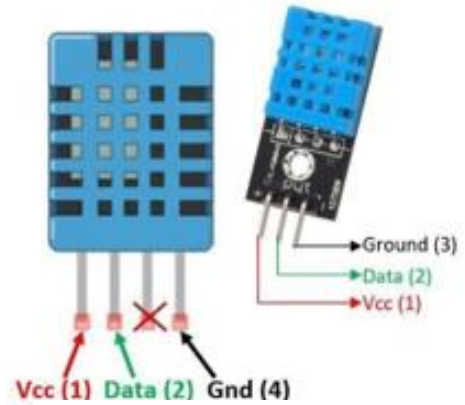
In essence, the ESP8266 adds WiFi connectivity to our projects, enabling wireless connection to a local network or the internet. This allows for numerous possibilities, such as turning electrical devices on or off (using a relay) or controlling other mechanical systems in our home via the internet from a smartphone or any internet-connected computer.

Picture 1- ESP8266



DHT11

DHT11 is a humidity-temperature sensor that uses a capacitive humidity sensor and a thermistor, whose resistance depends significantly on temperature. Its power supply is 3.5V, making it quite economical.



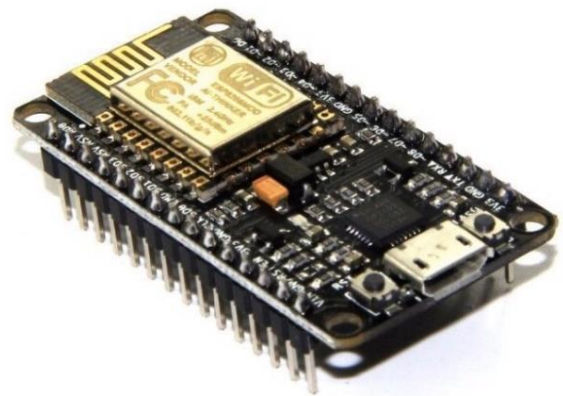
Picture 2 - DHT11

Experimental Procedure

The components used in this assignment include a NodeMCU board ESP8266 WiFi module ESP-12E Lua WiFi:

Data sheet

- Manufacturer: OEM
- Part Number: ESP8266-NODEMCU
- Net Weight: 0.001Kg
- Country of Origin: Kίνα
- Breadboard Compatible
- Includes USB-TTL, plug & play
- 10 GPIO, each GPIO can be PWM, 12C
- FCC CERTIFIED Wi-Fi module
- PCB antenna



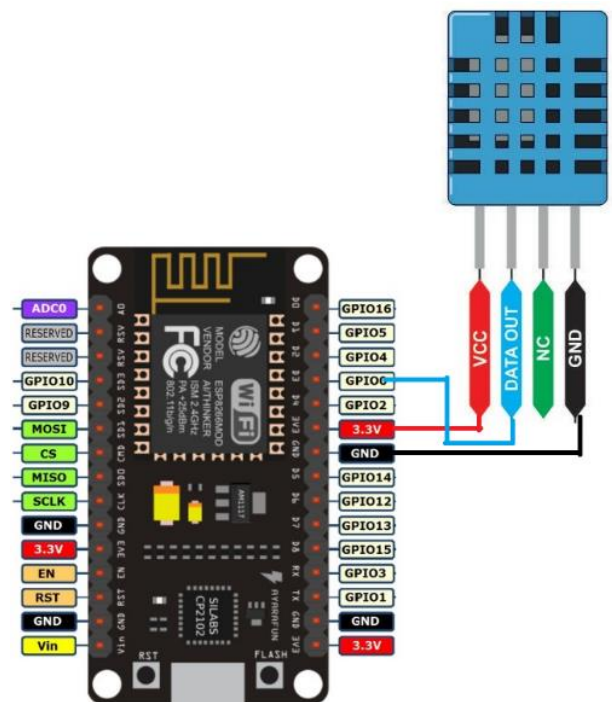
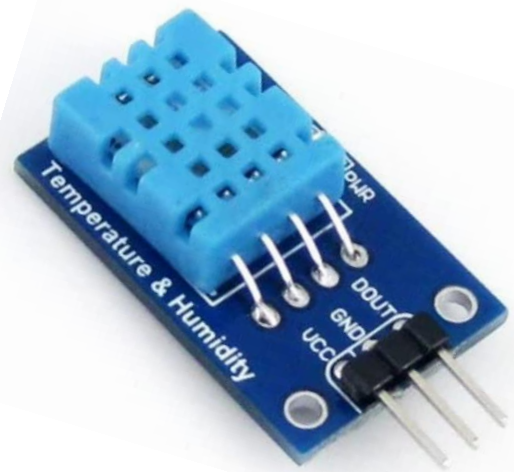
Διαστάσεις

- PCP: 48.3 x25.8 x5mm
- With Pin Headers: 50 x 25.8 x 12.5mm

For measuring the required temperature and humidity, a DHT11 environmental sensor was used.

