Creating an end-to-end IoT solution

In part 3 of this workshop we will create an IoT node which will be connected to our TTN node as build in part 1, using NodeRed as installed in part 2.

Prepare the ESP Wemos modules

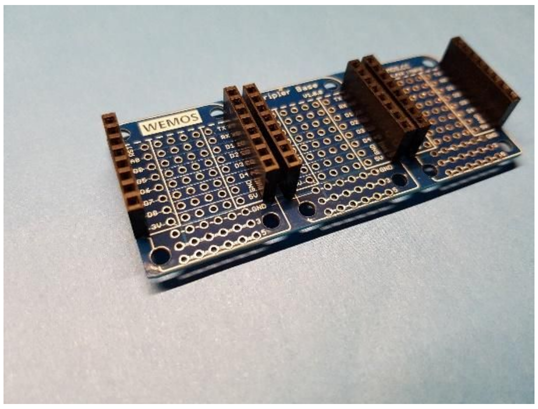
Parts needed:

- **Wemos Dual Base**
- **Wemos D1 mini v3 – ESP8266**
- **Wemos OLED shield**
- **Servo SG90 mini servo**
- **3 male-male jumper wires**

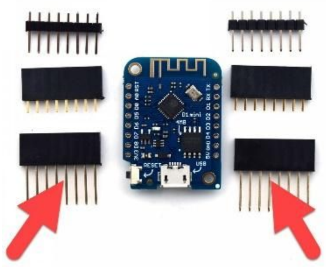
Soldering

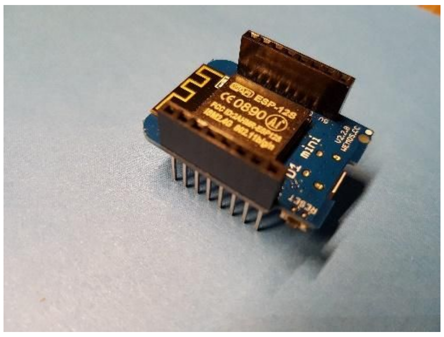
If your kit is not pre-soldered we have to solder the headers first. Be ware of the pin numbers and orientation. Each Wemos board has a small notch on the left.

The Dual Base has four female headers to solder, put them on the front side:

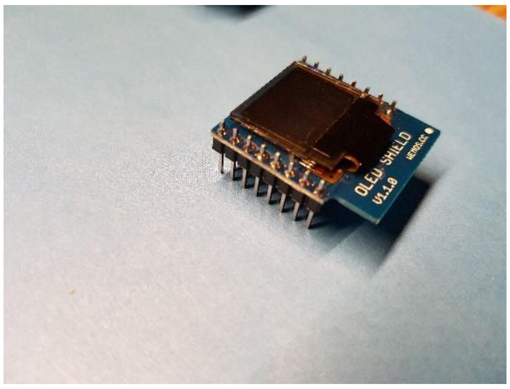


The Wemos D1 mini v3 – ESP8266 comes with three possible options. For experimental purposes use the longer female connectors.

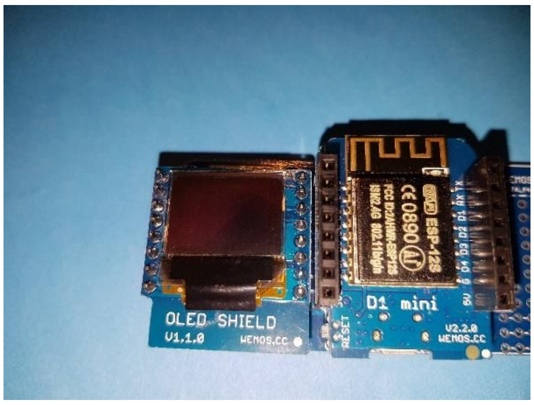




The Oled display must be always 'on top', so solder the connectors on the backside, display in front.



Now we can create our test configuration: Wemos and OLED: You can stack them (align the notches) or put them next to each other on the Dual Base:



Arduino Software

To use the ESP8266 we need the ESP toolchain. Installation instructions can be found here:
http://esp8266.github.io/Arduino/versions/2.0.0/doc/installing.html

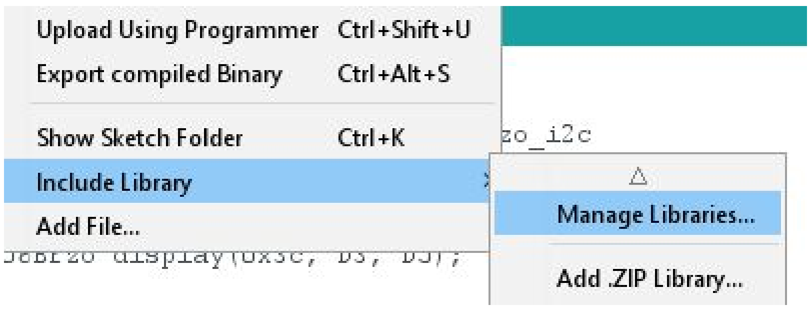
Installing with Boards Manager

Starting with 1.6.4, Arduino allows installation of third-party platform packages using Boards Manager. We have packages available for Windows, Mac OS, and Linux (32 and 64 bit).

