

Environment Monitoring



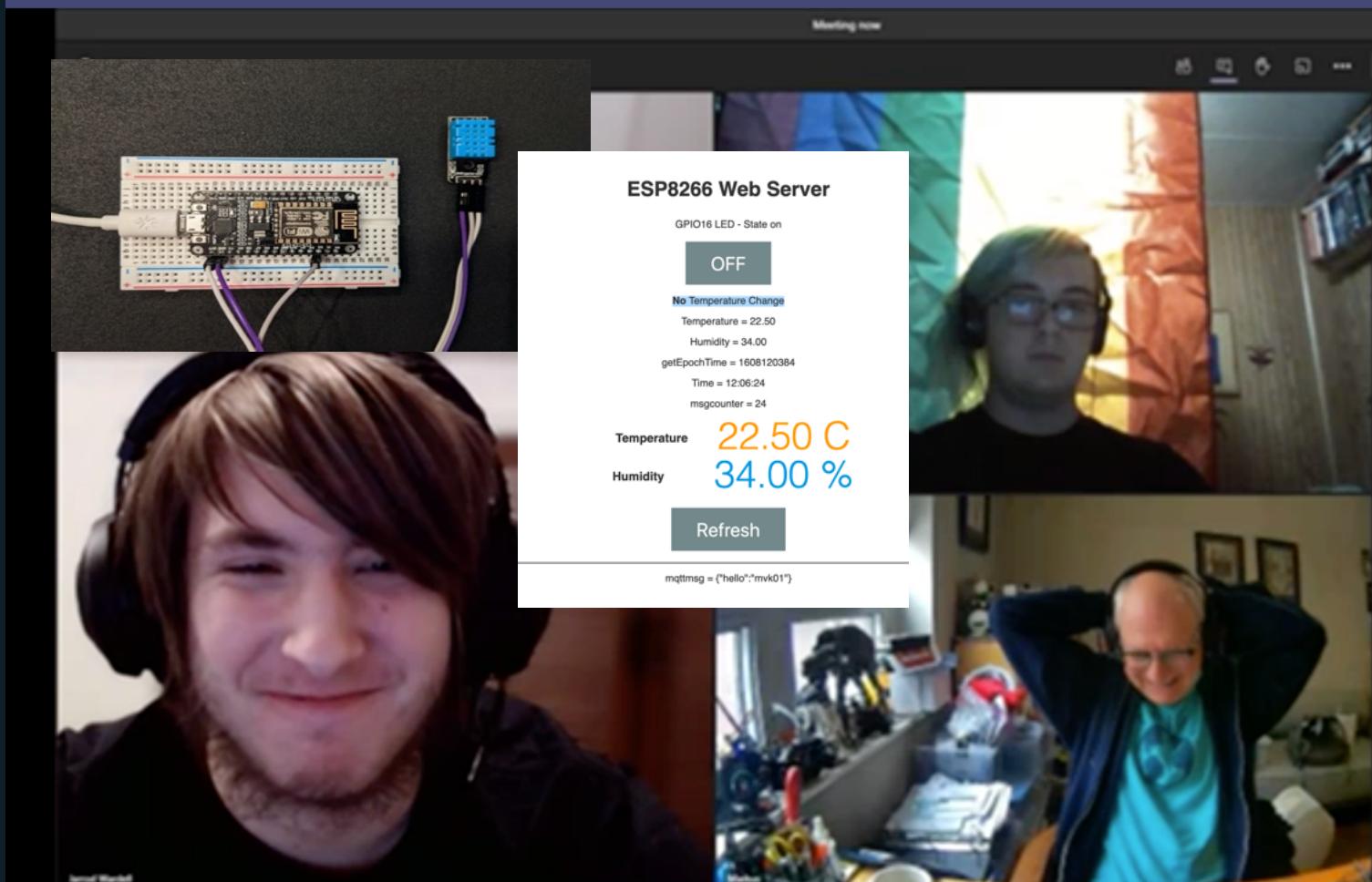
Markus van Kempen
IBM **SPEED**

✉ mvk@ca.ibm.com

🐦 @markusvankempen

Version: 20210210

<https://discord.com/channels/809863505908858931/809866377765715978>



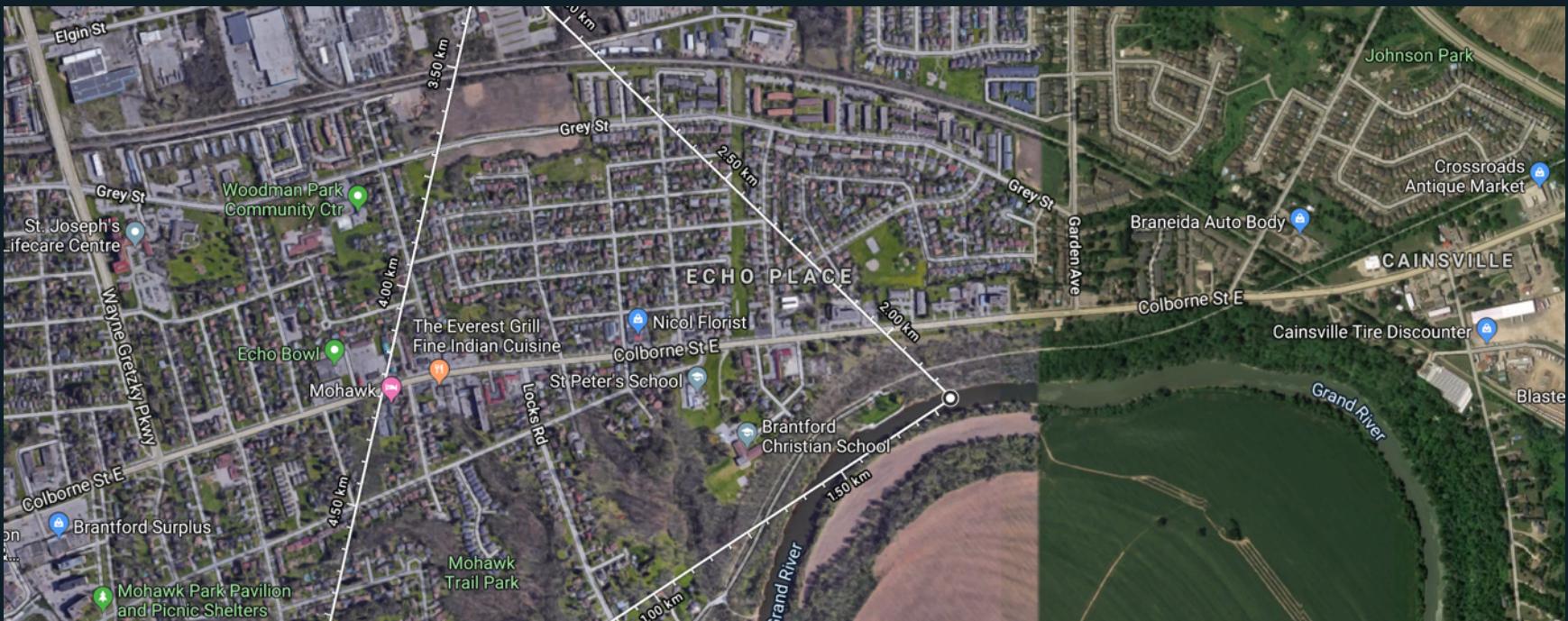
SNP STEAM ACADEMY

Code and instruction for this class can be found here

<https://github.com/markusvankempen/esp8266andDHT11>



Our Goal – Environmental Monitoring Project



Why LoraWAN

- It's free - no Telco costs
- LORA = LongRange 10km and LowPower (LPWan)
- Lots of use-cases / examples and support available
-
- More infos
- <https://www.semtech.com/lora/why-lora>
- <https://os.mbed.com/docs/mbed-os/v5.15/tutorials/LoRa-tutorial.html>



Lora Gateway & LoraServer



- Has Wifi/lora/GPS/LTE
- Power via Power over Ethernet (POE)
- More infos here
- <https://www.rakwireless.com/en-us/products/lpwlan-gateways-and-concentrators/rak7249>
- So far tested Gateway setup with different lora Nodes connection to TTN and MQTT
- ToDo's
 - Test LTE setup
 - Need Solar Kit
 - <https://store.rakwireless.com/products/solar-kit?variant=31385712885805>

Lora Nodes –with batteries and solar panel – TTGO -T-Beam



TTGO ... lots of software support Arduino based and wifi support
Pro

Really good power management ... all one ... with gps options for tracking

<http://www.lilygo.cn/products.aspx?TypeId=50003&FlId=t3:50003:3>

https://github.com/JoepSchyns/Low_power_TTGO_T-beam

<https://github.com/LilyGO/TTGO-T-Beam>

https://tinyomics.com/wiki/TTGO_T-Beam



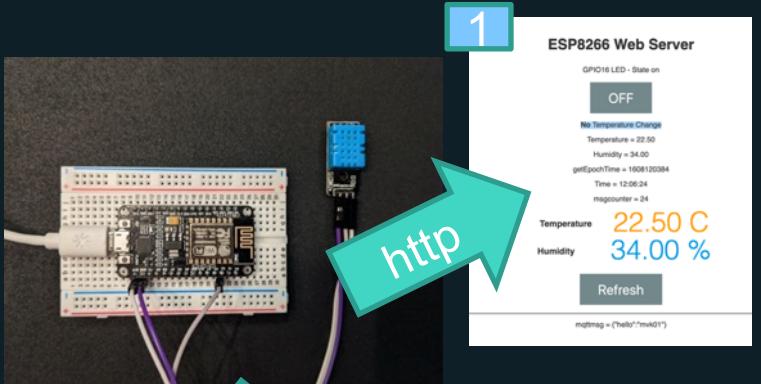
Today



We will simulate the LoRa network using a WIFI network with our esp8266 & DHT11 to send temperature data via mqtt to a server and save the data into a database and than “surf” the data and visualize it using Grafana.

In our esp8266 journey we will setup the Arduino IDE, using the esp8266 as a webserver, Integrate the NetworkTimeProtocol and look at the deep sleep mode for es8266 and and of course send and store the temperature data.

