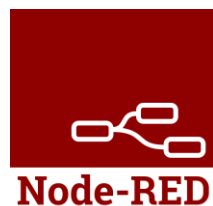


17/01/2021

Tutoriel pour la mise en place du protocole Coap & Mqtt



CoAP
Constrained Application Protocol
(Web Protocol for IoT)



4^{ème} année

Polytech Sorbonne
IOT

Charles LADZRO & AMADOU DIA
POLYTECH SORBONNE

1. Réalisation du montage

Pour l'envoi des informations dans le sens montant, nous allons utiliser un DHT11, et pour le sens descendant la led onboard de l'ESP32. La figure suivante montre le câblage à réaliser

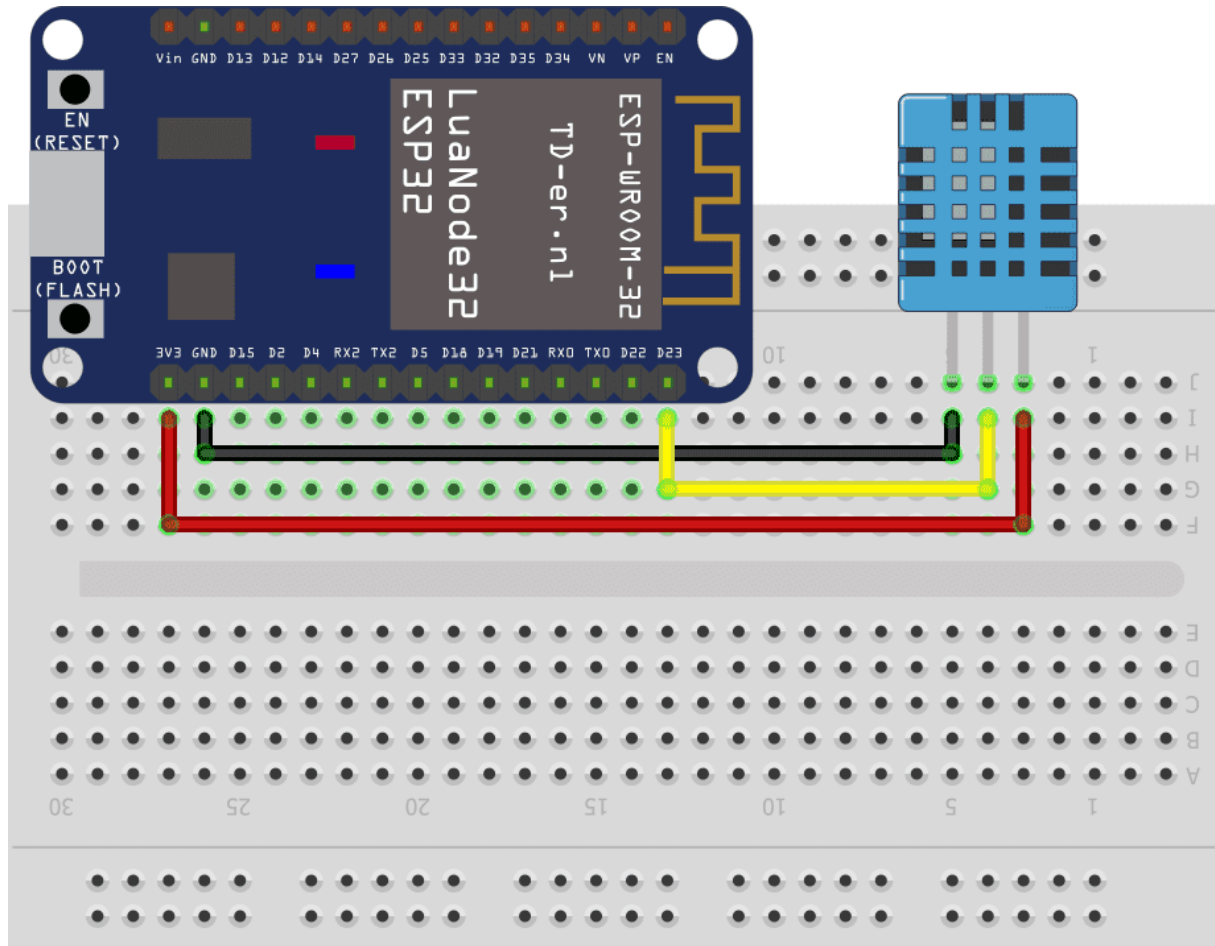
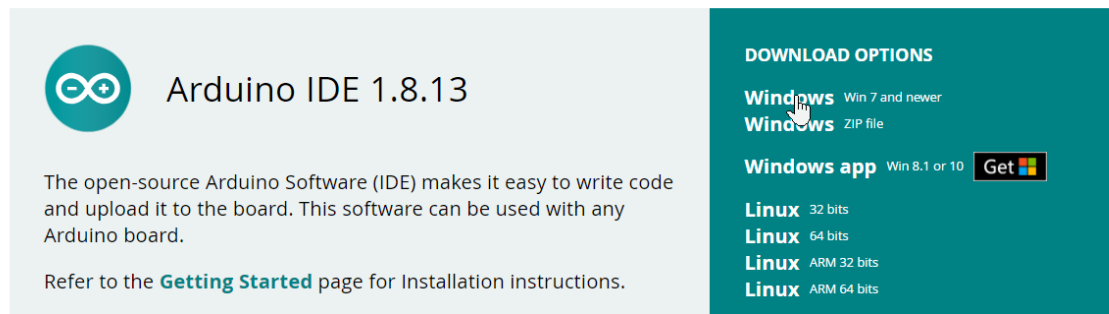


Figure 1: Schéma du câblage à réaliser avec ESP32 et DHT11 (source : <https://www.robotique.tech/tutoriel/envoyer-des-donnees-temperature-et-humidite-vers-thingspeak-en-utilisant-esp32/>)

2. Téléchargement et prise en main de Arduino

Le système d'exploitation utilisé tout au long de ce tutoriel est Windows. Le lien de téléchargement est le suivant : <https://www.arduino.cc/en/software>

Downloads



On clique sur “Windows Win7 and newer” et l’on va sur la page suivante. Sur cette page, vous pouvez faire un don ou juste cliquer sur « JUST DOWNLOAD »