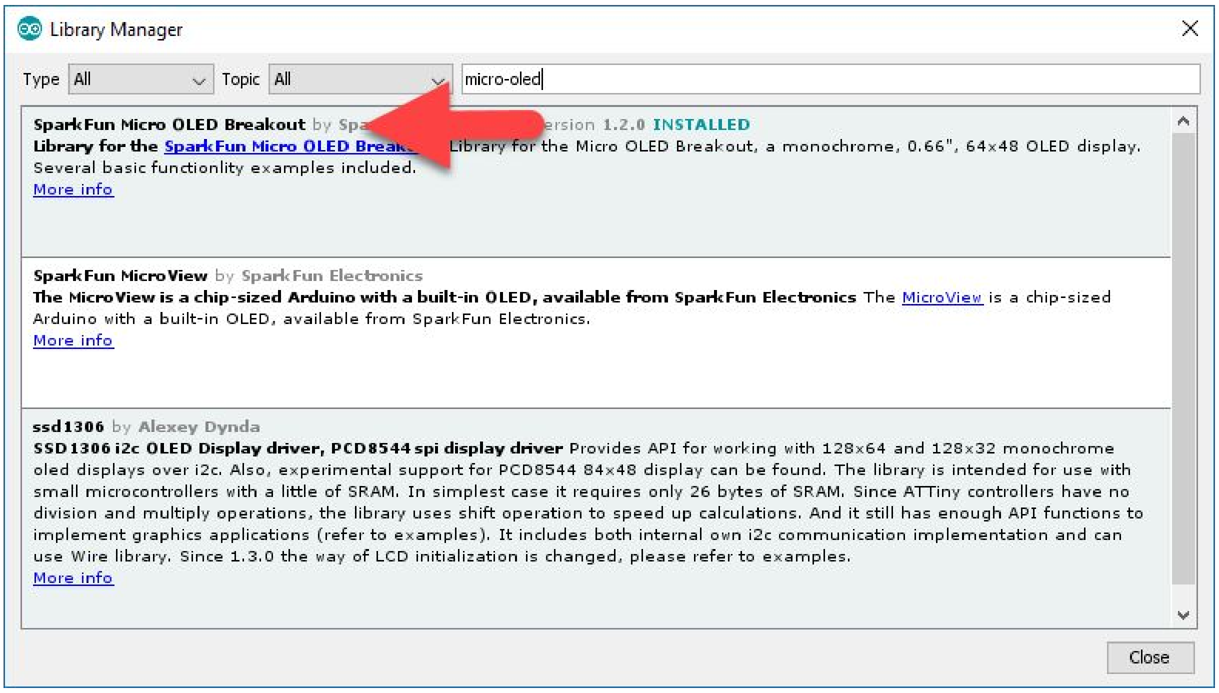
- Install the current upstream Arduino IDE at the 1.8 level or later. The current version is at the Arduino website.
- Start Arduino and open Preferences window.
- Enter `http://arduino.esp8266.com/stable/package_esp8266com_index.json` into *Add itional Board Manager URLs* field. You can add multiple URLs, separating them with commas.
- Open Boards Manager from Tools > Board menu and install *esp8266* platform (and don't forget to select your ESP8266 board from Tools > Board menu after installation).

After installation we have to add two libraries we want to use, OLED and MQTT:

Go to Sketch, Include Library, Manage Libraries:


Upload Using Programmer Ctrl+Shift+U
Export compiled Binary Ctrl+Alt+S
Show Sketch Folder Ctrl+K
Include Library
Add File...
Manage Libraries...
Add .ZIP Library...

In the library manager search for 'micro oled' and find the 'SparkFun Micro OLED Breakout' and install this library:



Library Manager

Type All Topic All micro-oled

SparkFun Micro OLED Breakout by Spa... ersion 1.2.0 INSTALLED
Library for the SparkFun Micro OLED Break... Library for the Micro OLED Breakout, a monochrome, 0.66", 64x48 OLED display. Several basic functionlity examples included.
More info

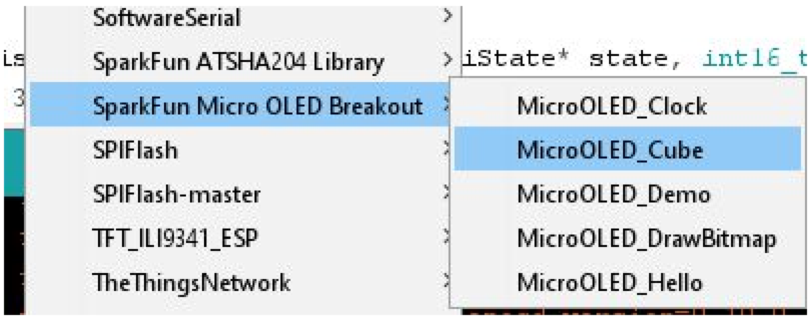
SparkFun MicroView by SparkFun Electronics
The MicroView is a chip-sized Arduino with a built-in OLED, available from SparkFun Electronics The MicroView is a chip-sized Arduino with a built-in OLED, available from SparkFun Electronics.
More info

ssd1306 by Alexey Dynda
SSD1306 i2c OLED Display driver, PCD8544 spi display driver Provides API for working with 128x64 and 128x32 monochrome oled displays over i2c. Also, experimental support for PCD8544 84x48 display can be found. The library is intended for use with small microcontrollers with a little of SRAM. In simplest case it requires only 26 bytes of SRAM. Since ATTiny controllers have no division and multiply operations, the library uses shift operation to speed up calculations. And it still has enough API functions to implement graphics applications (refer to examples). It includes both internal own i2c communication implementation and can use Wire library. Since 1.3.0 the way of LCD initialization is changed, please refer to examples.
More info