Overview – ESP8266 and DHT11



Agenda

1st Chapter (One) – Arduino and ESP8266 Setup

2nd Chapter (two) – ESP8266 as a webserver

3rd Chapter (three) – ESP8266 and NTP/RTC (Time Server)

4th Chapter (four) – ESP8266 and MQTT



1st Chapter (One)



Arduino IDE setup

ESP and DHT11 wiring

Esp8266 and dht11 test program

Some exercises/play

Arduino IDE setup - esp8266 board

We need to add the esp8266 board and libraries to the IDE

Instructions

- Start Arduino and open Preferences window.
- Enter https://arduino.esp8266.com/stable/package_esp8266com_index.json into *Additional Board Manager URLs* field. You can add multiple URLs, separating them with commas.
- Open Boards Manager from Tools > Board menu and find esp8266 platform.
- Select the version you need from a drop-down box.
- Click *install* button.
- Don't forget to select your ESP8266 board from Tools > Board menu after installation

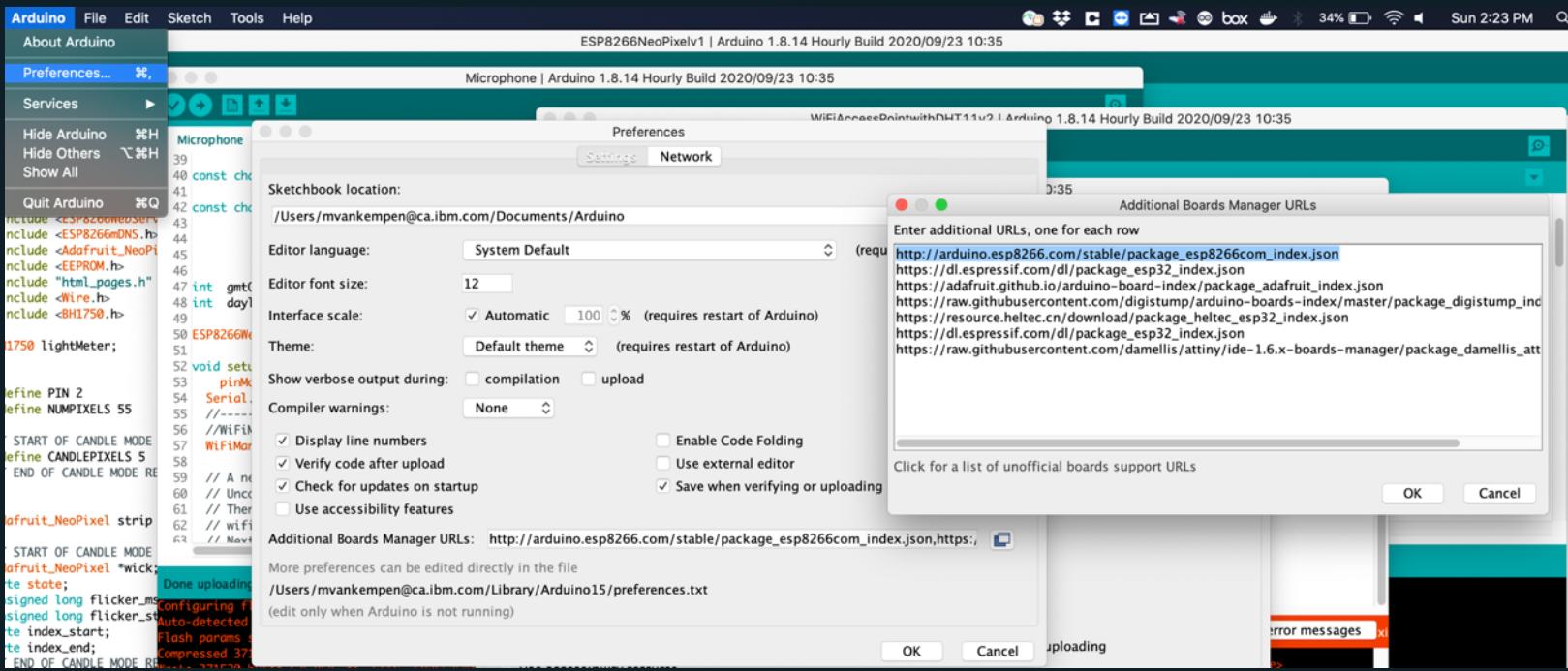
More reference

<https://arduino-esp8266.readthedocs.io/en/latest/installing.html>

<https://www.hackster.io/rajthavti/dht11-sensor-interfacing-with-nodemcu-086762>



ESP8266 board import



Arduino IDE setup - dht11 library

You should have the [Arduino IDE](#) software running at this time. Next is to install our DHT library, which can be done through the Arduino Library Manager:

Sketch→Include Library→Manage Libraries...

Enter “dht” in the search field and look through the list for “**DHT sensor library by Adafruit.**” Click the “Install” button, or “Update” from an earlier version.

DHT Sensor Library: <https://github.com/adafruit/DHT-sensor-library>

More reference

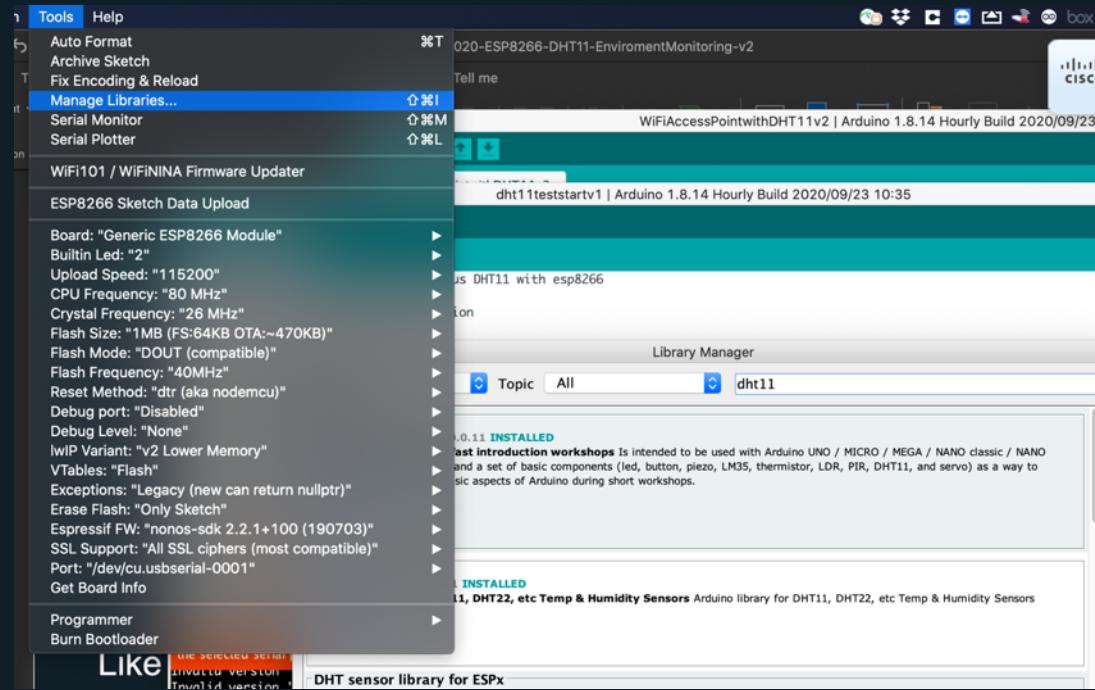
<https://arduino-esp8266.readthedocs.io/en/latest/installing.html>

<https://www.hackster.io/rajthavti/dht11-sensor-interfacing-with-nodemcu-086762>

<https://learn.adafruit.com/dht/downloads?view=all>



Import DHT11 Library



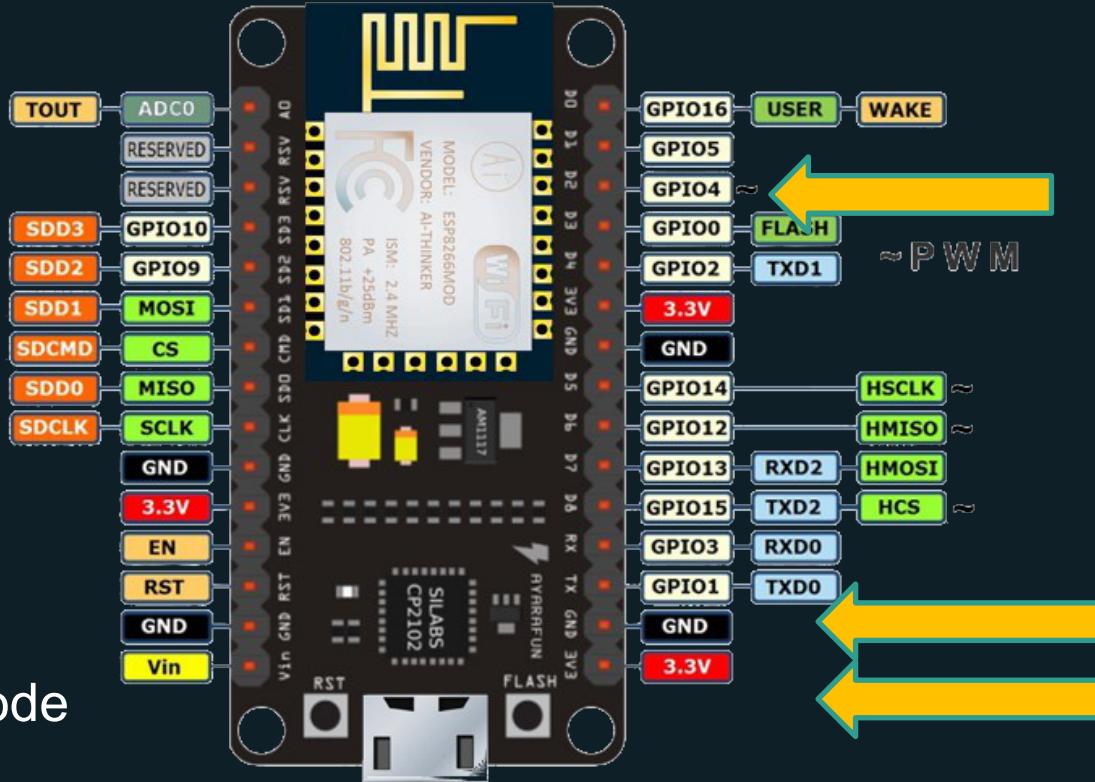
ESP8266 and DHT11 wiring

PIN on the esp8266

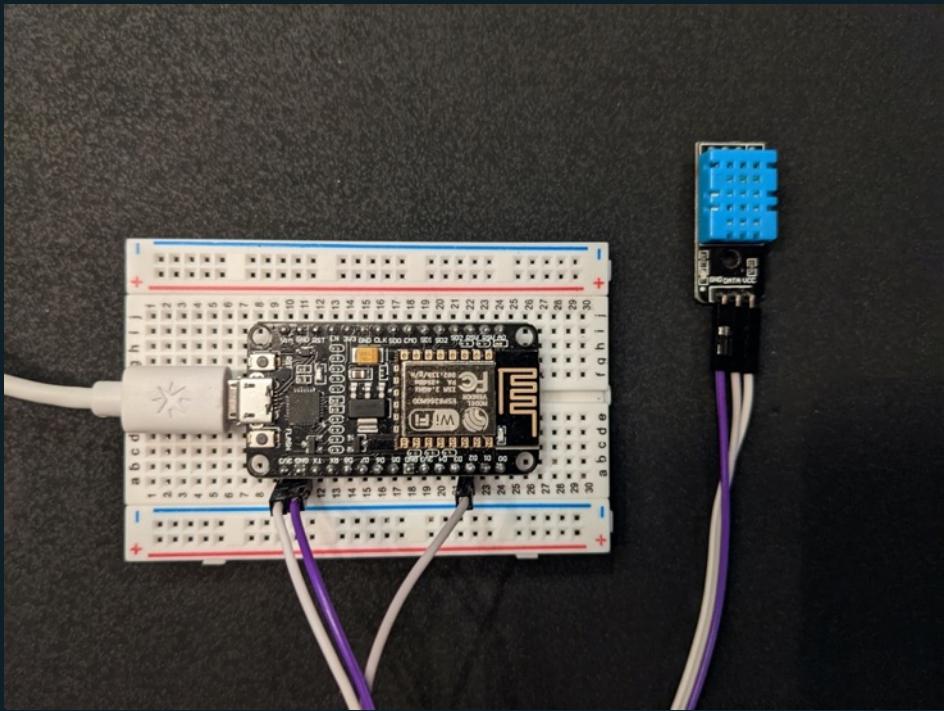
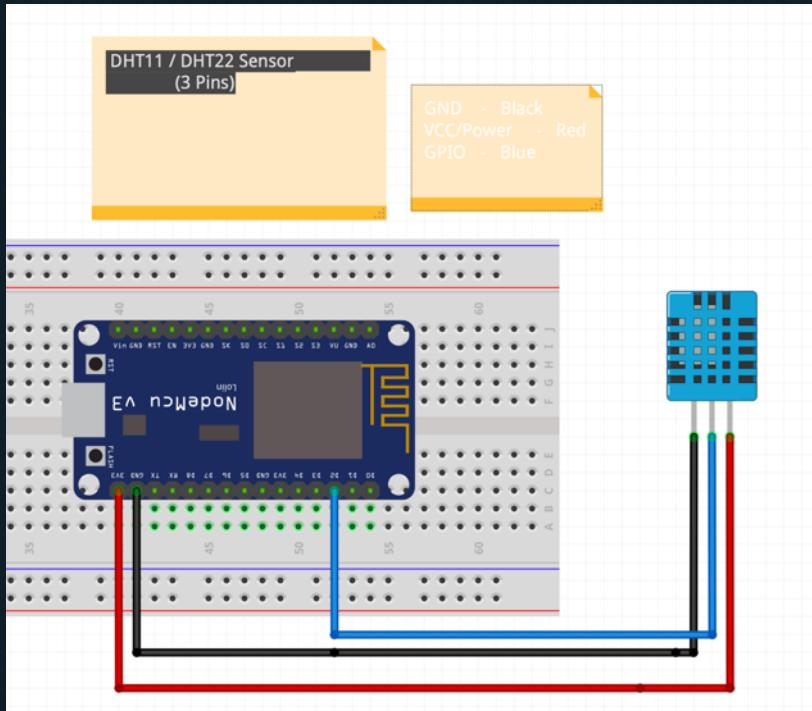
NOTE: we will be using
The GPIO numbers
NOT the PINs