Data sheet

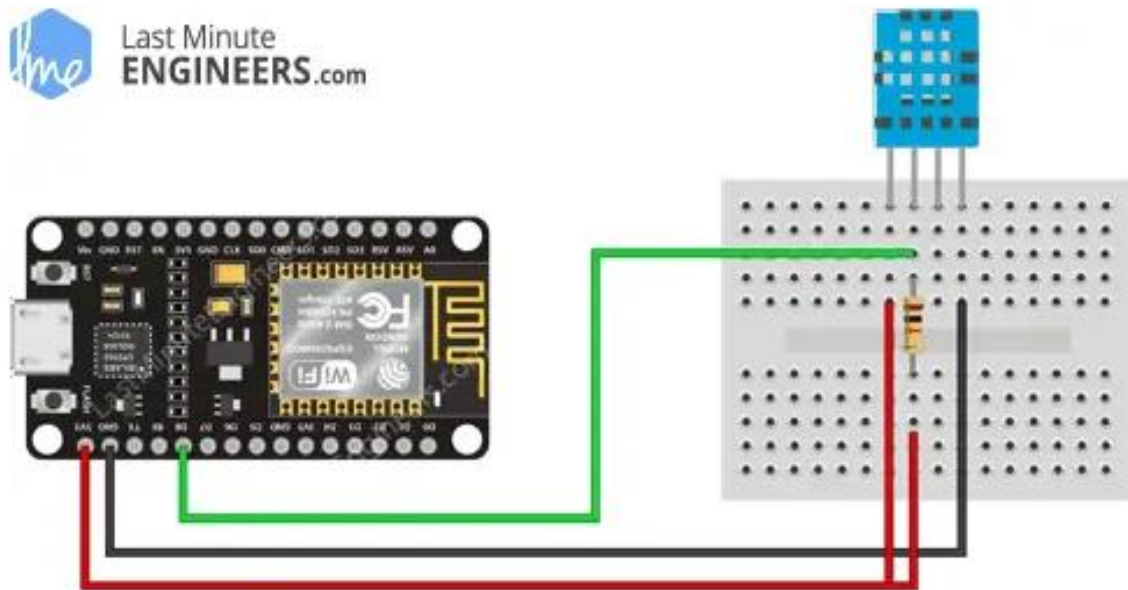
- Manufacturer: WaveShare
- Part Number: 9535
- Net Weight: 0.007kg
- Country of Origin: Kiva
- Breadboard Friendly:
- Sensor Type: Temperature, Humidity
- Typical Input Voltage: 3.3VDC - 5VDC - 5.5VDC
- Operating Current: 0.3mA
- Interface: Digital
- Communication Protocol: Single Wire
- Temperature-Humidity Sensor, DHT11 Onboard
- Temperature
- Resolution: 1°C
- Accuracy: $\pm 2^{\circ}\text{C}$
- Measuring range: 0°C ~ 50°C
- Humidity
- Resolution: 1%RH
- Accuracy: $\pm 5\%\text{RH}$ (0~50°C)
- Measuring range: 20%RH ~ 90%RH (25°C)
- Operating voltage: 3.3V ~ 5.5 V
- Recommended storage condition
- Temperature : 10°C ~40°C
- Humidity: 60%RH or below



The following wiring was used:

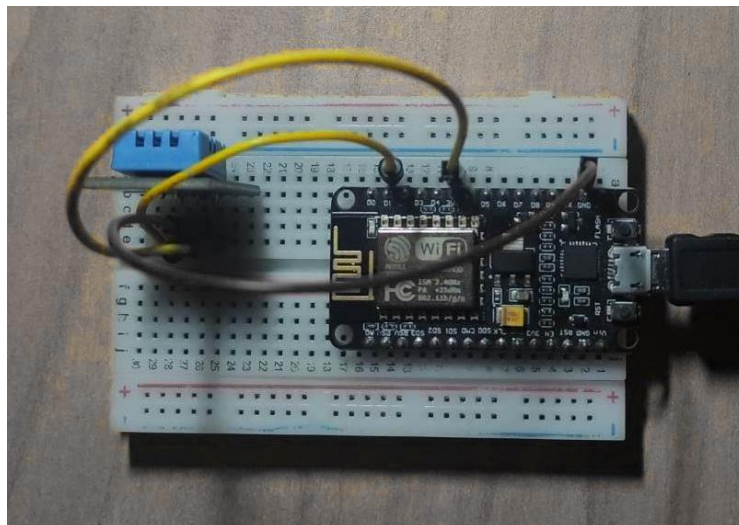
Picture 3 - Wiring

This setup on a breadboard, according to the tutorial "Wiring a DHT11/DHT22 sensor to an ESP8266," appears as follows:



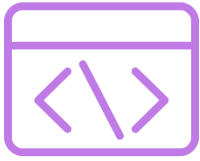
Picture 4 – Breadboard Wiring

This is what our own implementation looks like:



Picture 5 - Implementation

Programming



Arduino IoT Cloud is fully integrated into the Arduino Create ecosystem. Therefore, template code can be created in Arduino IoT Cloud, then edited and uploaded to the board using the Arduino Web Editor.

Here is the process of implementing Arduino IoT Cloud in steps:

Step 1

Temperature and humidity sensor Setup Sketch **6** Metadata

Cloud Variables **ADD**

	Name ↓	Last Value	Last Update	
<input type="checkbox"/>	faros <code>bool faros;</code>	-		⋮
<input type="checkbox"/>	humidityValue <code>CloudRelativeHumidity humidityValue;</code>	88	01 May 2023 21:20:10	⋮
<input type="checkbox"/>	tempValue <code>CloudTemperatureSensor tempValue;</code>	21.1	01 May 2023 21:20:10	⋮

Associated Device

Sharl

ID: 65c6936b-182c-427b-9fda-...

Type: NodeMCU 1.0 (ESP-12E Module)

Status: ● Online

Change Detach

Picture 6

Three variables were defined here. The first variable is "faros," referring to the LED we placed but burned out because I forgot to install the capacitor in the first implementation.

Next, we have the variables "humidityValue" and "tempValue," referring to the different values of humidity and temperature, respectively. The name assigned to the Arduino is "Sharl," and the type of board used (NodeMCU 1.0) is displayed.

Step 2

Here, in images 7 and 8, the variables are defined. For temperature, we set that the indication would be calculated in Celsius degrees, and the measurement would take place every 60 seconds to observe any changes. Similarly, humidity was set to display as a percentage, and its measurement also takes place every 60 seconds.

Name

tempValue

Sync with other Things

i

Temperature Sensor (°C) eg. 1 °C

Declaration

CloudTemperatureSensor tempValue ;

i

Variable Permission

i

☐

Read & Write

☒

Read Only

Variable Update Policy

i

☐

On change

☒

Periodically

Every

60

s

Picture 7

Name

humidityValue

Sync with other Things

i

Relative Humidity eg. 1 %

Declaration

CloudRelativeHumidity humidityValue ;