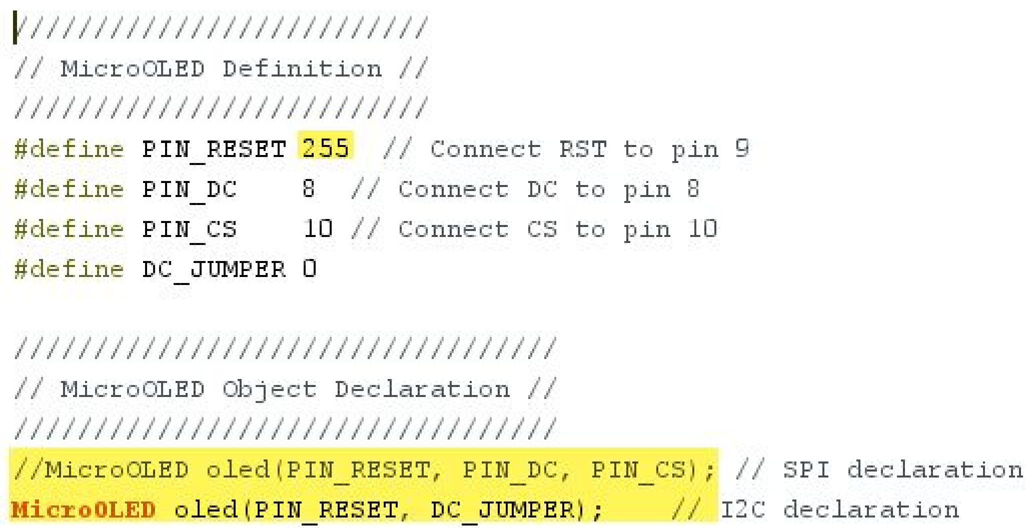
Close

Do the same for 'PubSubClient'.

We can test our setup by using the example 'MicroOLED_Cube. This can be found under 'Examples, SparkFun Micro OLED Breakout'.



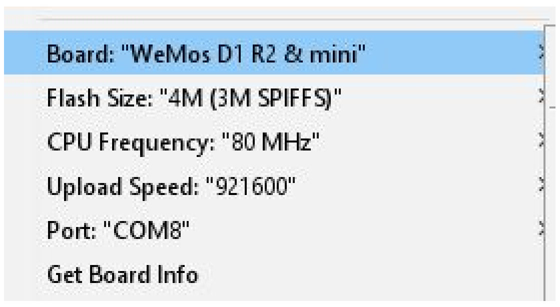
Change the code from SPI to I2C (PIN_RESET 255, comment out the SPI declaration and add the I2C declaration.



```
////////////////////////////
// MicroOLED Definition //
////////////////////////////
#define PIN_RESET 255  // Connect RST to pin 9
#define PIN_DC    8  // Connect DC to pin 8
#define PIN_CS    10 // Connect CS to pin 10
#define DC_JUMPER 0

////////////////////////////////////
// MicroOLED Object Declaration //
////////////////////////////////////
//MicroOLED oled(PIN_RESET, PIN_DC, PIN_CS); // SPI declaration
MicroOLED oled(PIN_RESET, DC_JUMPER);     // I2C declaration
```

Connect your Wemos with a micro USB data cable, choose the right COM port and board settings:



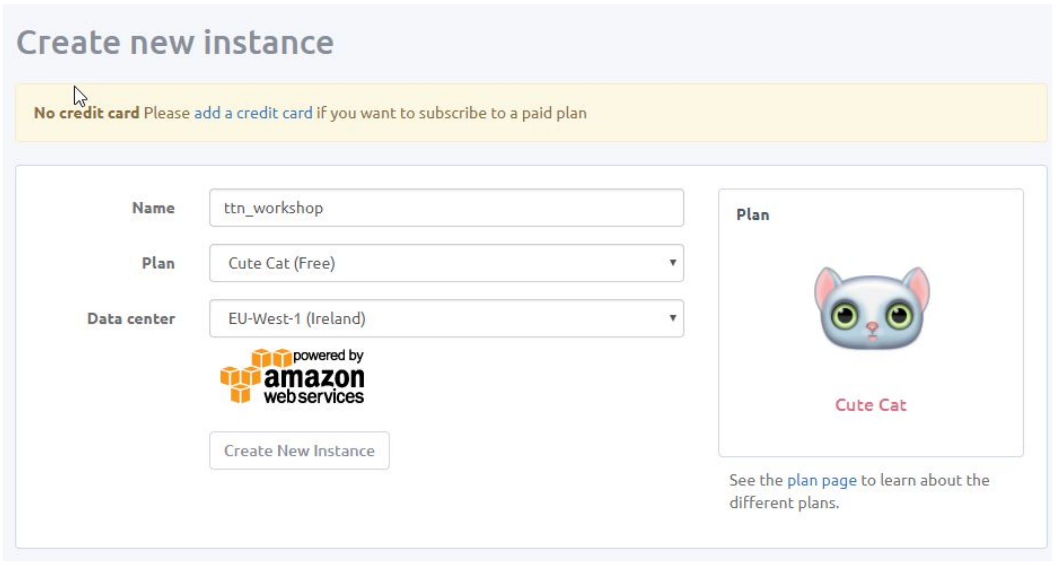
Board: "WeMos D1 R2 & mini"
Flash Size: "4M (3M SPIFFS)"
CPU Frequency: "80 MHz"
Upload Speed: "921600"
Port: "COM8"
Get Board Info