Support the Arduino IDE

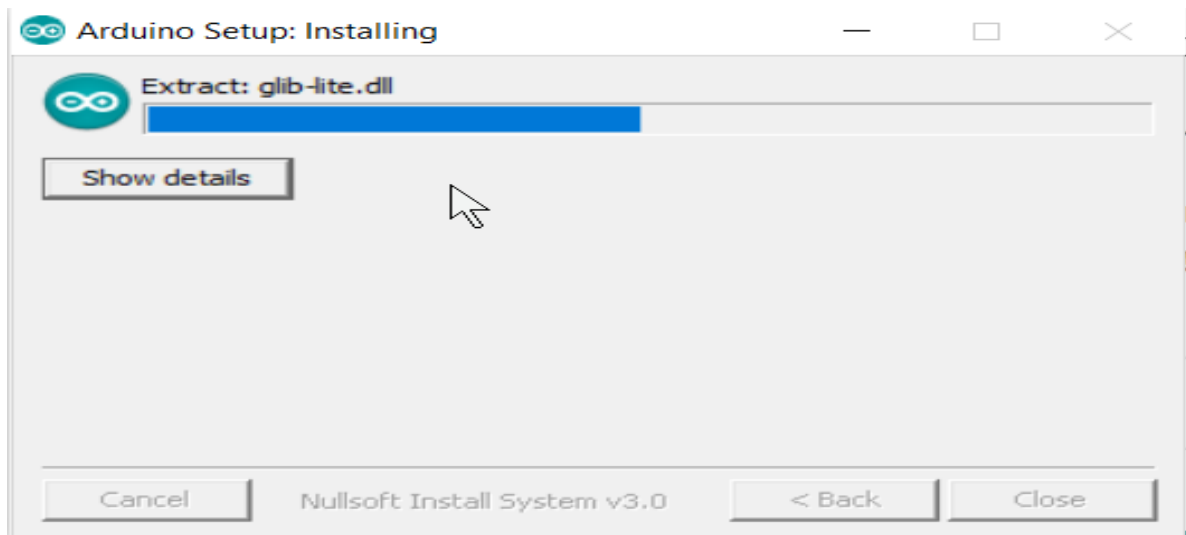
Since its first release in March 2015, the Arduino IDE has been downloaded **48 411 023** times — impressive! Help its development with a donation.

\$3	\$5	\$10	\$25	\$50	Other
-----	-----	------	------	------	-------

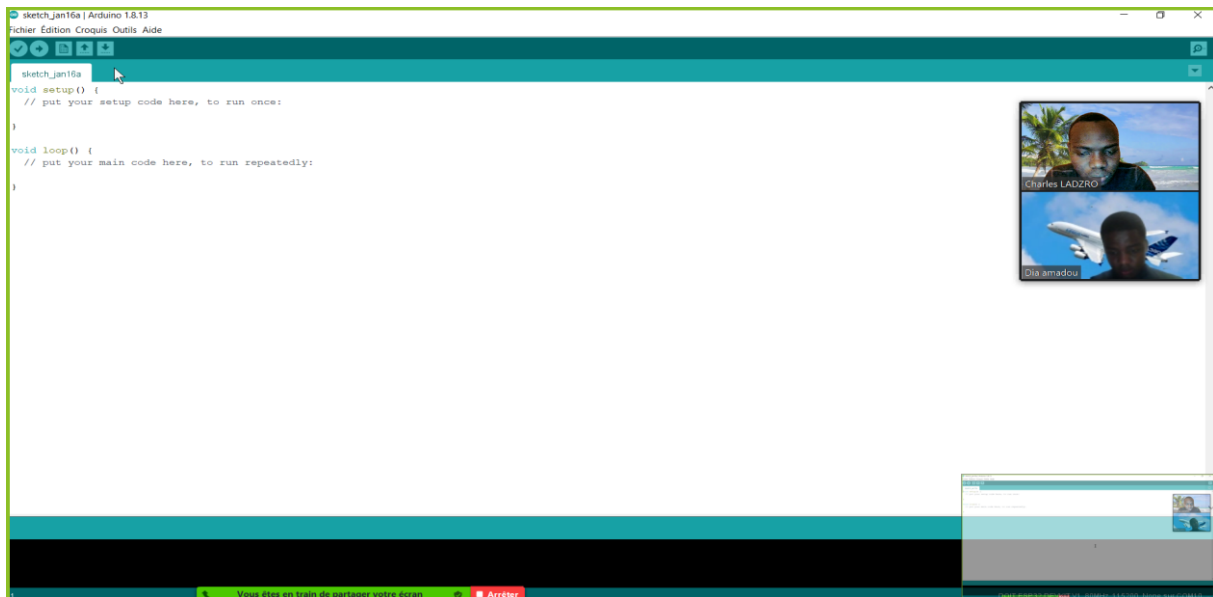
JUST DOWNLOAD

CONTRIBUTE & DOWNLOAD

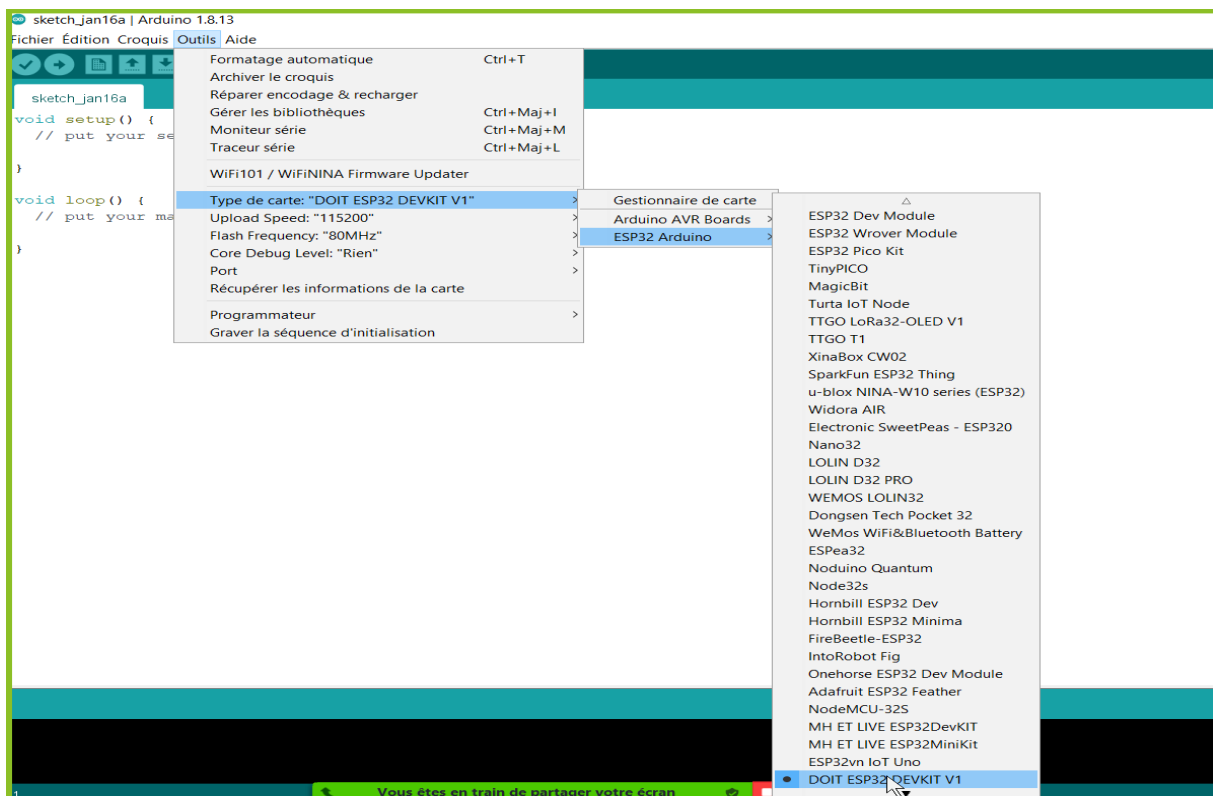
Après le téléchargement, on exécute le logiciel. On clique sur Suivant, Suivant et Terminer.