Like Pin D2 = GPIO4

We use number 4 in our code



ESP8266 & DHT11 Wiring





Code for esp8266 with dht11 - Number one

All code and instruction can be found here

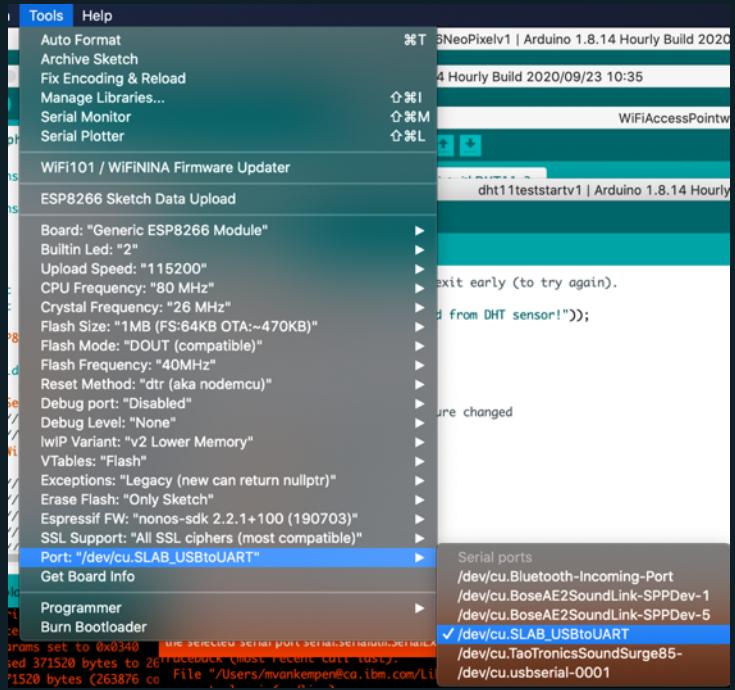
<https://github.com/markusvankempen/esp8266andDHT11>

Here our 1st code to upload to the esp8266 which is connected to DH11

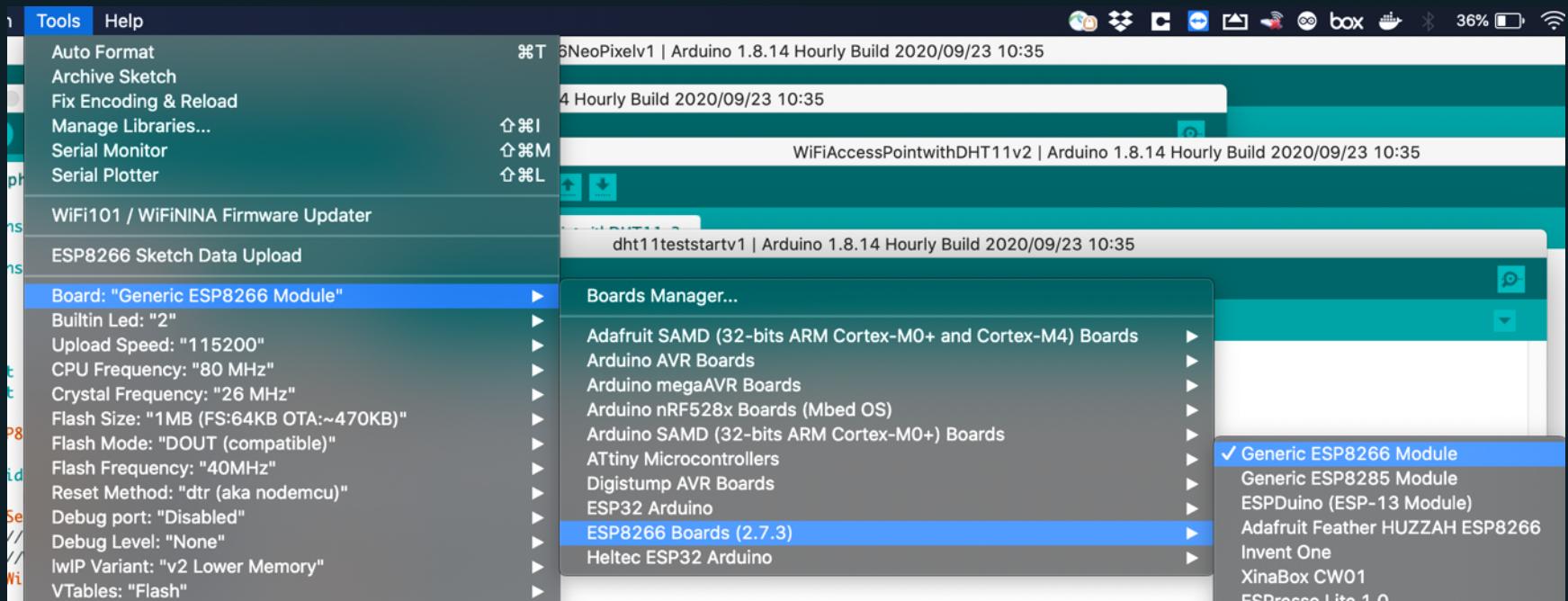
Note:

- Make sure u select the right board
- Copy the code into your IDE make sure to select the right serial Port

Serial Port Selection



Board Selection



Upload the code to esp8266

Code :

<https://raw.githubusercontent.com/markusvankempen/esp8266andDHT11/main/code/esp-dht11-1.ino>

Once you upload and open the serial terminal /

make sure the Baudrate is 115200

The screenshot shows the Arduino IDE interface. The top menu bar includes File, Sketch, Tools, and Help. Below the menu is a toolbar with icons for file operations. The central workspace has two main sections: the Serial Monitor on the left and the code editor on the right.

Serial Monitor: The title bar says "SNP-2020-ESP8266-DHT11-EnviromentMonitoring-v2" and the port is set to "/dev/cu.SLAB_USBtoUART". The monitor displays a continuous stream of log messages from the ESP8266. The messages show the DHT sensor reading temperature and humidity at regular intervals. The log output is as follows:

```
14:37:56.640 -> readTemperature in C = 26.20
14:37:56.640 -> readHumidity in % = 36.00
14:37:56.640 -> readTemperature in C = 26.20
14:37:56.640 -> readHumidity in % = 37.00
14:37:56.640 -> readTemperature in C = 26.20
14:37:56.640 -> readHumidity in % = 37.00
14:37:56.640 -> readTemperature in C = 26.20
14:37:56.640 -> readHumidity in % = 36.00
14:38:01.276 -> readTemperature in C = 26.30
14:38:01.276 -> readHumidity in % = 36.00
14:38:01.276 -> readTemperature in C = 26.30
14:38:01.276 -> readHumidity in % = 37.00
14:38:01.276 -> readTemperature in C = 26.30
14:38:01.276 -> readHumidity in % = 37.00
14:38:02.892 -> readTemperature in C = 26.30
14:38:02.892 -> readHumidity in % = 37.00
14:38:04.883 -> readTemperature in C = 26.30
14:38:04.919 -> readHumidity in % = 37.00
14:38:06.939 -> readTemperature in C = 26.30
14:38:06.939 -> readHumidity in % = 38.00
14:38:08.954 -> readTemperature in C = 26.30
14:38:08.954 -> readHumidity in % = 38.00
14:38:10.984 -> readTemperature in C = 26.30
14:38:10.984 -> readHumidity in % = 38.00
```

Code Editor: The title bar says "esp8266-dht11-1 | Arduino 1.8". The code is written in C++ and defines a DHT object and prints its readings to the serial port. The code includes comments explaining the sensor's behavior and how to handle failed reads.

```
#include <DHT.h>
#include <SoftwareSerial.h>

// Set the pins for the DHT sensor
#define DHTPIN 2
#define DHTTYPE DHT11

// Create a DHT object
DHT dht(DHTPIN, DHTTYPE);

void setup() {
    // Initialize serial communication
    Serial.begin(115200);
}

void loop() {
    // Wait a few seconds between measurements
    delay(2000);

    // Reading temperature or humidity takes about 250 milliseconds!
    // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
    float h = dht.readHumidity();
    // Read temperature as Celsius (the default)
    pt = t;
    t = dht.readTemperature();

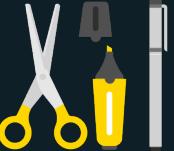
    // Check if any reads failed and exit early (to try again).
    if (isnan(h) || isnan(t)) {
        Serial.println(F("Failed to read from DHT sensor!"));
        return; //restart
    }

    // Print the results to the serial port
    Serial.print("readTemperature in C = ");
    Serial.print(t);
    Serial.print("readHumidity in % = ");
    Serial.print(h);
}
```

The status bar at the bottom right indicates "Done uploading."



Exercise #1 ... display temperature and Humidity in the Arduino IDE plotter



<https://arduinogetstarted.com/tutorials/arduino-serial-plotter>

Format is Label:Value Space Label: Value Space

```
Serial.print("readTemperature:"+String(t));
```

```
Serial.println(" readHumidity:"+String(h));
```

The screenshot shows the Arduino IDE interface. The top menu bar includes 'Arduino', 'File', 'Edit', 'Sketch', 'Tools' (which is currently selected), and 'Help'. Below the menu bar is a toolbar with icons for AutoSave, Home, Copy, Paste, and Undo/Redo. The main workspace displays a sketch titled 'esp8266-dht11-1'. The code in the sketch reads temperature and humidity from a DHT11 sensor connected to pins 2 and 3 respectively, and prints them to the serial monitor. The 'Tools' menu is open, showing options like Auto Format, Archive Sketch, Fix Encoding & Reload, Manage Libraries..., Serial Monitor, and Serial Plotter, with 'Serial Plotter' being the selected item. The bottom right corner of the screen shows the text 'ESP8266 Sketch Data Upload'.