(COM port might differ!)

Upload your code, and see the cube rotating!

MQTT

Register an account at www.cloudmqtt.com. An account is free for limited data use and devices.



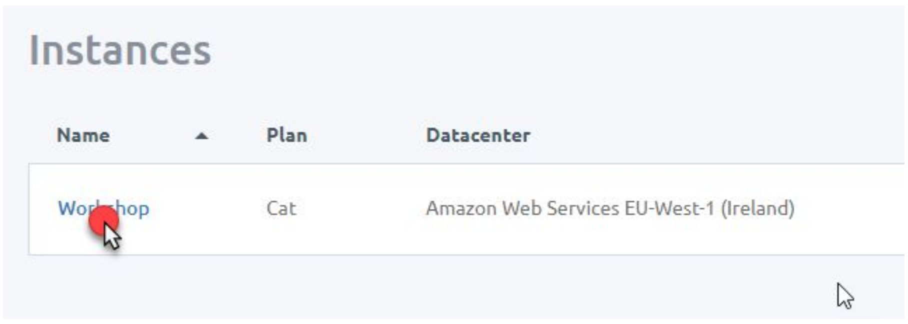
Create new instance

No credit card Please add a credit card if you want to subscribe to a paid plan

Name	ttn_workshop
Plan	Cute Cat (Free)
Data center	EU-West-1 (Ireland)

See the plan page to learn about the different plans.

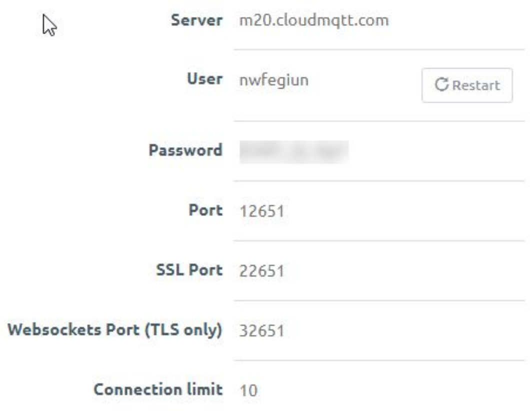
In your list of instances, select your instance:



Instances

Name	Plan	Datacenter
Workshop	Cat	Amazon Web Services EU-West-1 (Ireland)

Now the settings for your MQTT instance are shown, we will use these later on:



Server	m20.cloudmqtt.com
User	nwfegiun
Password	[redacted]
Port	12651
SSL Port	22651
Websockets Port (TLS only)	32651
Connection limit	10

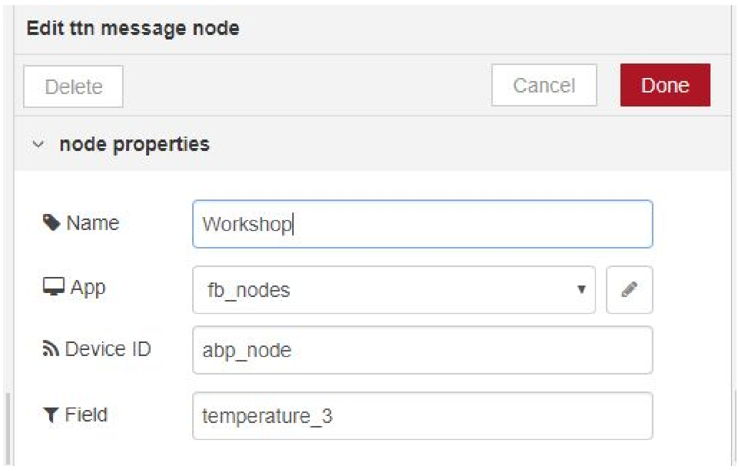
Prepare NodeRed

To sent your data to MQTT we can use 'NodeRed'.

We start with a clean 'flow'.

Add a TTN message node by drag and drop it on your flow screen.