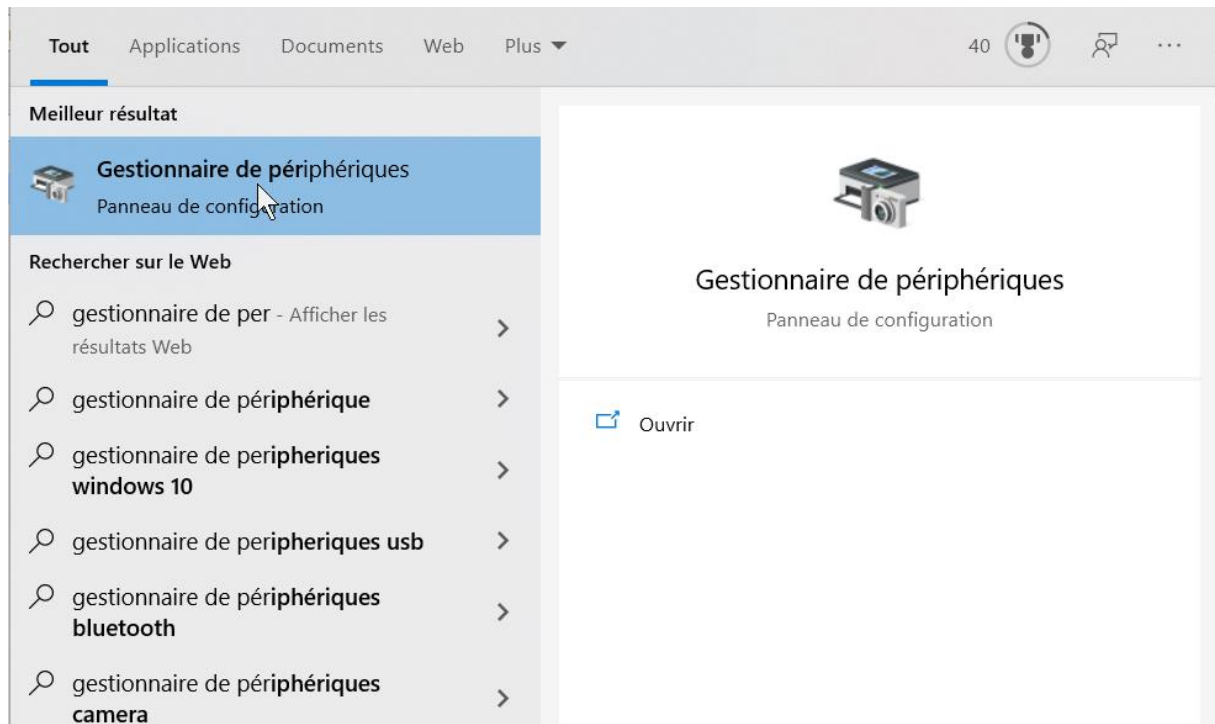
Après l’installation, on lance le programme Arduino, et l’on obtient :



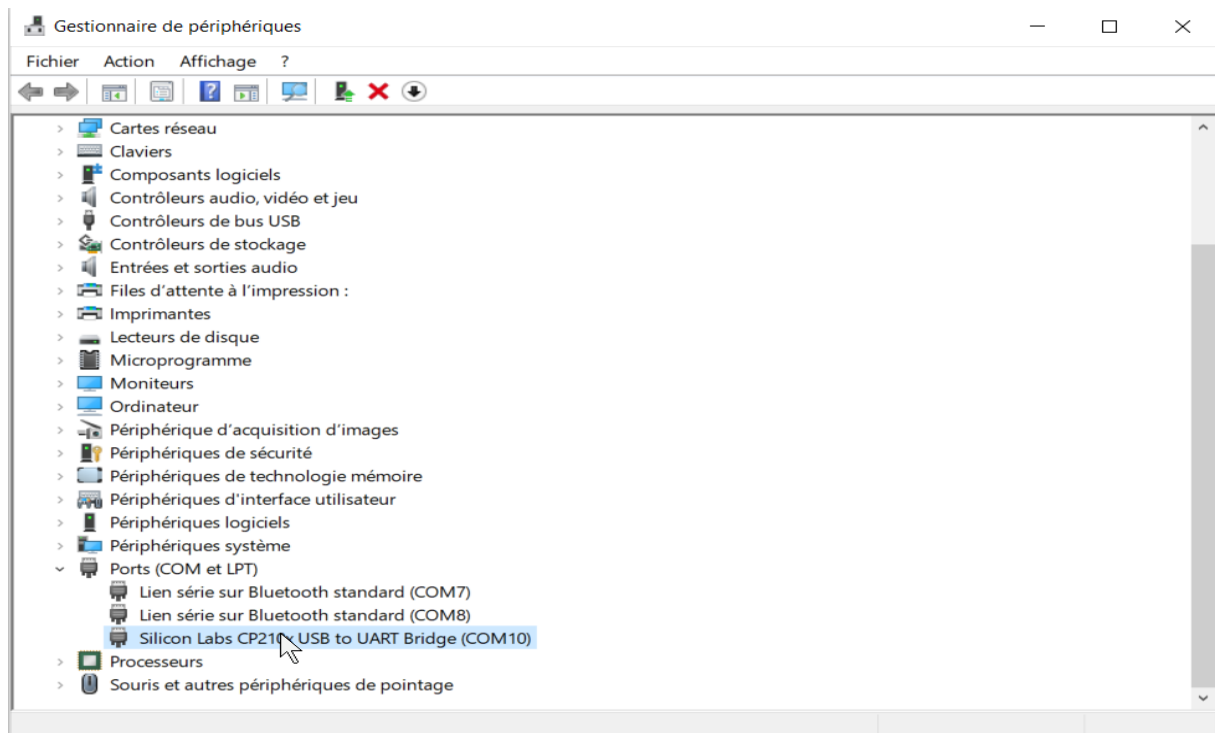
Pour pouvoir utiliser l'ESP32, il faut choisir la carte dans le logiciel Arduino comme suit :