Done Saving.

Call us. 1-800-555-1234

Configuring flash size...
Auto-detected Flash size: 1MB

Auto-detected Flash size: 4MB
Flash parameters set to 0x0310



Exercise #2 ... print temperature only if previous temperature has changes

```
//  
//  
// if(pt != t) // Check if temperature changed  
//{  
//   Serial.println("Same temp");  
// ...
```



2nd Chapter (two)

Web Server



2nd Chapter (two)



Create a webserver and Display humidity and temperature and other data

ESP8266 Web Server

GPIO16 LED - State on

OFF

No Temperature Change

Temperature = 22.50

Humidity = 34.00

getEpochTime = 1608120384

Time = 12:06:24

msgcounter = 24

Temperature 22.50 C

Humidity 34.00 %

Refresh

mqttmsg = {"hello": "mvk01"}



Esp8266 and Wifi

Note:

The esp8266 and works as wifi access point – meaning you have to join the esp8266 wifi network to access a webpage

Or

The esp8266 can join your Wifi network which is what we will be using to display the humidity etc on a web pages

Esp8266 WebServer



In the code you need to adjust the SSID / password to your wifi credentials as well as personalize some other parameters – Note: 2.4 G wifi only

```
////>>>>>>> CHANGE HERE
// Replace with YOUR network credentials
const char* ssid    = "YOURSSID"; //>>>>>>> CHANGE HERE
const char* password = "YOURPASSWORD"; //>>>>>>> CHANGE HERE
String DEVICEID   = "markus01"; //>>>>>>> CHANGE HERE

// find ur LAT LON use https://www.latlong.net/ and ur cityname like Brantford,canada for Branford =43.139410,-80.263650

String LATITUDE = "43.6711581" ; //>>>>>>> CHANGE HERE
String LONGITUDE = "-79.4129989" ; //>>>>>>> CHANGE HERE
```

The Code and Deployment

Code can be found here

<https://raw.githubusercontent.com/markusvankempen/esp8266andDHT11/main/2nd-Chapter/2-esp8266-webserver-with-dht11-c2b.ino>

Note: If you can not upload the code via serial port maybe blocked just hit the rest button or unplug the device and select the serial port in the IDE again.



Upload the code and check the serial monitor to message

Serial Monitor should show

13:05:26.861 -> WiFi connected.

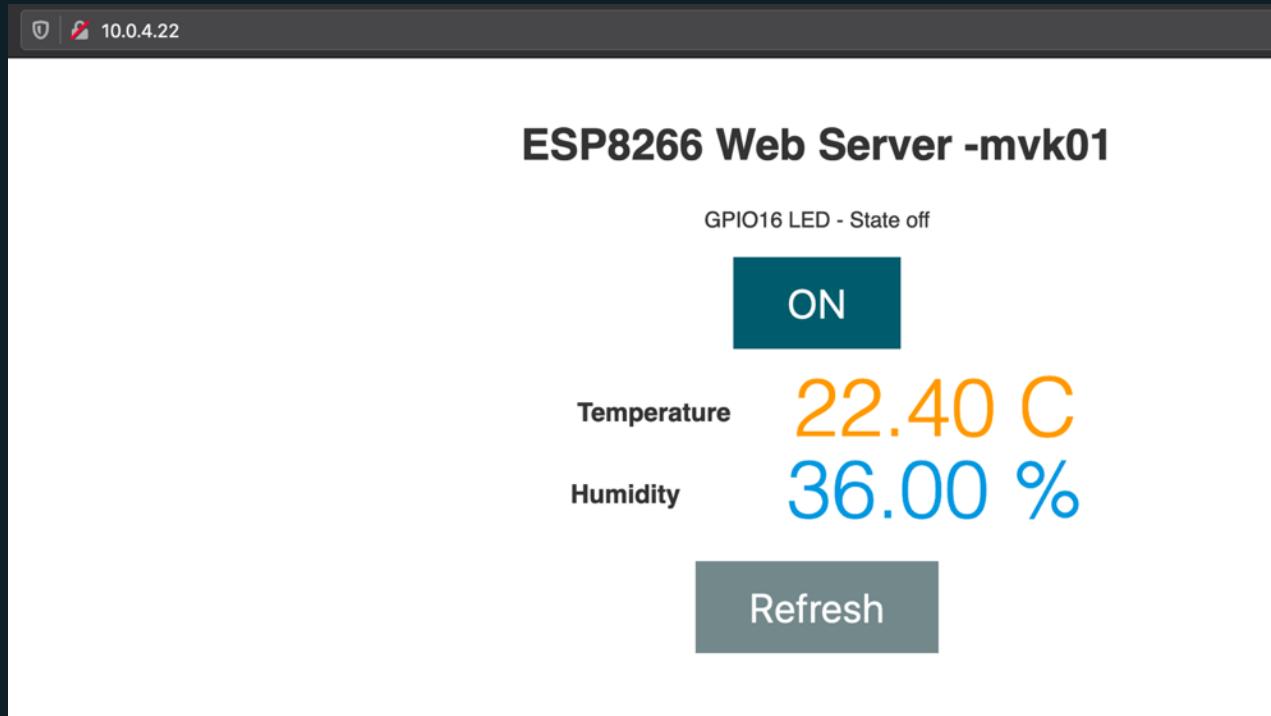
13:05:26.861 -> IP address:

13:05:26.861 -> 192.168.1.26 <<< Your IP address should be different

Use the ip address and open it in a browser like **<http://192.168.1.26/>**

Display play the WebPage of the esp8266

<http://your-ip-address> - you should see





Exercise:

- Display the Voltage Parameter
- Display a message if the Temperature changes

ESP8266 Web Server - YOURNAME

GPIO16 LED - State off

ON

prevTemp = 25.10

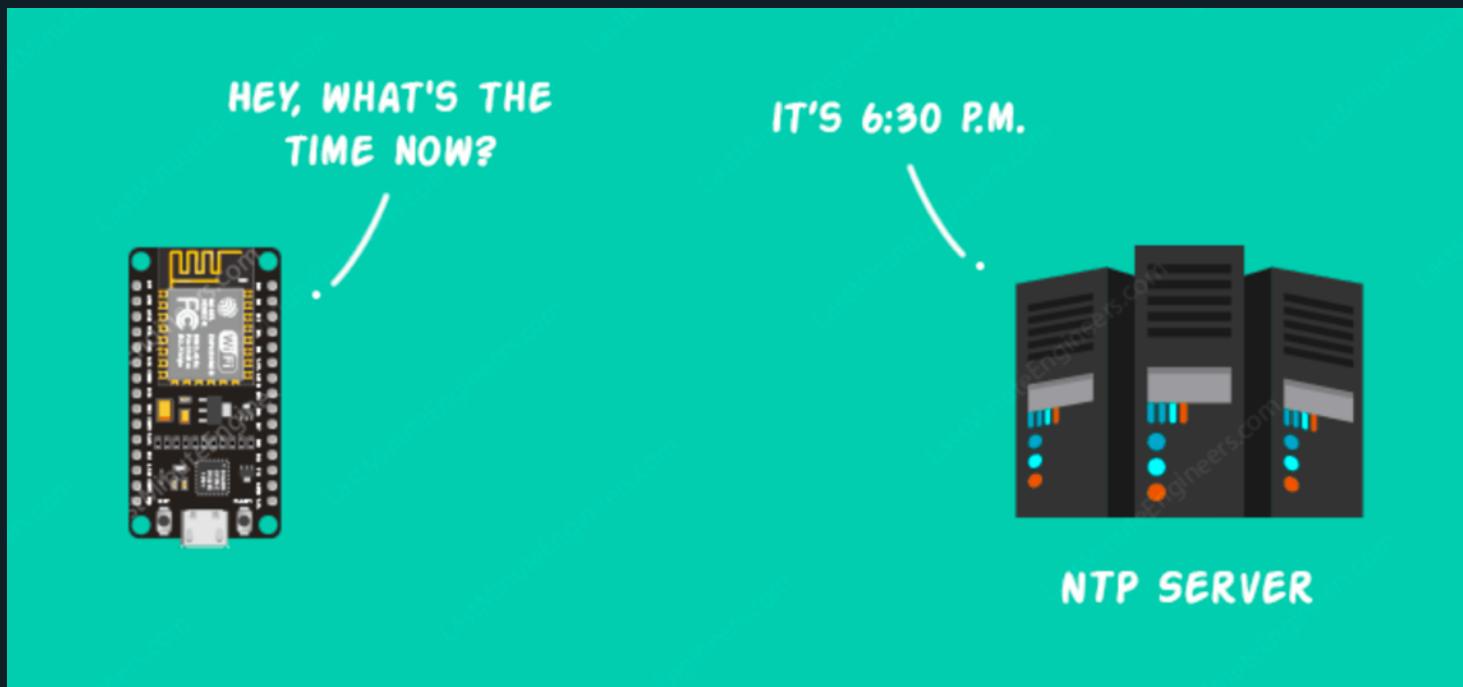
Temperature Changed from 25.10

Temperature	25.30 C
Humidity	24.00 %
Voltage	2.89 V

Refresh

IBM Watson IoT 31

3rd CHAPTER (three)



3rd CHAPTER (three)

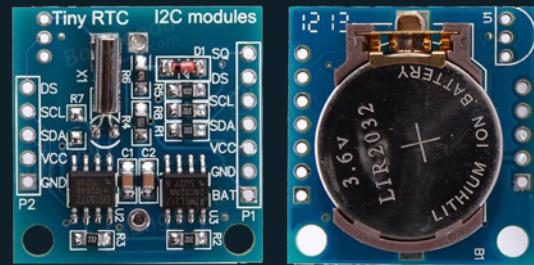


The esp8266 has no RealTimeClock aka RTC so we need to connect to a NTP(**Network Time Protocol**) Server to get the current time.