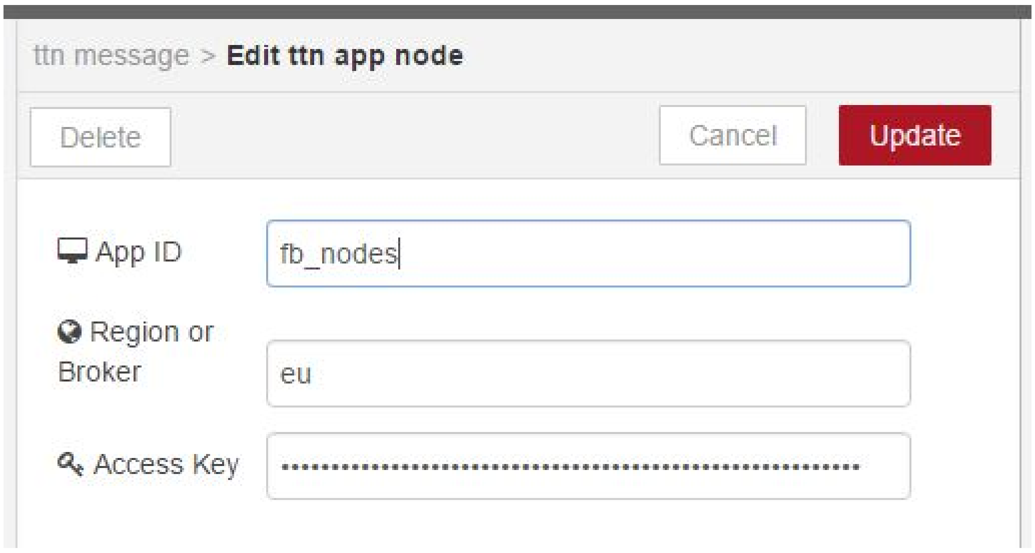
By double clicking on the TTN message node you can change the values:



Name: Workshop
App: fb_nodes
Device ID: abp_node
Field: temperature_3

The name can be any name,

App refers to your TTN console. With the Pencil you can add your application here:

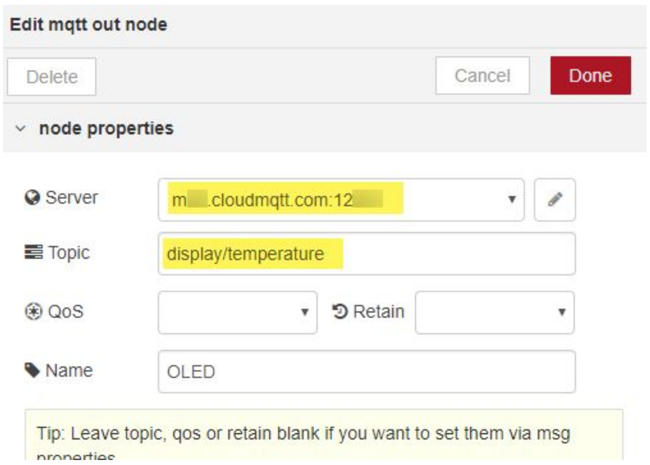


App ID: fb_nodes
Region or Broker: eu
Access Key: ••

- **App ID is the written name 'Application ID' from the TTN console**
- **Region is eu (you have to fill this in!!)**
- **Access Key is a copy of the access key of your application (access key of application, default key).**
- **Check Update and select your App in the previous screen.**
- **Device ID is the Device ID in TTN console**

You can use temperature_3 if you use the ttn_bmp_280_abp_cayenne.ino sketch. Beware that you use the same payload function in TTN as used in the previous nodes workshop!! If you do not have a payload function loaded, you can use an empty 'field', all the output will be shown.

Now add a ‘MQTT’ output node to your flow and connect the both with a wire:


Edit mqtt out node

Delete | Cancel | Done

node properties

Server: m▒▒.cloudmqtt.com:12▒▒

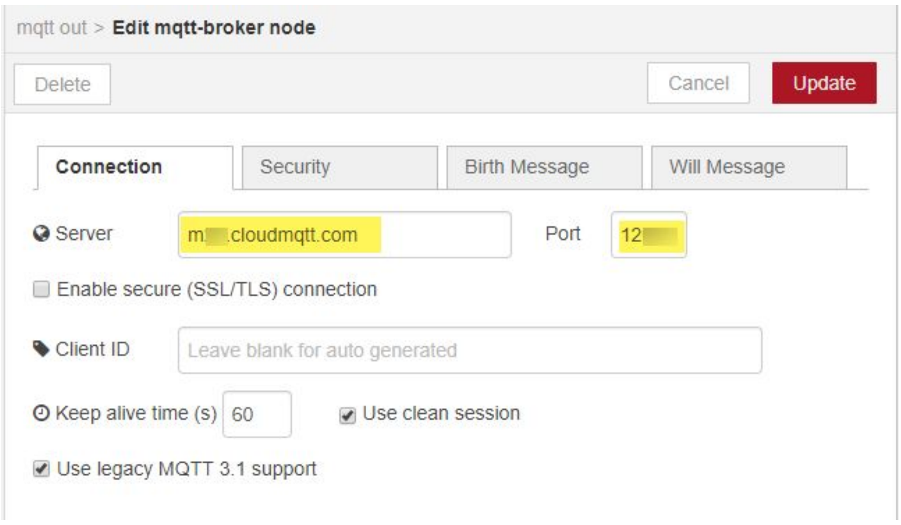
Topic: display/temperature

QoS: | Retain:

Name: OLED

Tip: Leave topic, qos or retain blank if you want to set them via msg properties

Fill in the server name and use the pencil to add your user/password.