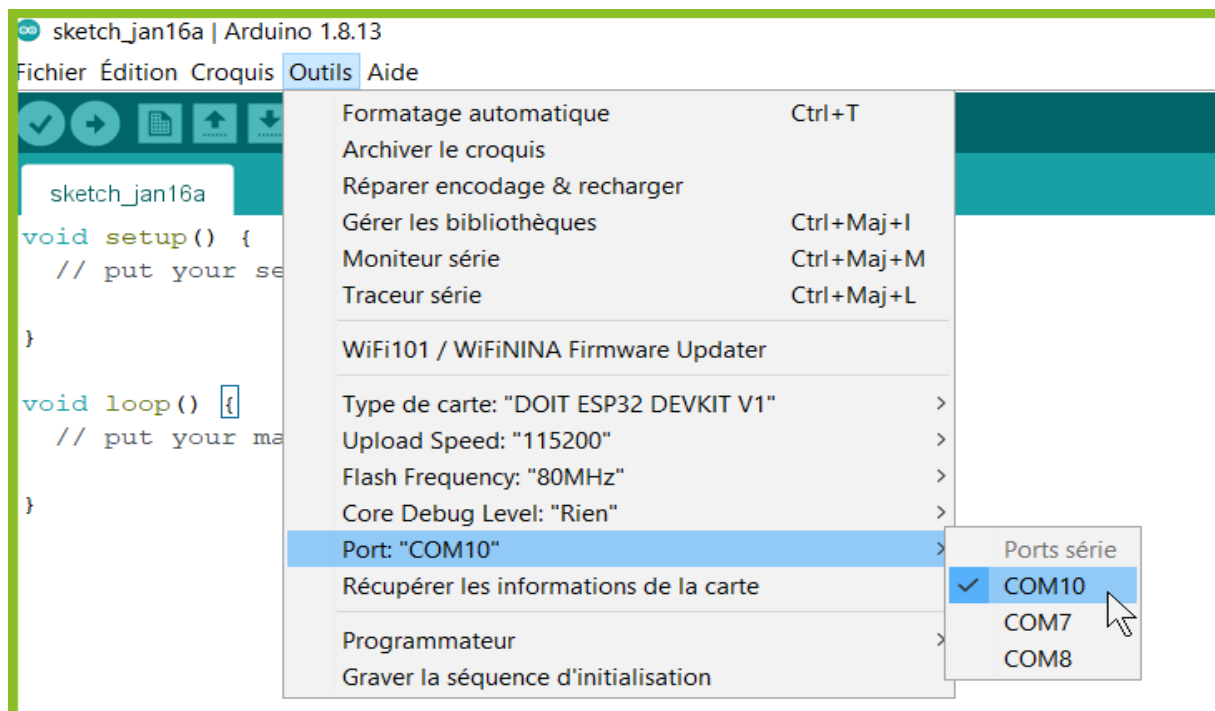
Ensuite il faut brancher la carte ESP32 à l'ordinateur et vérifier le port pour le mettre après dans le logiciel Arduino. Pour cela, il faut faire une recherche dans la barre d'outil Windows sur « Gestionnaire de Periph »



En cliquant dessus, l'on obtient la figure suivante. Ensuite, on navigue dans Ports (COM et LPT) et on cherche « Silicon Labs » et on note le port inscrit à côté. Dans notre cas, on est sur le **COM10**.

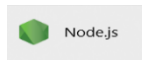


L'étape suivante est d'aller dans le logiciel Arduino pour choisir le port.

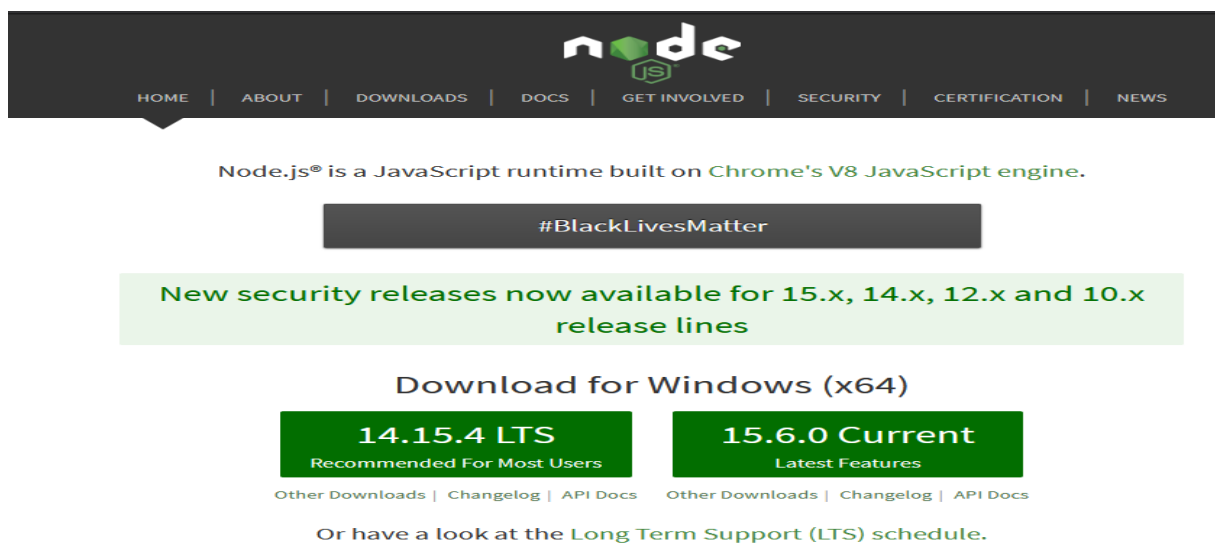


En utilisant le lien suivant : https://github.com/charlesladzro/Protocole-MQTT_COAP/raw/main/Tutorial_Getting_Started_ESP32.pdf, vous aurez accès au document « Tutorial_Getting_Started_ESP32.pdf », qui explique la mise en place de l'esp32 avec le logiciel Arduino ainsi qu'un code de test.

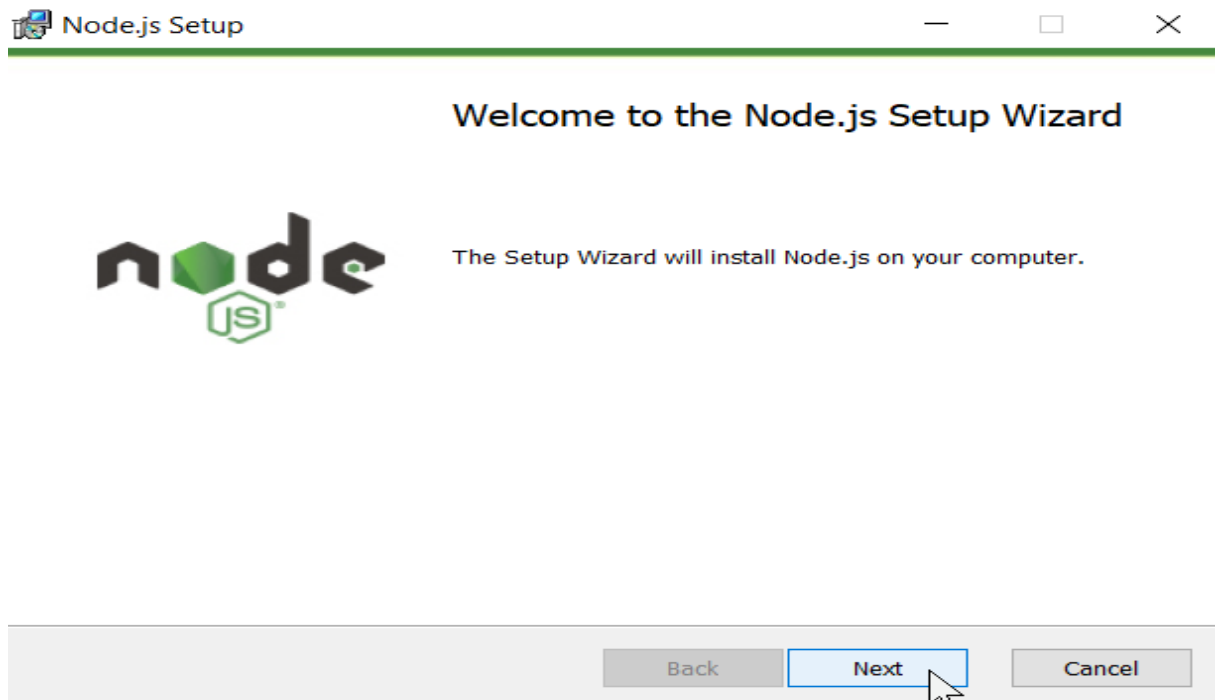
3. Téléchargement et installation du node-red local