We need to load the NTPClient Library into the IDE

You can use the example code from the library for testing

File>Examples>NTPClient-Basic

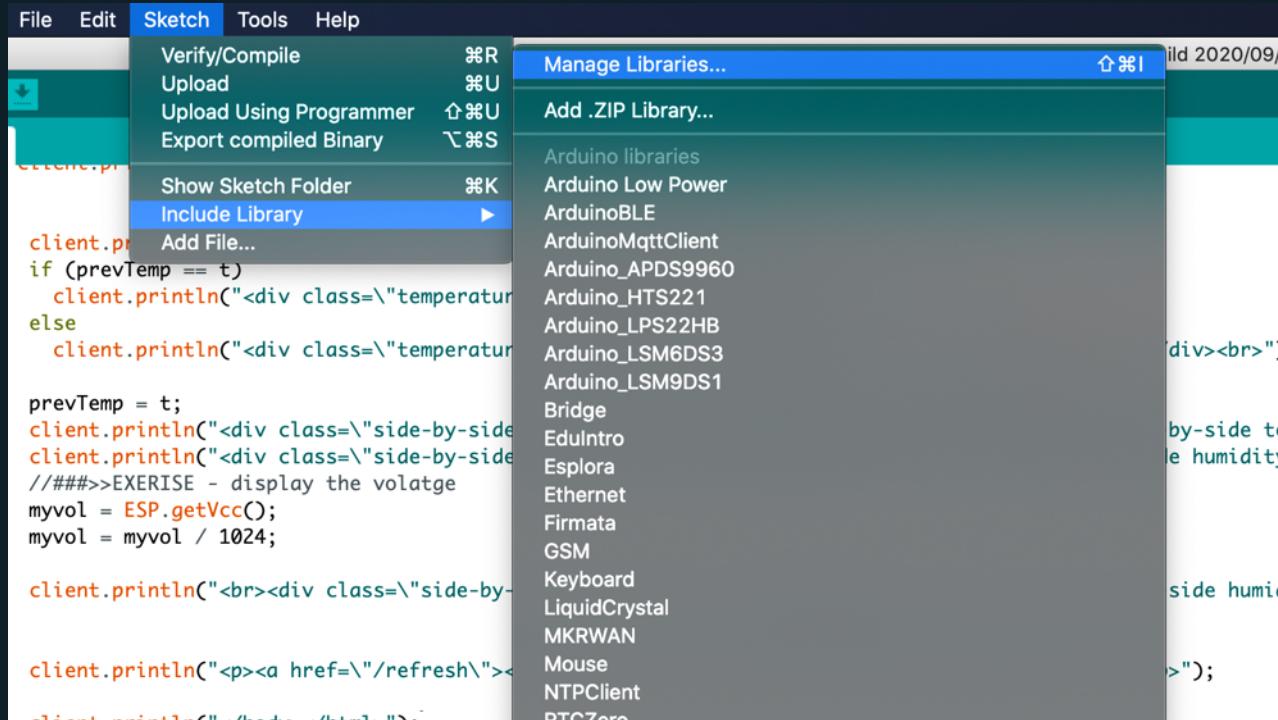


Example of RTC IBM Watson IoT 33



Download and Import the Library

<https://randomnerdtutorials.com/esp32-ntp-client-date-time-arduino-ide/>



Test NTP Client via the example code



The screenshot shows the Arduino IDE interface with the 'File' menu open, specifically the 'Examples' submenu. The submenu contains 11 items numbered 01 to 11, each representing a different example sketch. The 'NTPClient' example is currently selected, as indicated by a blue highlight bar at the bottom of the list. To the left of the menu, a portion of the code for the 'NTPClient' example is visible, showing the setup and loop functions. The code includes connections to an Adafruit Feather M0 WiFi module and an LCD display, and it demonstrates how to retrieve and display time information from an NTP server.

- New ⌘N
- Open... ⌘O
- Open Recent ▶
- Sketchbook ▶
- Examples ▶
- Close ⌘W
- Save ⌘S
- Save As... ⌘⌘S
- Page Setup ⌘P
- Print ⌘P

client.println("<p>prev
if (prevTemp == t)
 client.println("<div
else
 client.println("<div

prevTemp = t;
client.println("<div class="list-item"
client.println("<div class="list-item"
//###>EXERISE - display
= ESP.getVCCo;
=myvol /1000;

client.println("
<div class="list-item"

client.println("<p><a href="list-item"
client.println("</body>"list-item"

Built-in Examples

- 01.Basics
- 02.Digital
- 03.Analog
- 04.Communication
- 05.Control
- 06.Sensors
- 07.Display
- 08.Strings
- 09.USB
- 10.StarterKit_BasicKit
- 11.ArduinoISP

Examples for any board

- Adafruit Circuit Playground
- Arduino_APDS9960
- Arduino_HTS221
- Arduino_LPS22HB
- Arduino_LSM6DS3
- Arduino_LSM9DS1
- ArduinoMqttClient
- Bridge
- Edulintro
- Ethernet
- Firmata
- LiquidCrystal
- NTPClient
- SD
- Stepper

fiWebServerWithDH

oniz>>off<button

/div>
");
"+(String)prevTem

ure</div><div class="list-item"
iv><div class="list-item"
"list-item">Refresh<button

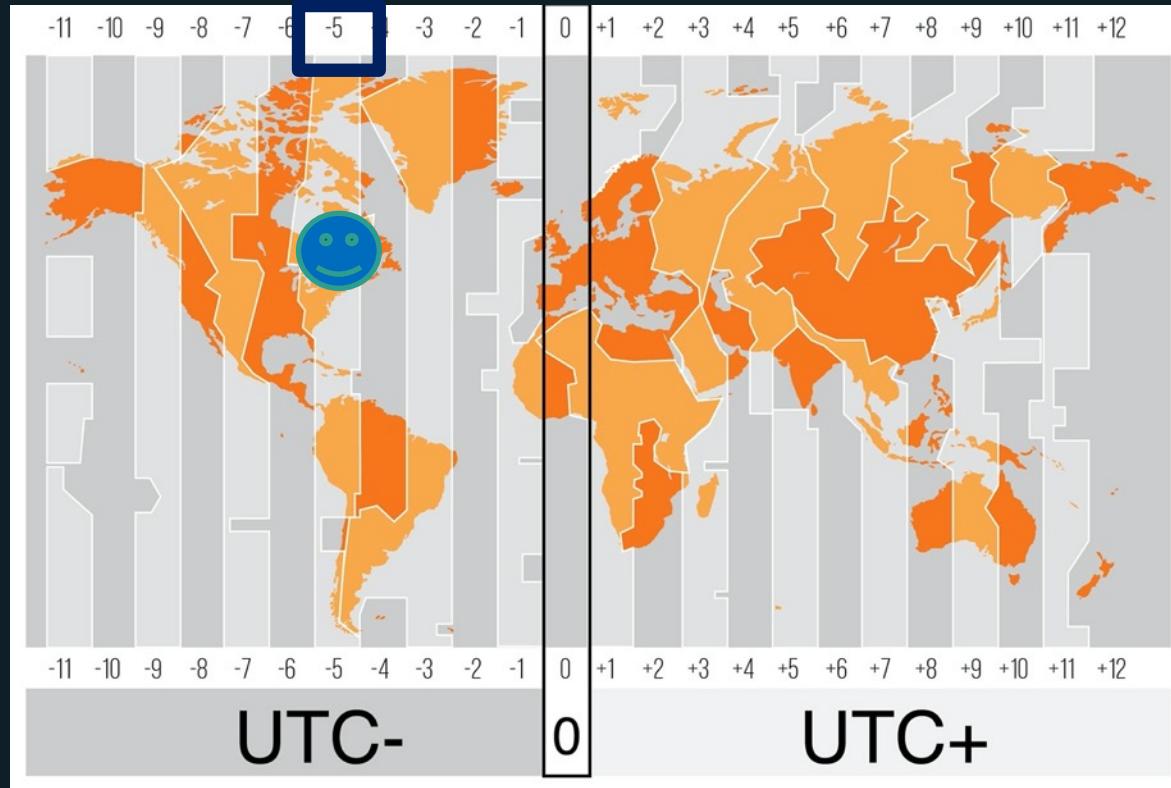
iv><div class="list-item"
"list-item">Advanced

Basic

The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** Arduino
- Serial Monitor Header:** /dev/cu.SLAB_USBtoUART
- Code Area:** The code editor displays a sketch named "Basic". It includes #include statements for NTPClient.h, ESP8266WiFi.h, WiFi.h, WiFiUdp.h, and WiFiClient.h. The setup() function initializes the serial port at 115200 bps, sets up WiFi, and starts a WiFiUDP object. The loop() function checks WiFi status, updates the time client, prints the formatted time to the serial port, and delays for 1000ms. A timestamp column on the right indicates the time each line was printed.
- Serial Monitor:** Shows the output of the code execution, including the formatted time: 14:36:05.077, 14:36:11.263, 14:36:12.253, 14:36:13.269, 14:36:14.254, 14:36:15.245, 14:36:16.247, 14:36:17.256, 14:36:18.271, 14:36:19.278, 14:36:20.277, 14:36:21.246, 14:36:22.265, 14:36:23.250, 14:36:24.275, 14:36:25.262.
- Status Bar:** Both NL & CR, indicating line endings.

UTC/GMT vs EST



NTP service



The NTP service will return the GMT/UTC time which is 5 Hour before EST
We will need to adjust the time using an offset

Detailed information here

<https://randomnerdtutorials.com/esp32-ntp-client-date-time-arduino-ide/>

Adjust the code with the offset time in seconds e.g $5*60*60 = -18000$

```
NTPCClient timeClient(ntpUDP, "pool.ntp.org", -18000, 60000); //with EST adjustment
```

EST Timezone time

```
NTPClient timeClient(ntpUDP, "pool.ntp.org", -18000, 60000);
```

The screenshot shows the Arduino IDE interface. On the left, the code for an NTP client is displayed:

```
10
11 WiFiUDP ntpUDP;
12 NTPClient timeClient(ntpUDP, "pool.ntp.org", -18000, 60000);
13
14 void setup(){
15   Serial.begin(115200);
16
17   WiFi.begin(ssid, password);
18
19   while ( WiFi.status() != WL_CONNECTED ) {
20     delay ( 500 );
21     Serial.print ( "." );
22   }
23
24   timeClient.begin();
25 }
26
27 void loop() {
28   timeClient.update();
29
30   Serial.println(timeClient.getFormattedTime());
31
32   delay(1000);
33 }
```

On the right, the Serial Monitor window displays the received time information:

```
.....14:47:57
14:47:58.054 -> 14:47:58
14:47:59.081 -> 14:47:59
14:48:00.056 -> 14:48:00
14:48:01.055 -> 14:48:01
14:48:02.088 -> 14:48:02
14:48:03.062 -> 14:48:03
14:48:04.083 -> 14:48:04
```

At the bottom of the Serial Monitor window, there are two checkboxes: Autoscroll and Show timestamp.