mqtt out > Edit mqtt-broker node

Delete | Cancel | Update

Connection | Security | Birth Message | Will Message

Server: m▒▒.cloudmqtt.com Port: 12▒▒

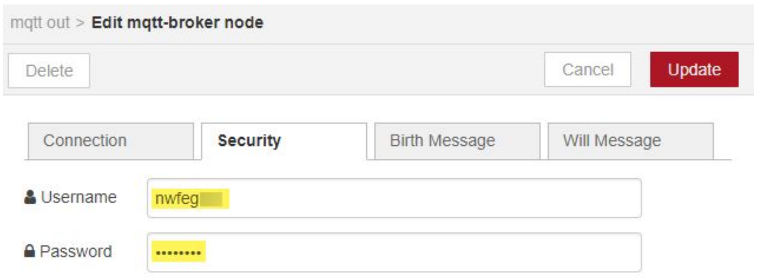
☐ Enable secure (SSL/TLS) connection

Client ID: Leave blank for auto generated

Keep alive time (s): 60 ☑ Use clean session

☑ Use legacy MQTT 3.1 support

On the tab security:


mqtt out > Edit mqtt-broker node

Delete | Cancel | Update

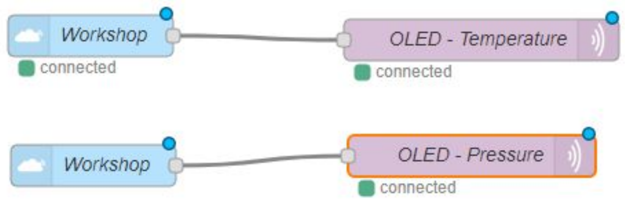
Connection | Security | Birth Message | Will Message

Username: nwfeg▒▒

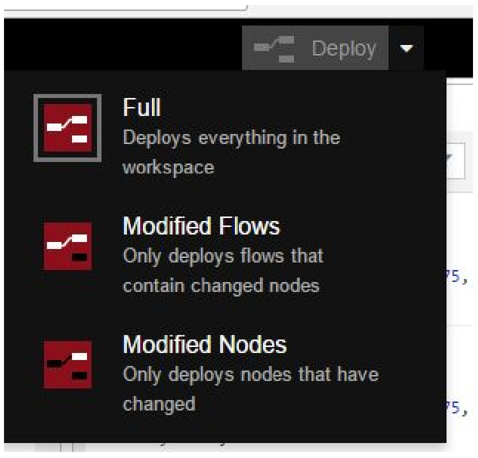
Password: ••••••••

Repeat above steps for the ‘Airpressure_4’ node, and use ‘display/pressure’ as topic.

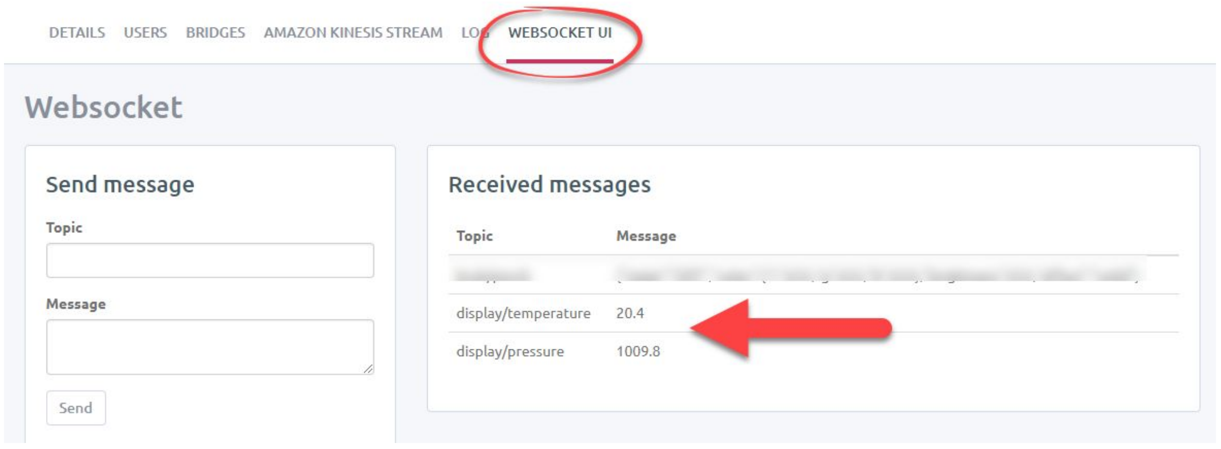
Your flow will look like this:



Now activate your flow with Deploy->Full:


Deploy
Full
Deploys everything in the workspace
Modified Flows
Only deploys flows that contain changed nodes
Modified Nodes
Only deploys nodes that have changed

We can check the working of MQTT in the CloudMQTT console. Select the Websocket UI, and you will see the values published.