Pour utiliser node-red en local il faut Télécharger node.js par le lien : <https://nodejs.org/en/>

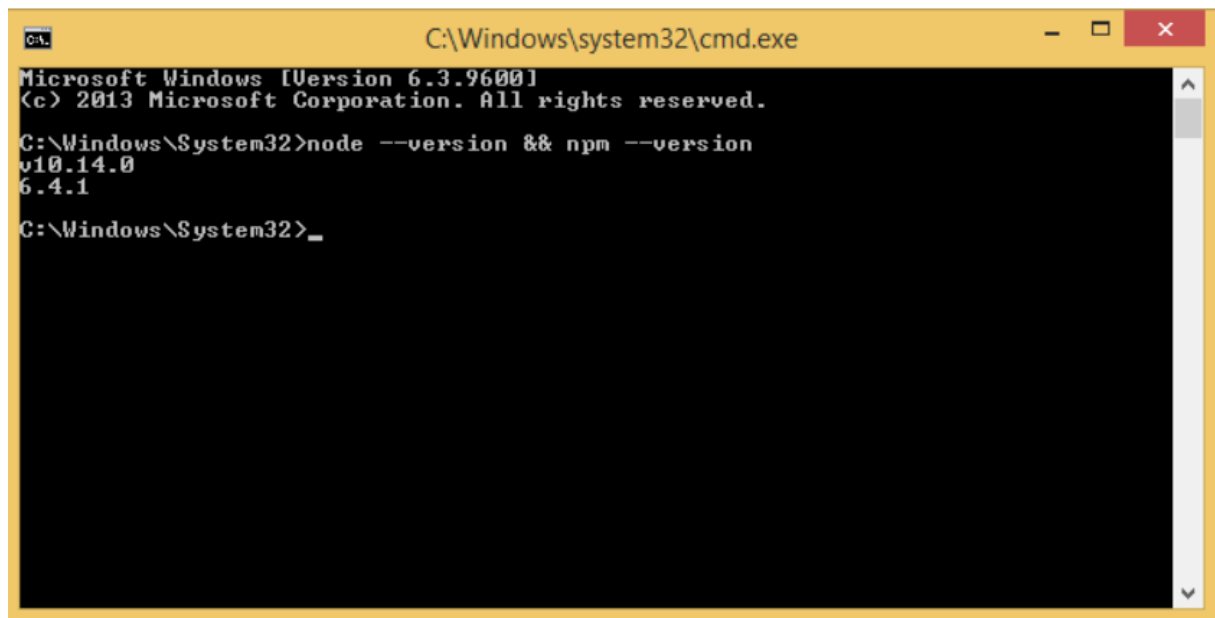


Si on accède sur cette page il faut appuyer sur le bouton **14.15.4 LTS**. Une fois le setup du node.js téléchargé, il faut l'exécuter. Lors de l'installation, cliquer sur « Next » pour poursuivre l'installation.



Une fois l'installation terminée, ouvrez une invite de commande Windows, vous pouvez l'ouvrir en tapez cmd puis une fois ouverte, entrez ce qui suit :

Tapez cette commande `node --version && npm --version`



NPM est un gestionnaire de paquets pour le langage de programmation JavaScript, il est nécessaire pour l'installation du paquet Node-Red supplémentaire dont nous aurons besoin plus tard. NPM est installé par défaut lors de l'installation de Node.js.

Nous pouvons maintenant commencer à installer Node-Red, naviguer jusqu'à CMD et entrer la commande :

Tapez cette commande **npm install -g --unsafe-perm node-red**

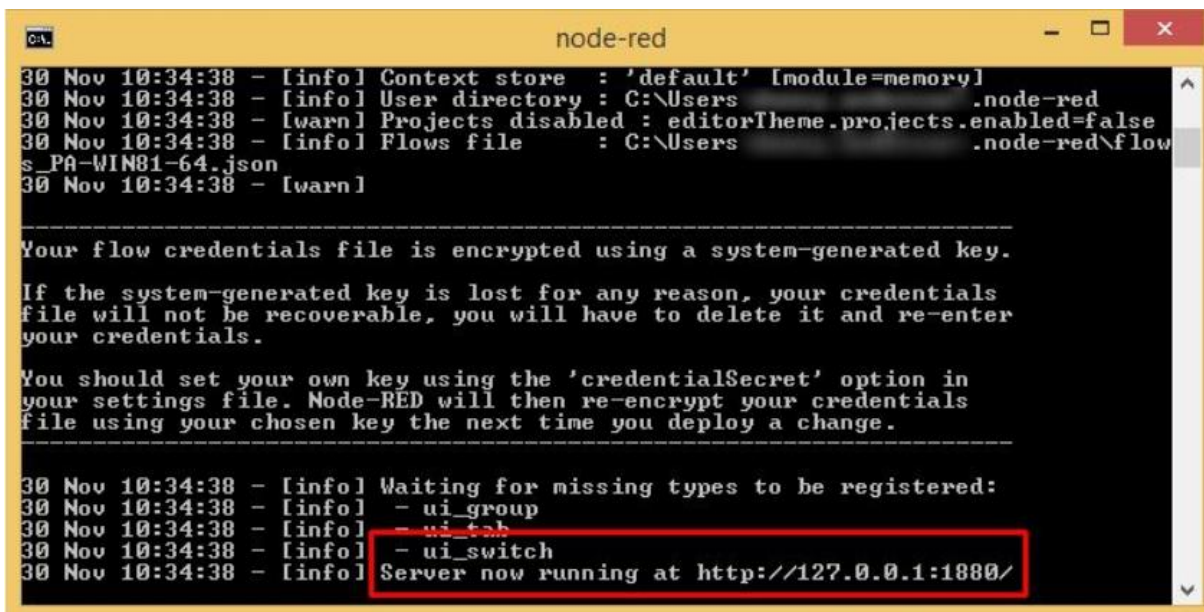


```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Windows\System32>npm install -g --unsafe-perm node-red
npm WARN registry Using stale data from https://registry.npmjs.org/ because the
host is inaccessible -- are you offline?
npm WARN registry Using stale package data from https://registry.npmjs.org/ due
to a request error during revalidation.
npm WARN deprecated mailparser@0.6.2: Mailparser versions older than v2.3.0 are
deprecated
npm WARN deprecated nodemailer@1.11.0: All versions below 4.0.1 of Nodemailer ar
e deprecated. See https://nodemailer.com/status/
npm WARN deprecated minilib@0.3.1: This project is unmaintained
npm WARN deprecated mailcomposer@2.1.0: This project is unmaintained
npm WARN deprecated buildmail@2.0.0: This project is unmaintained
C:\Users\AppData\Roaming\npm\node-red -> C:\Users
AppData\Roaming\npm\node_modules\node-red\red.js
C:\Users\AppData\Roaming\npm\node-red-pi -> C:\Users
on\AppData\Roaming\npm\node_modules\node-red\bin\node-red-pi
+ node-red@0.19.5
added 10 packages and updated 5 packages in 68.649s

C:\Windows\System32>
```