Exercise

Add NTP code to the WebServer and display the time
On the WebPage



ESP8266 Web Server - mvk01

GPIO16 LED - State on

OFF

15:14:11

prevTemp = 22.40

NO Temperature Change

Temperature 22.40 C
Humidity 34.00 %
Voltage 2.96 V

Refresh

Code here

<https://raw.githubusercontent.com/markusvankempen/esp8266andDHT11/main/3rd%20Chapter/3-WifiWebServerWithDHT11AndNTP.ino>



Agenda

- 4th CHAPTER (four) – MQTT messaging
 - deep sleep / HW reset
- 5th CHAPTER – Real time Visualization
- 6th CAPTER (6) – Saving the data to an SQL database
- 7th CHAPTER (7) - Visualizing Time series Data



ESP8266 Web Server

GPIO16 LED - State on

OFF

No Temperature Change

Temperature = 22.50

Humidity = 34.00

getEpochTime = 1608120384

Time = 12:06:24

msgcounter = 24

Temperature

22.50 C

Humidity

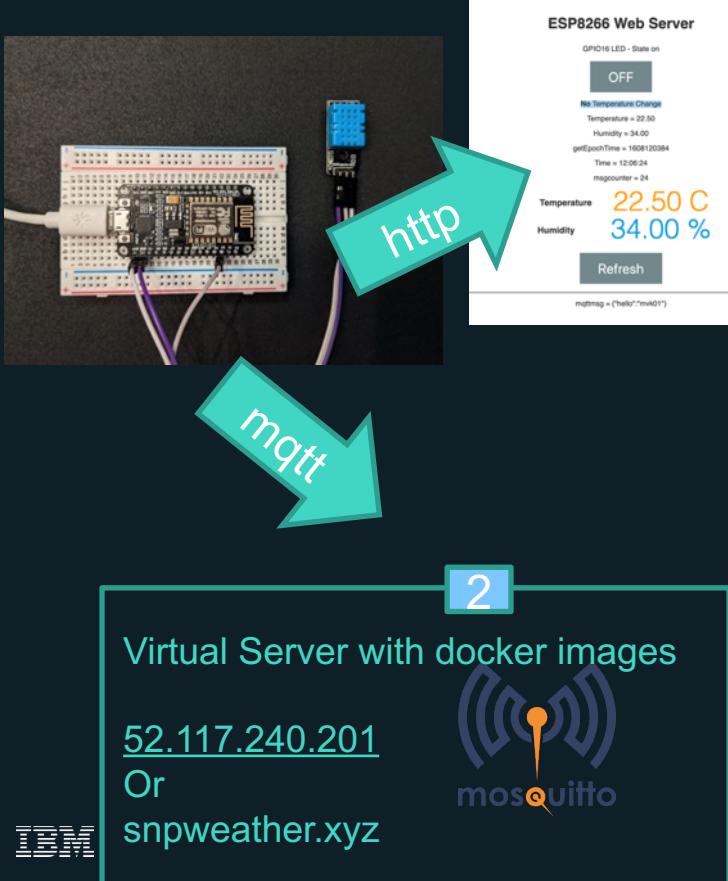
34.00 %

Refresh

mqttmsg = {"hello": "mvk01"}

Recap

4th CHAPTER (four)



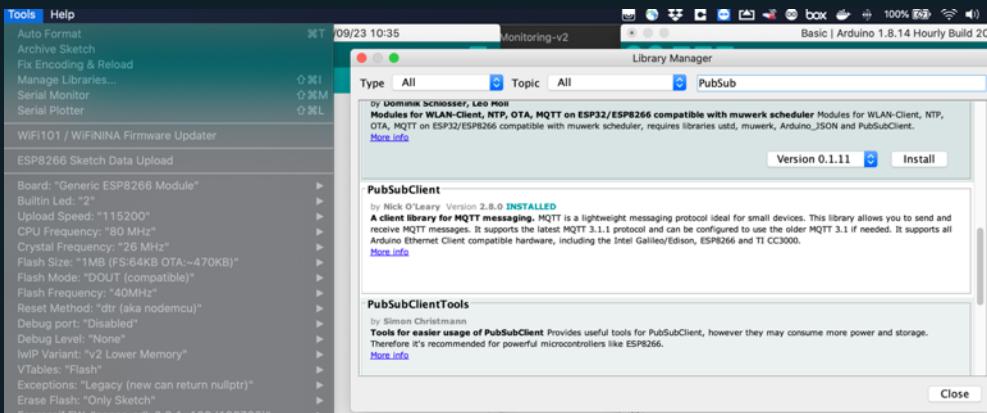
- Send Data to external Server
- Use the DeepSleep Mode
- Use the external reset

4th CHAPTER (four)



In this section we will add a mqtt client to our WebServer and send a json message with temperature, voltage etc to a mqtt broker.

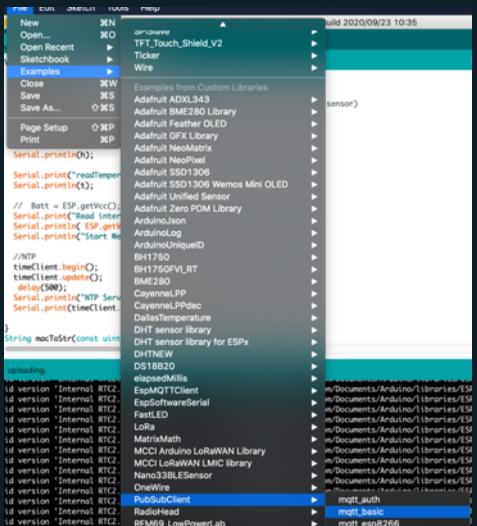
We will need to add a mqtt client library called PubSubClient



MQTT Client and Message Publishing

You can use the mqtt client example to send a test message to our mqtt broker (which is a docker on our virtual Server) once you add the library to the IDE. The Library is called PubSubClient.h

The example Code is under Examples – choose esp8266 example

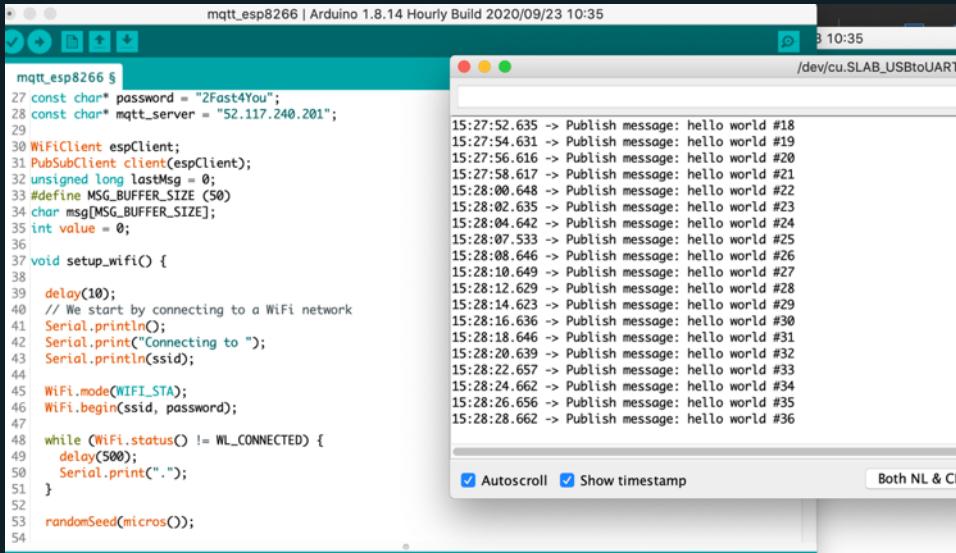


Test MQTT Message Publishing

Add the our Mqtt broker IP

```
const char* mqtt_server = "52.117.240.201";
```

And the topic or message



The image shows a screenshot of an Arduino IDE and a terminal window. The Arduino IDE window displays the following code:

```
mqtt_esp8266 §
27 const char* password = "2Fast4You";
28 const char* mqtt_server = "52.117.240.201";
29
30 WiFiClient espClient;
31 PubSubClient client(espClient);
32 unsigned long lastMsg = 0;
33 #define MSG_BUFFER_SIZE (50)
34 char msg[MSG_BUFFER_SIZE];
35 int value = 0;
36
37 void setup_wifi() {
38
39   delay(10);
40   // We start by connecting to a WiFi network
41   Serial.println();
42   Serial.print("Connecting to ");
43   Serial.println(ssid);
44
45   WiFi.mode(WIFI_STA);
46   WiFi.begin(ssid, password);
47
48   while (WiFi.status() != WL_CONNECTED) {
49     delay(500);
50     Serial.print(".");
51   }
52
53   randomSeed(micros());
54 }
```

The terminal window to the right shows a series of publish messages being sent at regular intervals:

```
15:27:52.635 -> Publish message: hello world #18
15:27:54.631 -> Publish message: hello world #19
15:27:56.616 -> Publish message: hello world #20
15:27:58.617 -> Publish message: hello world #21
15:28:00.648 -> Publish message: hello world #22
15:28:02.635 -> Publish message: hello world #23
15:28:04.642 -> Publish message: hello world #24
15:28:07.533 -> Publish message: hello world #25
15:28:08.646 -> Publish message: hello world #26
15:28:10.649 -> Publish message: hello world #27
15:28:12.629 -> Publish message: hello world #28
15:28:14.623 -> Publish message: hello world #29
15:28:16.636 -> Publish message: hello world #30
15:28:18.646 -> Publish message: hello world #31
15:28:20.639 -> Publish message: hello world #32
15:28:22.657 -> Publish message: hello world #33
15:28:24.662 -> Publish message: hello world #34
15:28:26.656 -> Publish message: hello world #35
15:28:28.662 -> Publish message: hello world #36
```

At the bottom of the terminal window, there are checkboxes for "Autoscroll" and "Show timestamp", and a button for "Both NL & CR".

Example code also here:

<https://raw.githubusercontent.com/markusvankempen/esp8266andDHT11/main/4th%20Chapter/4-mqtt-esp8266-test.ino>



Check if message are received by the mqtt broker

You can check if mqtt messages are received on the server using a mosquito client. U can install mosquito client locally or use the server via ssh.

Example:

ssh snpuser@52.117.240.201

Or snpuser@snpweather.xyz

Example receiving on server from sensor

mosquitto_sub -v -t "sensor/YOURDEVICEID/#"

Example

Publishing on server to sensor

mosquitto_pub -h localhost

-t sensor/YOURNAME/command

-m "{\"value1\":20,\"value2\":40}"

The screenshot shows a terminal window with two panes. The left pane displays the command-line interface for publishing and subscribing to MQTT topics. The right pane shows the MQTT broker's log output, which includes a series of publish messages from the client and a message arrived event from a sensor topic.

```
52:02.466 -> Publish message: hello world #55
52:02.466 -> Publish message: hello world #56
52:02.466 -> Publish message: hello world #57
52:02.466 -> Publish message: hello world #58
52:02.466 -> Publish message: hello world #59
52:05.312 -> Publish message: hello world #60
52:07.521 -> Publish message: hello world #61
52:08.249 -> Publish message: hello world #62
52:10.249 -> Publish message: hello world #63
52:12.247 -> Publish message: hello world #64
52:14.265 -> Publish message: hello world #65
52:16.254 -> Publish message: hello world #66
52:18.264 -> Publish message: hello world #67
52:20.231 -> Message arrived [sensor/YOURNAME/command] {"value1":20,"value2":40}
52:20.268 -> Publish message: hello world #68
52:22.238 -> Publish message: hello world #69
52:24.270 -> Publish message: hello world #70
52:26.253 -> Publish message: hello world #71
52:28.239 -> Publish message: hello world #72
52:30.276 -> Publish message: hello world #73
52:32.273 -> Publish message: hello world #74
52:34.271 -> Publish message: hello world #75
52:36.248 -> Publish message: hello world #76
52:38.244 -> Publish message: hello world #77
snip install snpuser@snpserver01:~$ mosquitto_sub -t '#' -h 52.117.240.201.
apt install Unable to connect (Lookup error.).
See 'snap i snpuser@snpserver01:~$ mosquitto_sub -t '#' -h 52.117.240.201.
root@snpser reading pac Unable to connect (Lookup error.).
root@snpser Building de {"hello":"mvk01"}
reading sta {"sensor":"gps","time":1351824120,"data":[48.756080,2.3
The followi e" : -79.4129989}
libc-ares hello world #1511
The followi libc-ares hello world #1512
libc-ares ) upgraded, hello world #1513
need to get hello world #1514
After this hello world #1515
Do you want hello world #1516
Set:1 http: hello world #1517
Set:2 http: hello world #1518
```