i

Variable Permission

i

☒

Read & Write

☐

Read Only

Variable Update Policy

i

☐

On change

☒

Periodically

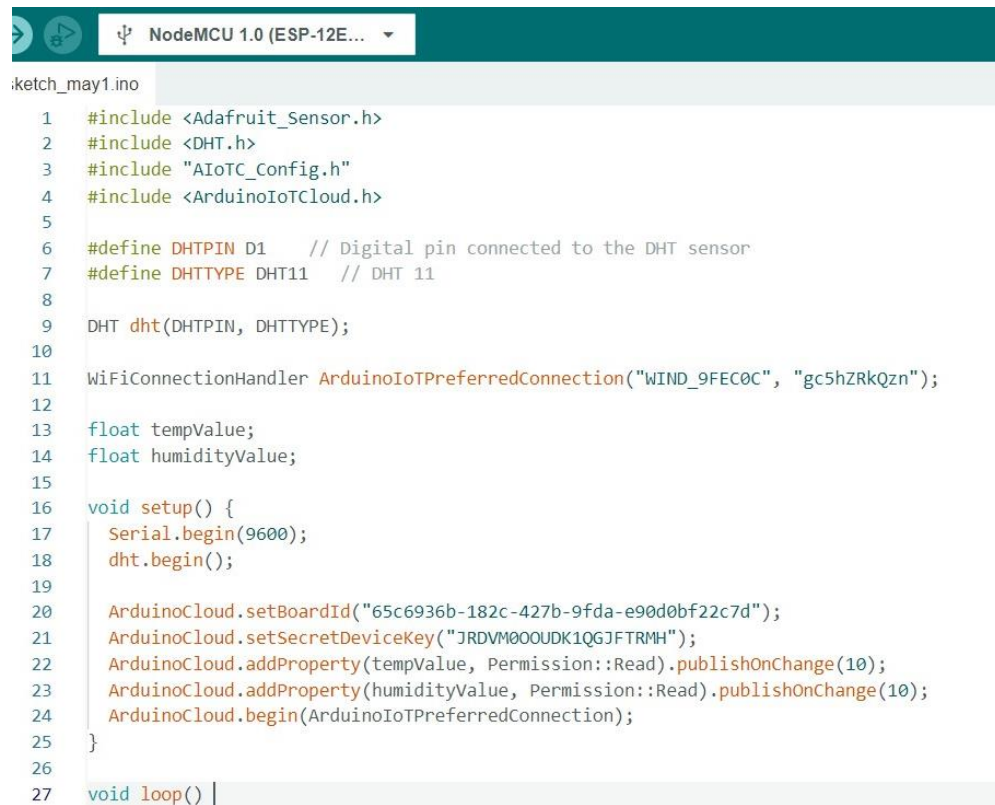
Every

60

s

Picture 8

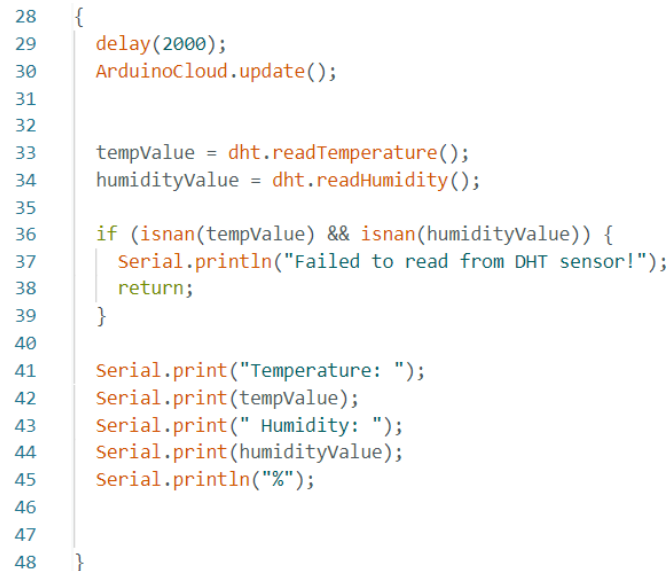
Step 3



```
1  #include <Adafruit_Sensor.h>
2  #include <DHT.h>
3  #include "AIoTC_Config.h"
4  #include <ArduinoIoTCLOUD.h>
5
6  #define DHTPIN D1    // Digital pin connected to the DHT sensor
7  #define DHTTYPE DHT11    // DHT 11
8
9  DHT dht(DHTPIN, DHTTYPE);
10
11 WiFiConnectionHandler ArduinoIoTPreferredConnection("WIND_9FEC0C", "gc5hZRkQzn");
12
13 float tempValue;
14 float humidityValue;
15
16 void setup() {
17   Serial.begin(9600);
18   dht.begin();
19
20   ArduinoCloud.setBoardId("65c6936b-182c-427b-9fda-e90d0bf22c7d");
21   ArduinoCloud.setSecretDeviceKey("JRDVM000UDK1QGJFTRMH");
22   ArduinoCloud.addProperty(tempValue, Permission::Read).publishOnChange(10);
23   ArduinoCloud.addProperty(humidityValue, Permission::Read).publishOnChange(10);
24   ArduinoCloud.begin(ArduinoIoTPreferredConnection);
25 }
26
27 void loop() |
```