Pour démarrer Node-Red il faut juste taper sur le cmd **node-red**



```
node-red
30 Nov 10:34:38 - [info] Context store : 'default' [module=memory]
30 Nov 10:34:38 - [info] User directory : C:\Users\...\.node-red
30 Nov 10:34:38 - [warn] Projects disabled : editorTheme.projects.enabled=false
30 Nov 10:34:38 - [info] Flows file : C:\Users\...\.node-red\flow
s_PA-WIN81-64.json
30 Nov 10:34:38 - [warn]

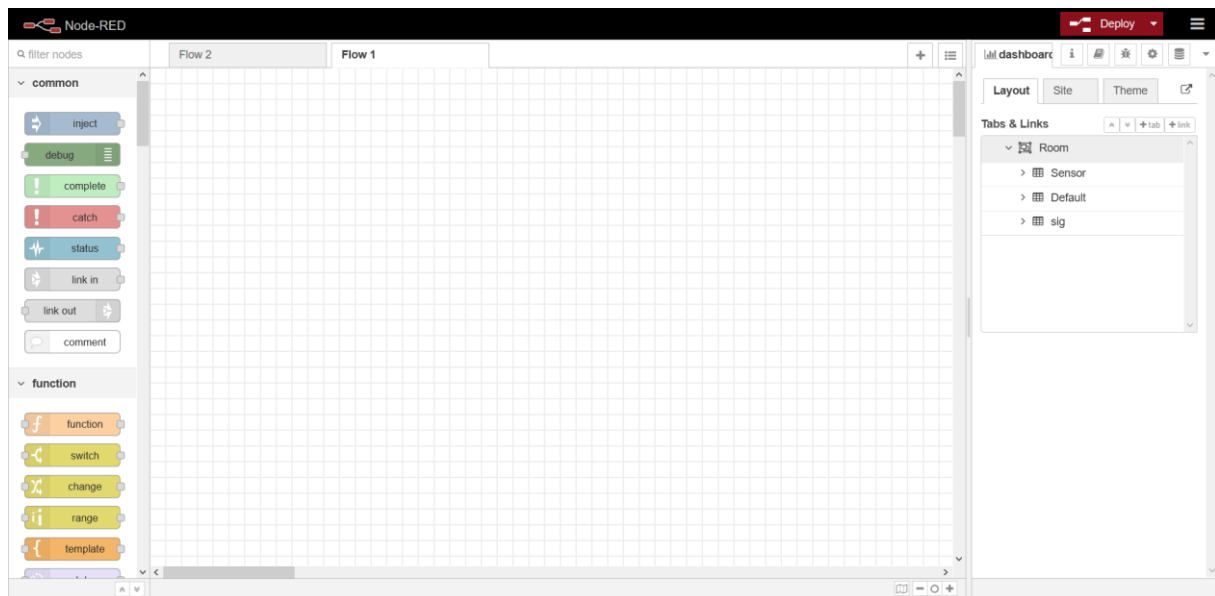
-----
Your flow credentials file is encrypted using a system-generated key.

If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.

You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
-----

30 Nov 10:34:38 - [info] Waiting for missing types to be registered:
30 Nov 10:34:38 - [info]   - ui_group
30 Nov 10:34:38 - [info]   - ui_tab
30 Nov 10:34:38 - [info]   - ui_switch
30 Nov 10:34:38 - [info] Server now running at http://127.0.0.1:1880/
```


Prenez note de l'adresse IP du serveur web ainsi que le port de communication du node-red et tapez-la dans votre navigateur web.



Dans la suite du tutoriel, nous allons mettre en place les protocoles **MQTT** et **COAP**. Pour l'envoi des informations dans le sens montant, nous allons utiliser un DHT11 et pour le sens descendant la led onboard de l'ESP32.

4. Installation des librairies nécessaires dans Arduino

• Installation de la bibliothèque PubSubClient

La bibliothèque PubSubClient (<https://github.com/knolleary/pubsubclient>) fournit un client pour faire de simples messages de publication / abonnement avec un serveur qui prend en charge MQTT (permet essentiellement à votre ESP32 de parler avec Node-RED).

- 1) Télécharger la bibliothèque PubSubClient en utilisant ce lien : <https://github.com/knolleary/pubsubclient/archive/master.zip> . Vous devriez avoir un dossier .zip dans votre dossier Téléchargements
- 2) Décompressez le dossier .zip et vous devriez obtenir le dossier pubsubclient-master
- 3) Renommez votre dossier de pubsubclient-master en pubsubclient
- 4) Déplacez le dossier pubsubclient vers votre dossier de bibliothèque d'installation de l'IDE Arduino, généralement c'est **C:\Users\... \Documents\Arduino\libraries**
- 5) Ensuite, rouvrez votre IDE Arduino

La bibliothèque est livrée avec un certain nombre d'exemples de croquis. Voir Fichier> Exemples> PubSubClient dans le logiciel Arduino IDE.

- **Installation de la bibliothèque de capteurs DHT**

La bibliothèque de capteurs DHT (<https://github.com/adafruit/DHT-sensor-library>) offre un moyen facile d'utiliser n'importe quel capteur DHT pour lire la température et l'humidité avec vos cartes ESP32 ou Arduino.

- 1) Télécharger la bibliothèque de capteurs DHT en utilisant ce lien : <https://github.com/adafruit/DHT-sensor-library/archive/master.zip> . Vous devriez avoir un dossier .zip dans vos téléchargements
- 2) Décompressez le dossier .zip et vous devriez obtenir le dossier DHT-sensor-library-master
- 3) Renommez votre dossier de DHT-sensor-library-master en DHT
- 4) Déplacez le dossier DHT vers votre dossier de bibliothèque d'installation Arduino IDE
- 5) Ensuite, rouvrez votre IDE Arduino

- **Installation de la bibliothèque Adafruit_Unified_Sensor**

- 1) Télécharger la bibliothèque Adafruit_Unified_Sensor en utilisant ce lien : https://github.com/adafruit/Adafruit_Sensor/archive/master.zip . Vous devriez avoir un dossier .zip dans vos téléchargements
- 2) Décompressez le dossier .zip et vous devriez obtenir le dossier Adafruit_Sensor-master
- 3) Renommez votre dossier de Adafruit_Sensor-master en Adafruit_Unified_Sensor
- 4) Déplacez le dossier Adafruit_Unified_Sensor vers votre dossier de bibliothèque d'installation Arduino IDE
- 5) Ensuite, rouvrez votre IDE Arduino

5. Protocole MQTT

Pour mettre en place le protocole MQTT, nous nous sommes inspiré du lien suivant : <https://randomnerdtutorials.com/esp8266-and-node-red-with-mqtt/>