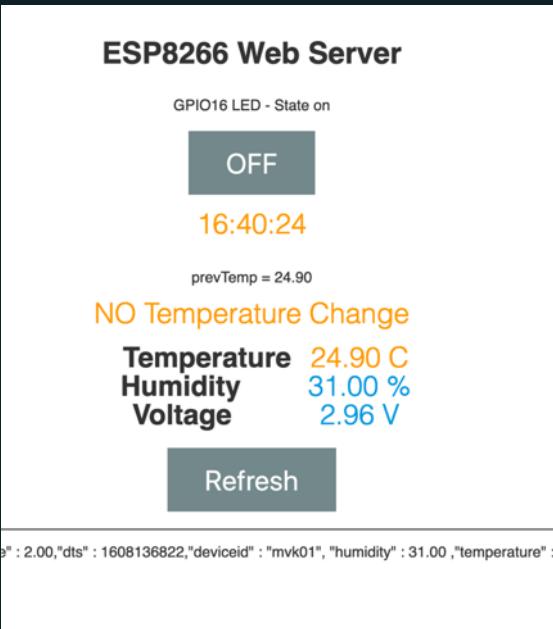


Add the mqtt client code to our WebServer

The code for the webserver,ntp and mqtt client is here

Code is here:

<https://raw.githubusercontent.com/markusvankempen/esp8266andDHT11/main/4th%20Chapter/4-WifiWebServerWithDHT11NTPandMqtt-c4a.ino>



WebServer With NTP and MQTT client

The WebServer is now getting the current time/date and send mqtt with temperature , humidity etc. as json to the broker

:31:08.584 -> Sending Sensor Data

```
16:31:08.584 -> {"msgcnt" : 59,"mac" : "ec:fa:bc:c2:fa:76","ip" :  
"10.0.4.22","voltage" : 2.00,"dts" : 1608136267,"deviceid" : "mvk01",  
"humidity" : 36.00 , "temperature" : 22.80 , "latitude"  
:43.6711581,"logitude" : -79.4129989}
```

Exercise



Add the message counter (see field msgcounter) to the json message and validate the message is received by the broker.

```
\"msg\" : "+String(msgcounter)+ "
```

Exercise – Enable/Test Deep sleep (optional)



Our device are eventually powered by batteries/Solarpanels and therefore we want to conserve energy and shutdown/ go into deepsleep if there is nothing to do. We also only want to send if the has changed.

Enable the code deepsleep code and check if the wifi connection is available on wake up!



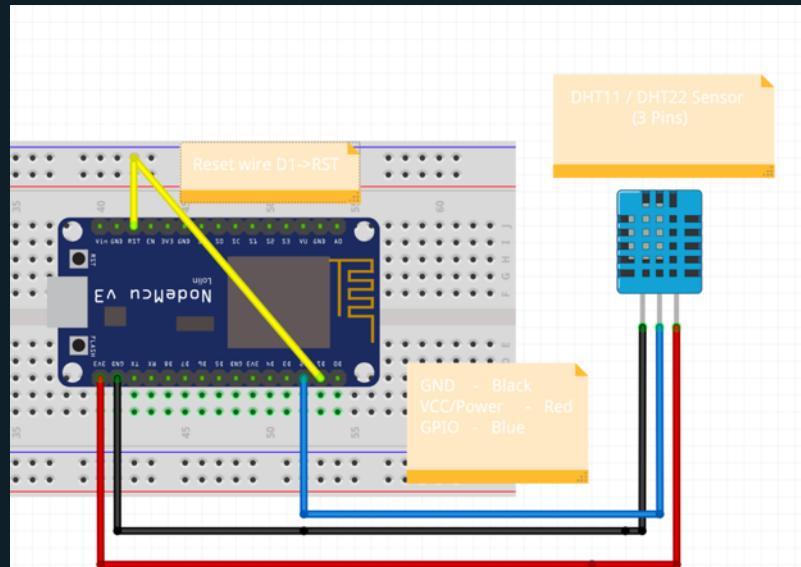
Exercise – Soft vs Hard reset (optional)

You can force a reboot via `ESP.reset()`/`ESP.reboot()`.

You can also use the Digital pin to enable Hard reset like via `D1->RST`
`pinMode(D1, OUTPUT);`

`digitalWrite(D1, HIGH); //D1 -> RESET PIN via a wire`

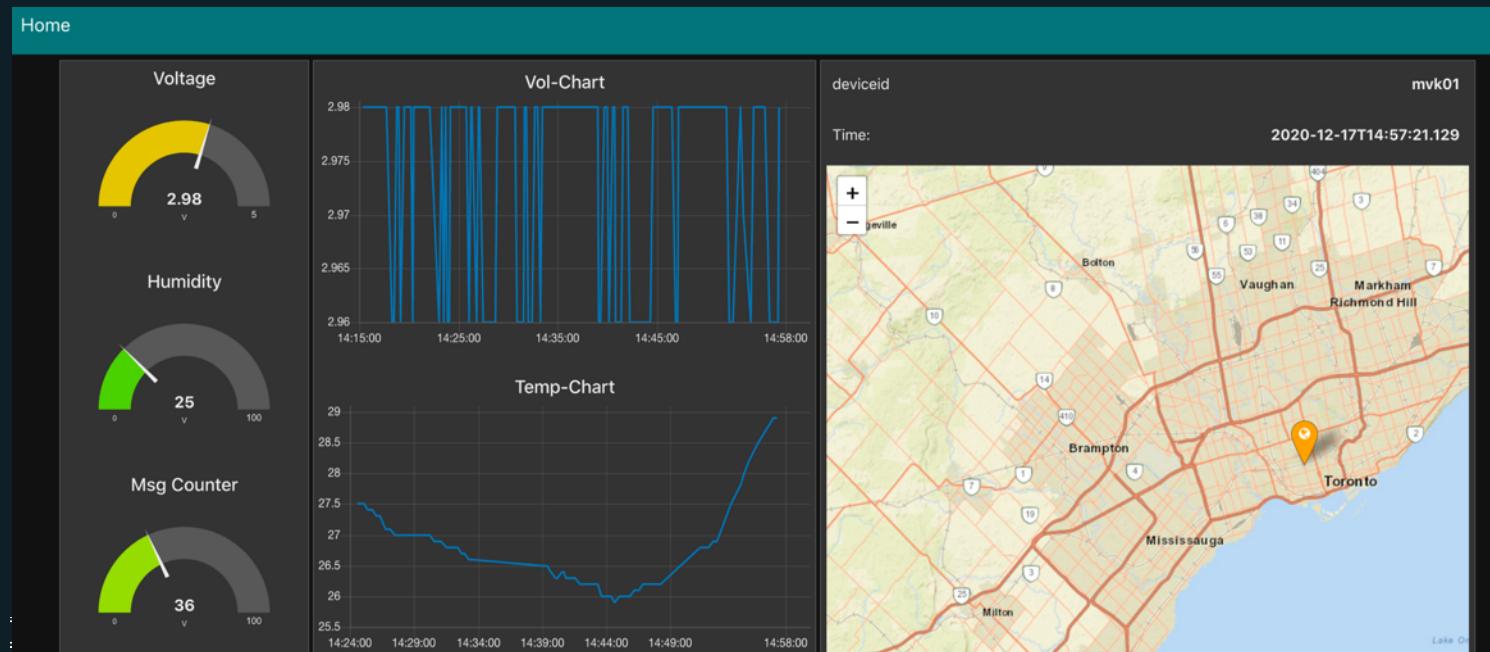
`digitalWrite(D1, LOW); // =>RESET`





5th CHAPTER – Real time Visualization

Now we have the mqtt message flowing to the server id would be nice to see visualize the data in a real time dash board



Import missing plugins

Like

node-red-contrib-web-worldmap

node-red-dashboard

Via Menu -> Manage Pallet



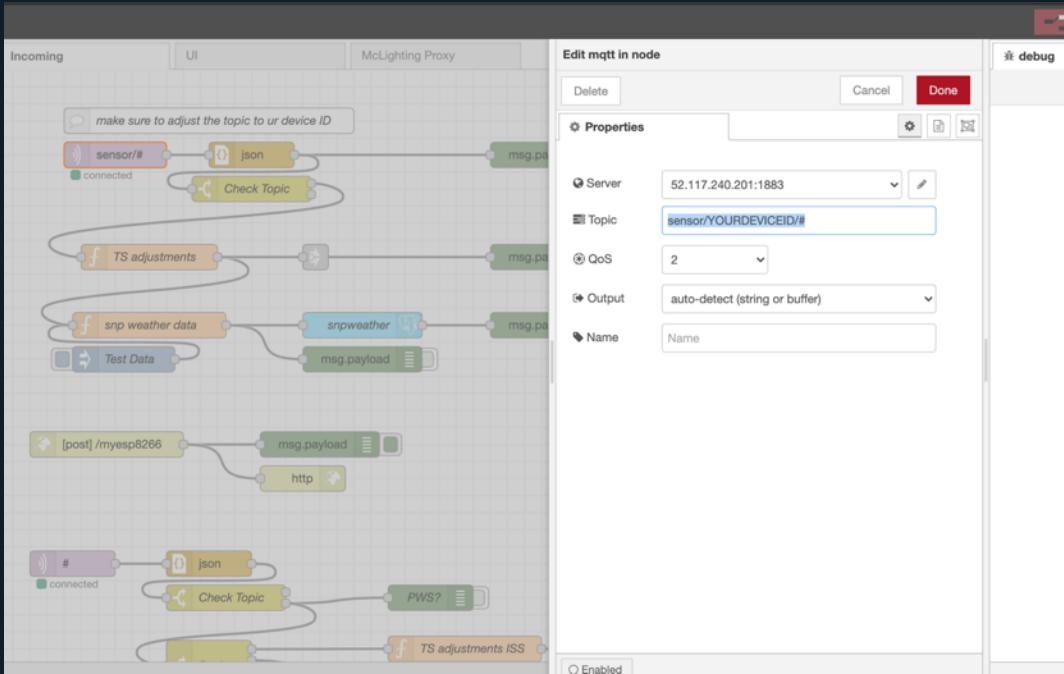
Real Time Dashboard

Choose one of Node-RED instance <http://52.117.240.201:1000> to
<http://52.117.240.201:1007>

- Import the node-red flow from here
- <https://raw.githubusercontent.com/markusvankempen/esp8266andDHT11/main/5th%20Chapter/node-red.flow>
- Adjust the deviceid in the mqtt topic to your deviceid
- Add the missing humidity charts and other message data to the dashboard
- Map all device from all students on your map
- You can access the

Node-RED flow

Make sure to adjust the Topic with Your Device ID



6th CAPTER (6) – Saving the data to an SQL database



The server has a posgres sql database docker instance which we can use to store the sensor data.

The admin tool is on port 80 and is called pgAdmin

<http://52.117.240.201/browser/>

Time series database

The database “snpweather” is already created and matches the json mqtt message format. We will walk you thru the admin tool and show you the fields and existing data.