Picture 9

Step 4



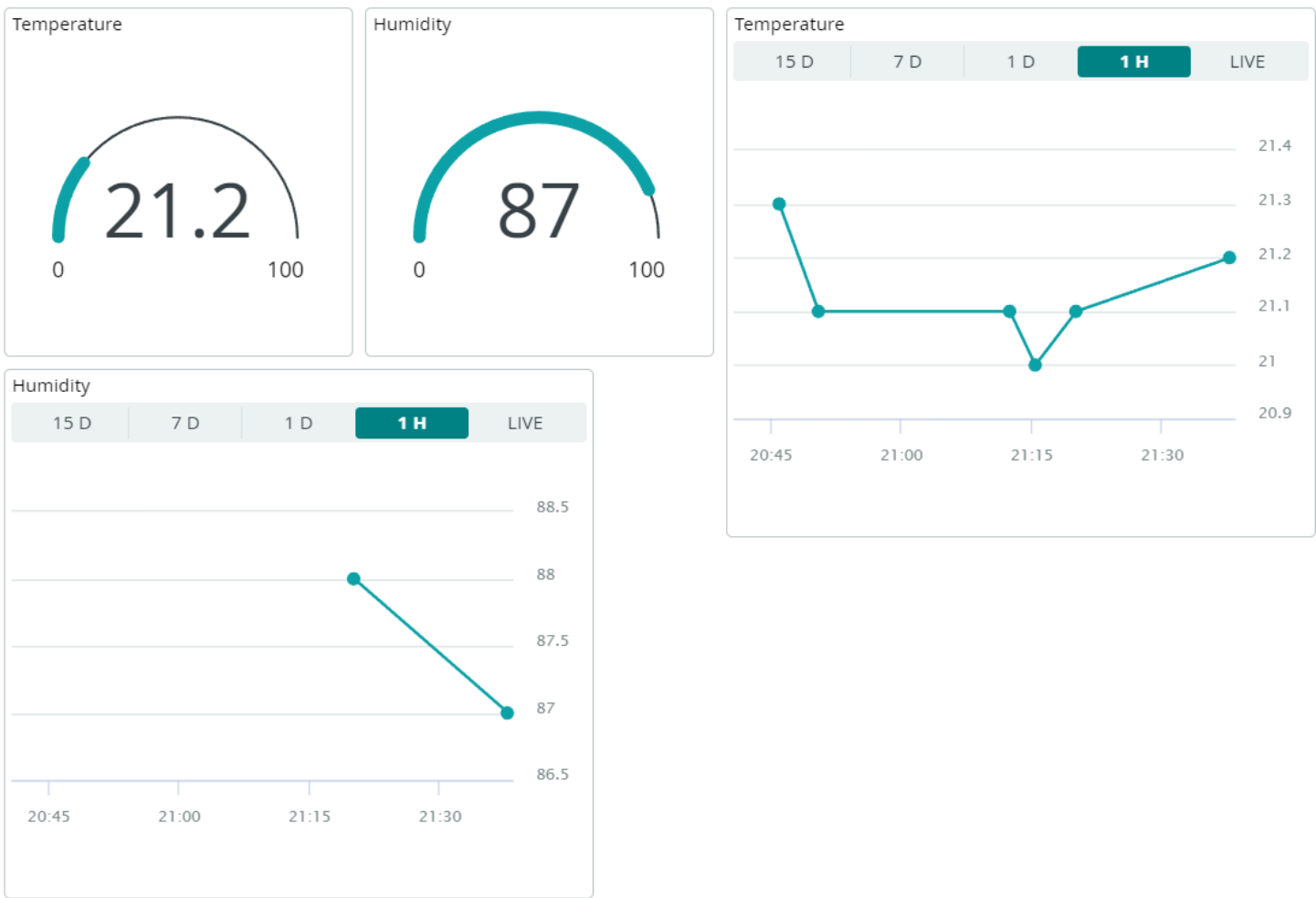
```
28 {
29   delay(2000);
30   ArduinoCloud.update();
31
32
33   tempValue = dht.readTemperature();
34   humidityValue = dht.readHumidity();
35
36   if (isnan(tempValue) && isnan(humidityValue)) {
37     Serial.println("Failed to read from DHT sensor!");
38     return;
39   }
40
41   Serial.print("Temperature: ");
42   Serial.print(tempValue);
43   Serial.print(" Humidity: ");
44   Serial.print(humidityValue);
45   Serial.println("%");
46
47
48 }
```

Picture 10

In images 9 and 10, the code developed for the microcontroller is presented. Initially, in image 9, after defining the necessary libraries, we write the name and password of the Wi-Fi network to be used. Then, we define the temperature and humidity as float variables. Next, some Arduino variables are defined, including the Secret Device Key, allowing the Arduino to be recognized and the measurements to be taken.

Below are the results:

Temperature and humidity monitoring



References

1. https://www.youtube.com/watch?v=ZZ_VzdnS3K4
 2. <https://lastminuteengineers.com/esp8266-dht11-dht22-web-server-tutorial/>
 3. https://grobotronics.com/nodemcu-lua-based-esp8266.html?fbclid=IwAR28OCfn--pIZTp1i-CsTf03KAB9MqfuQzbJ6SEI6R100_59JXdPHm4mG_g
 4. https://grobotronics.com/waveshare-temperature-humidity-sensor-dht11.html?fbclid=IwAR0zgukiKbru678AgbjvTzREuUDSU6qJgty3_p3brSPjJkkASmeAynlDmeg
 5. <https://edurobotics.gr/ti-einai-to-breadboard-kai-pws-leitoyrgei-me-to-arduino/>
 6. <https://create.arduino.cc/projecthub/products/arduino-iot-cloud>
-