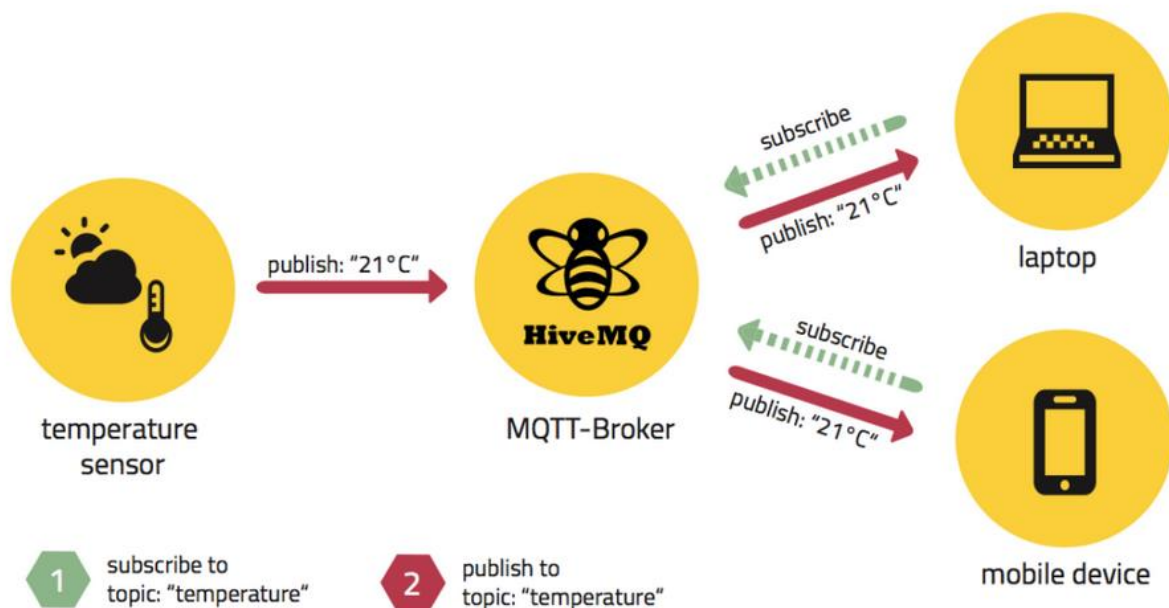
a. L'objectif de la mise en place du protocole

L'objectif est de montrer 2 démos qui permettent de vérifier la connexion bidirectionnelle :

- Dans le sens montant (uplink) remonter et afficher sur un graphique la température et l'humidité par l'intermédiaire du capteur DHT11.
- Dans le sens descendant (downlink) activer une LED depuis une interface web.

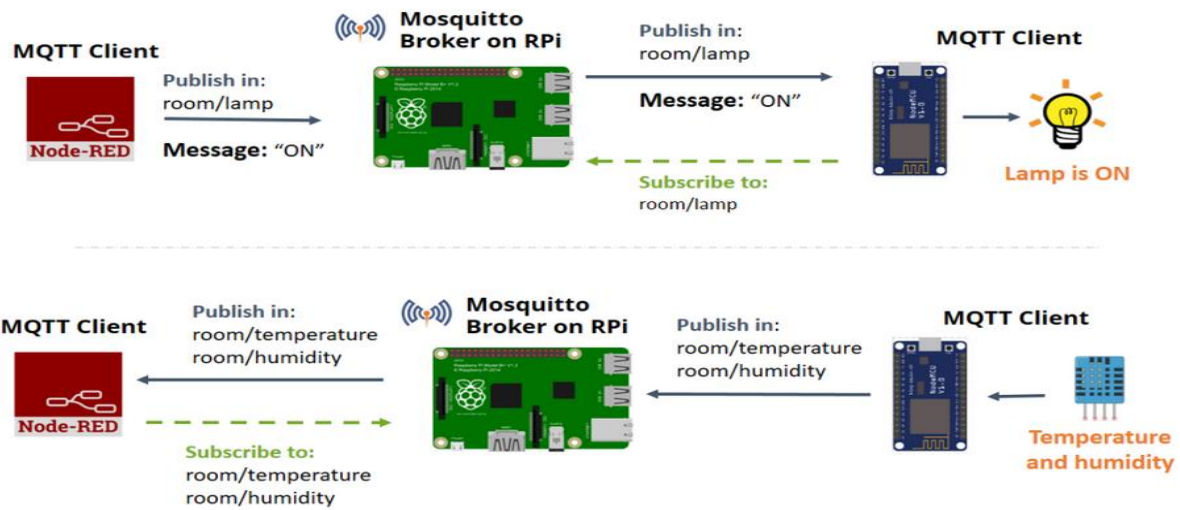
Pour mettre en évidence ce protocole, Il faut mettre en place 3 parties (voir figure suivante) :

- Une interface graphique pour une communication avec le client. Elle gère la réception des données et pourra aussi commander la communication downlink qui sera dans notre cas le contrôle de la LED.
- Un broker gère pour communiquer avec MQTT, c'est-à-dire un programme en charge de la réception des informations publiées afin de les transmettre aux clients abonnés. Le broker a un rôle de relais.
- Une partie de publication qui envoie les données des capteurs. Dans notre cas on met en évidence le capteur DHT11 par l'intermédiaire du ESP32.



b. Schéma d'explication des liaisons effectuées pour la mise en place du protocole

Sur le schéma ci-dessous, on voit bien l'affichage de la température de l'humidité qui sont récupérés par la communication en uplink mais aussi un bouton qui permet de gérer la communication downlink pour allumer la carte ESP32.



c. Mise en place du broker

Pour simplifier le processus, nous avons opté pour un broker en ligne dont les informations sont les suivantes (<http://www.mqtt-dashboard.com/>) :

MQTT connection settings

Host: `broker.hivemq.com`

TCP Port: `1883`

Websocket Port: `8000`

d. Programmation de la carte ESP32

C'est la partie programmation qui est faite sur ESP32. Cette partie gère la connexion du module ESP32 avec le broker ainsi de faire l'envoi des données de la température et de l'humidité. Un programme est disponible sur le lien du GitHub suivant : https://github.com/charlesladzro/Protocole-MQTT_COAP/raw/main/mqtt_nodered.zip. Extraire le fichier « **mqtt_nodered.ino** » et ouvrez le dans le logiciel Arduino.

La figure ci-dessous montre deux variables qui servent à connecter l'ESP32 sur le réseau wifi qu'on utilise. YOUR_SSID est le nom du WIFI et YOUR_PASSWORD le mot de passe du wifi.

```
// Change the credentials below, so your ESP8266 connects to your router
const char* ssid = "YOUR_SSID";
const char* password = "YOUR_PASSWORD";
```

La figure ci-dessous montre deux variables qui servent à mettre l'adresse IP du broker ainsi que le port du broker.