The screenshot shows the IBM Db2 Admin tool interface. The left sidebar displays the database schema structure under the 'Tables (1)' section, specifically the 'snpweather' table which contains 14 columns: location, name, temperature, humidity, deviceid, latitude, longitude, timestamp, dts, voltage, ip, mac, tsutc, msg, and b. The 'Query Editor' tab is active, showing a query that selects various fields from the 'public.snpweather' table where the 'msg' field is not equal to 0. The results of this query are displayed in a table below, showing multiple rows of data with columns for latitude, longitude, timestamp, dts, voltage, ip, mac, tsutc, and b.

latitude	longitude	timestamp	dts	voltage	ip	mac	tsutc	b
43.6711	-79.4129	2020-12-17 15:13:38.639	2020-12-17 15:13:37	2.98	10.0.4...	ecfab...	2020-12-17 20:13:38.639	
43.6711	-79.4129	2020-12-17 15:13:48.903	2020-12-17 15:13:48	2.98	10.0.4...	ecfab...	2020-12-17 20:13:48.903	
43.6711	-79.4129	2020-12-17 15:13:59.153	2020-12-17 15:13:58	2.98	10.0.4...	ecfab...	2020-12-17 20:13:59.153	
43.6711	-79.4129	2020-12-17 15:14:09.454	2020-12-17 15:14:08	2.98	10.0.4...	ecfab...	2020-12-17 20:14:09.454	
43.6711	-79.4129	2020-12-17 15:14:19.697	2020-12-17 15:14:18	2.98	10.0.4...	ecfab...	2020-12-17 20:14:19.697	
43.6711	-79.4129	2020-12-17 15:14:29.932	2020-12-17 15:14:29	2.98	10.0.4...	ecfab...	2020-12-17 20:14:29.932	
43.6711	-79.4129	2020-12-17 15:14:40.188	2020-12-17 15:14:39	2.98	10.0.4...	ecfab...	2020-12-17 20:14:40.188	
43.6711	-79.4129	2020-12-17 15:14:50.403	2020-12-17 15:14:49	2.98	10.0.4...	ecfab...	2020-12-17 20:14:50.403	
43.6711	-79.4129	2020-12-17 15:15:10.763	2020-12-17 15:15:09	2.98	10.0.4...	ecfab...	2020-12-17 20:15:10.763	
43.6711	-79.4129	2020-12-17 15:15:20.99	2020-12-17 15:15:20	2.98	10.0.4...	ecfab...	2020-12-17 20:15:20.99	
43.6711	-79.4129	2020-12-17 15:15:31.235	2020-12-17 15:15:30	2.98	10.0.4...	ecfab...	2020-12-17 20:15:31.235	
43.6711	-79.4129	2020-12-17 15:15:51.5	2020-12-17 15:15:50	2.98	10.0.4...	ecfab...	2020-12-17 20:15:51.5	



Storing sensor database in a time series database

We will use Node-RED to store the data with the mqtt topic sensor/YOURDEVICEID/data into the database. The flow is already there and we can add some missing fields to the table.

You may have to connect some of the flows and deploy the changes.

Storing the data

The PostGres Plugin has a nice SQL function to map and store the data

The screenshot shows a Node-RED flow. On the left, the palette lists common nodes like inject, debug, complete, catch, status, link in, link out, and comment; and function nodes like function, switch, change, range, template, delay, and trigger. The main canvas contains the following flow:

- An incoming message is processed by a 'Change Topic to your device' node.
- The topic is checked by a 'Check Topic' node.
- If successful, it triggers a 'TS adjustments' node.
- The message then passes through a 'snp weather data' function node.
- The output of this function is sent to a 'snpweather' node.
- A 'Test Data' node is connected to the 'snpweather' node.
- The message is then sent to an 'http' node with the URL '[post] /myesp8266'.
- The 'http' node outputs 'msg.payload'.
- This payload is then processed by another 'snp weather data' function node.
- The final output is 'msg.payload'.

On the right, a detailed view of the 'snp weather data' function node shows its properties and the function code:

```
Properties:
Name: snp weather data
Setup: 
Function:
1 var d = new Date();
2 var tsnow = (new Date(Date.now() - 1800000)).toISOString().slice(0, -1);
3 var tsutc = (new Date(Date.now()).toISOString().slice(0, -1));
4 //Greenwich Mean Time (GMT) is often interchanged or confused with
5
6
7 msg.payload.dtsinmilsec=msg.payload.dts*1000;
8 msg.payload.dts2 = (new Date(msg.payload.dtsinmilsec - (d.getTime() - tsnow)));
9
10 msg.payload.name="snp";
11 msg.payload.location="ca";
12 msg.payload = [
13   {
14     query: 'begin',
15   },
16   {
17     query:
18       'insert into snpweather (timestamp,msg,tsutc,deviceid,
19       values
20       ($timestamp,$msg,$tsutc,$dev:
21       dts:msg.payload.dts2,
22       mac:msg.payload.mac,
23       ip:msg.payload.ip,
24       voltage:msg.payload.voltage,
25       name:msg.payload.name,
26       deviceid:msg.payload.deviceid,
27       humidity:msg.payload.humidity,
28       temperature:msg.payload.temperature,
29       latitude:msg.payload.latitude,
```



Exercise



Add some of the missing fields like msg counter and mac address to same them into the database.

Add the mapping
ip:msg.payload.ip,
mac:msg.payload.mac,



7th CHAPTER (7) - Visualizing Time series Data

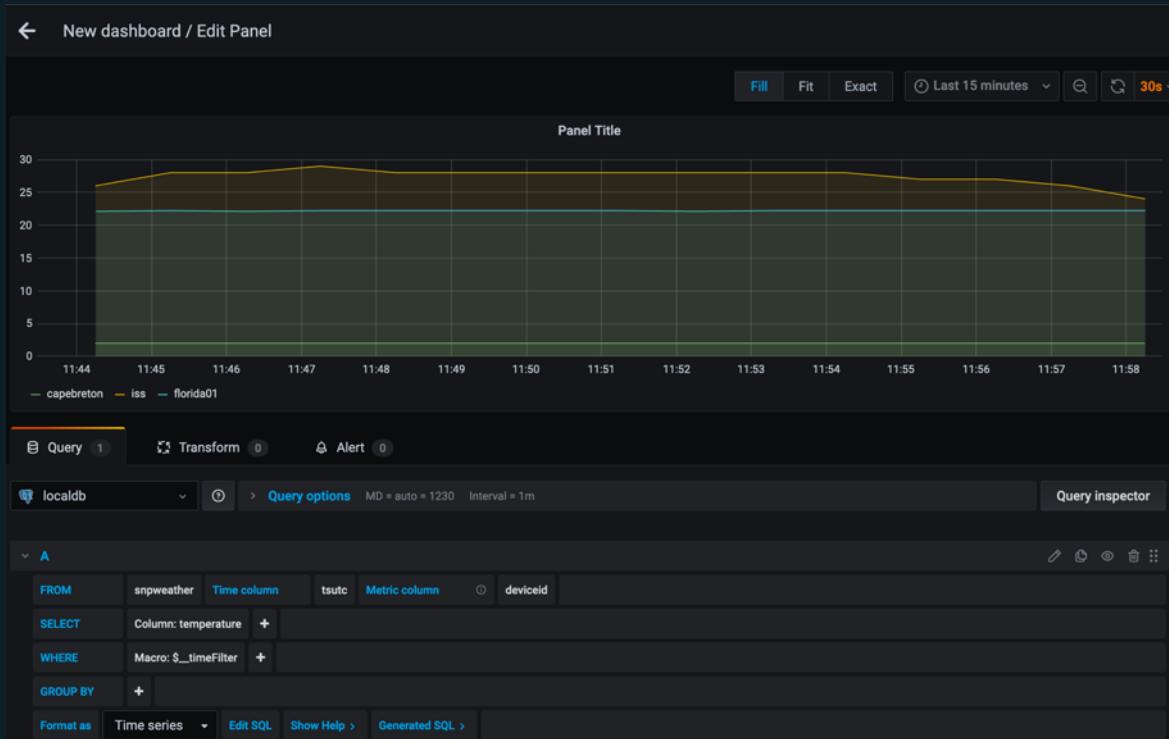
This this chapter we will visualize the timeseries Data using Grafana

Grafana is installed on our virtual server on port 3000
<http://52.117.240.201:3000/>

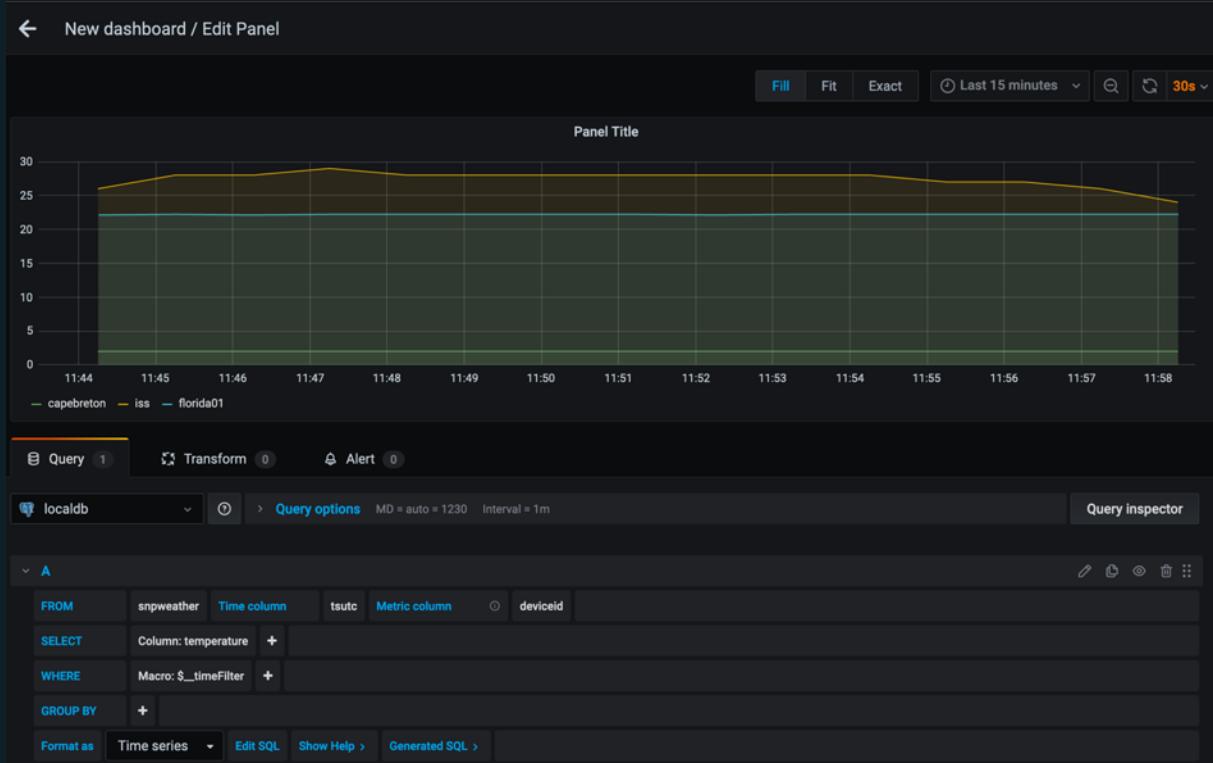
We have user1-6 setup to create dashboard.



Exercise – Create a dashboard which show the Temperature walkthrough



Exercise – create a Dashboard which show all device on a Map



Review

- We create a WebServer using an esp8266 with DHT11
- We use an NTP client to get the right time
- We send a json message to an mqtt broker
- We added some esp8266 deepsleep and reset functionality
- We displayed the DHT11 in a Node-RED realtime Dashboard
- We stored the Data into a SQL Database as a time series
- We visualized the time series data using grafana

Questions ?