Websocket

Send message

Topic

Message

Send

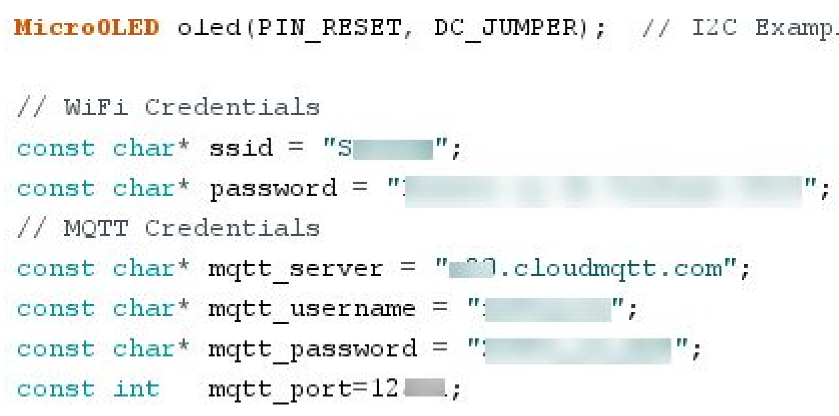
Received messages

Topic	Message
[redacted]	[redacted]
display/temperature	20.4
display/pressure	1009.8

ESP MQTT node

Download the ESP8266_MQTT_OLED sketch from
https://github.com/galagaking/esp8266_mqtt_oled

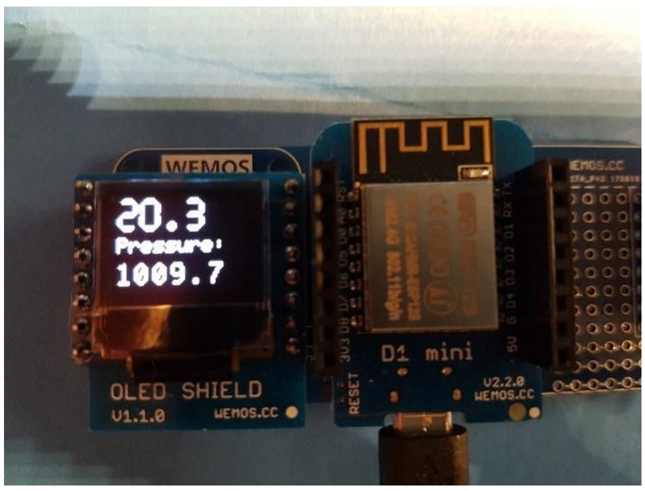
Copy the WiFi credentials to be used and your MQTT settings in the beginning of the code:



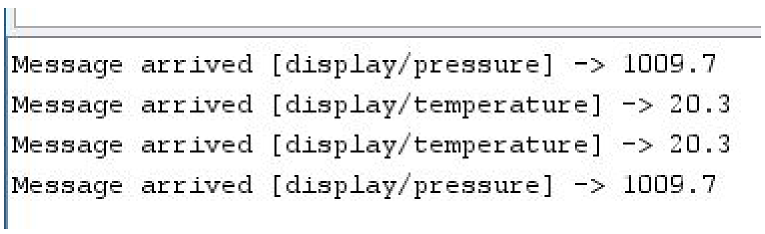
```
MicroOLED oled(PIN_RESET, DC_JUMPER);  // I2C Example

// WiFi Credentials
const char* ssid = "S[redacted]";
const char* password = "[redacted]";
// MQTT Credentials
const char* mqtt_server = "[redacted].cloudmqtt.com";
const char* mqtt_username = "[redacted]";
const char* mqtt_password = "[redacted]";
const int   mqtt_port=12[redacted];
```

Compile and upload your code to your ESP. It will reboot, connect to WiFi, your MQTT cloud server and listen to the topic 'display/+' and therefore any values coming in under 'display'. Temperature and pressure will be shown on the display:



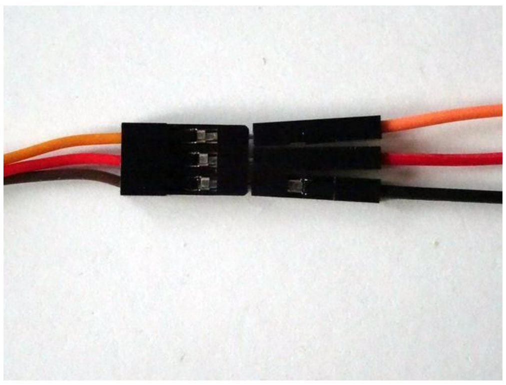
The console will display the values as well:



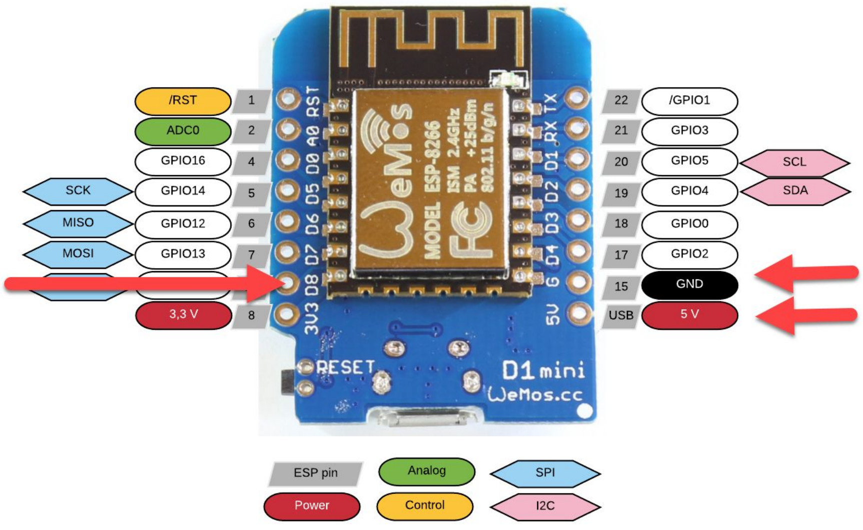
```
Message arrived [display/pressure] -> 1009.7
Message arrived [display/temperature] -> 20.3
Message arrived [display/temperature] -> 20.3
Message arrived [display/pressure] -> 1009.7
```

Connect the servo

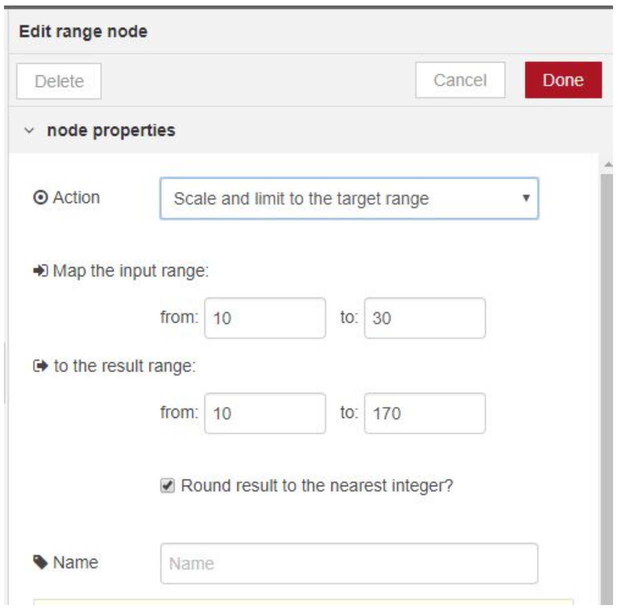
Use three jumper wires to connect your servo.



Connect the black wire (GND) to GND / G, the RED one to 5V, and the orange one to D8:



Add a 'range' node to your sketch:


Edit range node

Delete Cancel Done

node properties

Action: Scale and limit to the target range

Map the input range:
from: 10 to: 30

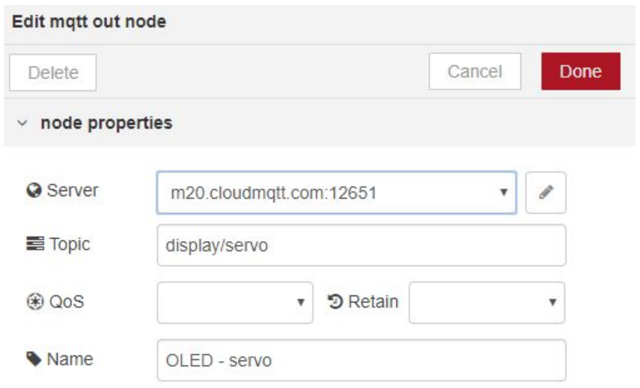
to the result range:
from: 10 to: 170

☑ Round result to the nearest integer?

Name: Name

This object will scale the temperature 10 to 30 to an angle 10 to 170 of the servo. Choose other settings if you want another range.

Copy the MQTT object for temperature and replace ‘temperature’ with ‘servo’:


Edit mqtt out node

Delete Cancel Done

node properties

Server: m20.cloudmqtt.com:12651

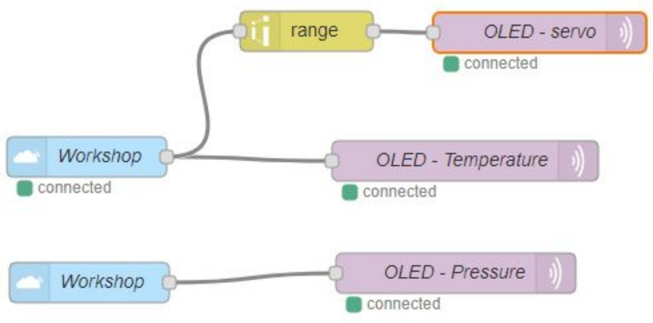
Topic: display/servo

QoS:

Retain:

Name: OLED - servo

Connect the objects:

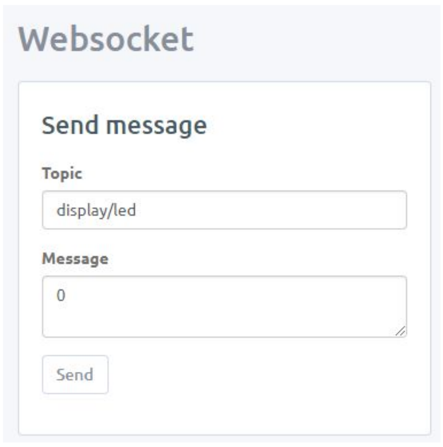


The angle of the servo will change when the temperature changes.

The ESP node can be used anywhere, just change the wifi credentials and it will connect to the (cloud) MQTT server and show your results. In this way you can move servos, but also switch lights

on and off by publishing values to MQTT. The ‘callback’ function can do the work for you, controlled by Node-Red.

You can use the webui of CloudMQTT (or any other publishing service) to publish values.


Websocket

Send message

Topic
display/led

Message
0

The onboard LED will react on display/led (0 is off, 1 is on).