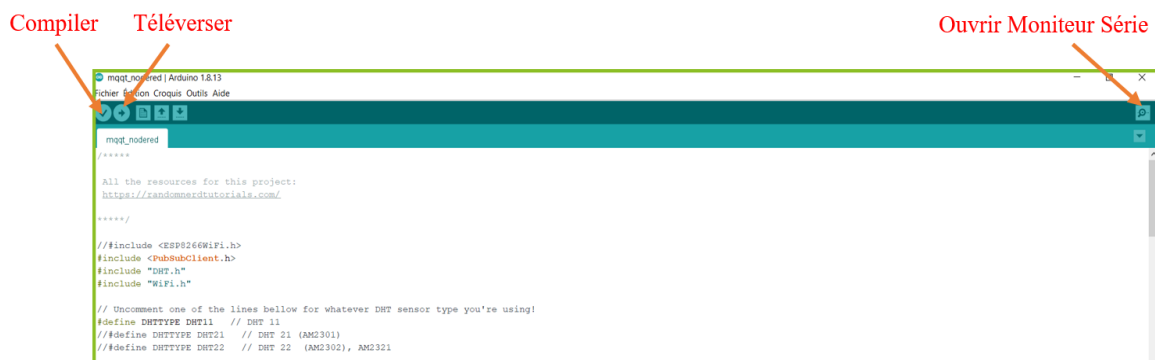
```
// Change the variable to your Raspberry Pi IP address, so it connects to your MQTT broker
const char* mqtt_server = "broker.hivemq.com";
const int mqtt_server_port = 1883;
```

La figure ci-dessous consiste à définir les pins utilisés pour le capteur DHT11 et le led onboard.

```
// DHT Sensor - GPIO 23 = D23 on ESP-12E NodeMCU board
const int DHTPin = 23;

// Lamp - LED - GPIO 2 = D2 on ESP-12E NodeMCU board
const int lamp = 2;
```

Après ces configurations il faut compiler et téléverser le code. Lors du téléversement, appuyé sur le bouton BOOT situé sur la carte ESP32 (voir figure ci-dessous).



```
COM3
```

```
[
Connecting to NumericUpDown=4586
.....
WiFi connected - ESP IP address: 192.168.0.27
Attempting MQTT connection...connected
Failed to read from DHT sensor!
Failed to read from DHT sensor!
Failed to read from DHT sensor!
Failed to read from DHT sensor!
Failed to read from DHT sensor!
Failed to read from DHT sensor!
esp8266 v 2016 01/22/157

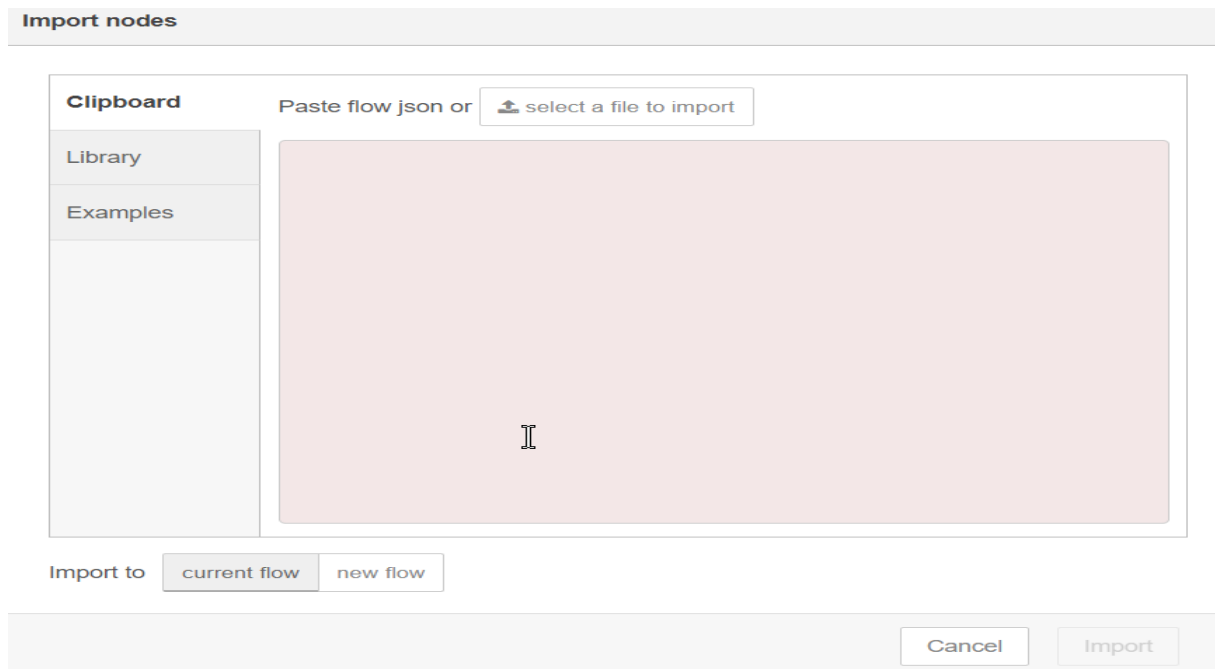
retrol (POWERON_RESET),boot(0x17 (SPI_FLASH_ROOT)
configuart 0, SPI0HSIbase
clk_devicex0,q_devicex0,q_devicex0,cad_devicex0,bd_devicex0,vp_devicex0
mode(DIO, clock devicex1
load(0x0ffff018,lenx4
load(0x0ffff01c,lenx1044
load(0x0ff00000,lenx1096
load(0x00000400,lenx516
entry 0x00000400

Connecting to NumericUpDown=4586
.....
WiFi connected - ESP IP address: 192.168.0.27
Attempting MQTT connection...connected
Humidity: 56.00 %      Temperature: 29.30 °C 82.94 °F      Heat index: 29.43 °C
Humidity: 55.00 %      Temperature: 27.80 °C 82.04 °F      Heat index: 28.47 °C
Humidity: 49.00 %      Temperature: 27.20 °C 80.96 °F      Heat index: 27.54 °C
Humidity: 47.00 %      Temperature: 26.70 °C 80.06 °F      Heat index: 26.99 °C
Message arrived on topic: room/lamp. Message: true
Changing Room lamp to true
Message arrived on topic: room/lamp. Message: false
Changing Room lamp to false
Humidity: 47.00 %      Temperature: 26.30 °C 79.34 °F      Heat index: 26.67 °C
Humidity: 47.00 %      Temperature: 26.00 °C 78.80 °F      Heat index: 25.88 °C
Humidity: 47.00 %      Temperature: 25.60 °C 78.08 °F      Heat index: 25.44 °C
Message arrived on topic: room/lamp. Message: true
Changing Room lamp to true
Humidity: 49.00 %      Temperature: 25.30 °C 77.54 °F      Heat index: 25.14 °C
Humidity: 49.00 %      Temperature: 25.10 °C 77.18 °F      Heat index: 24.94 °C
Humidity: 49.00 %      Temperature: 25.10 °C 77.18 °F      Heat index: 24.94 °C
Humidity: 50.00 %      Temperature: 24.70 °C 76.46 °F      Heat index: 24.53 °C
Humidity: 49.00 %      Temperature: 24.70 °C 76.46 °F      Heat index: 24.50 °C
Humidity: 50.00 %      Temperature: 24.60 °C 76.28 °F      Heat index: 24.42 °C
Humidity: 50.00 %      Temperature: 24.60 °C 76.28 °F      Heat index: 24.42 °C
Humidity: 50.00 %      Temperature: 24.60 °C 76.28 °F      Heat index: 24.42 °C
Humidity: 50.00 %      Temperature: 24.60 °C 76.28 °F      Heat index: 24.42 °C
Humidity: 50.00 %      Temperature: 24.60 °C 76.28 °F      Heat index: 24.42 °C
Humidity: 50.00 %      Temperature: 24.60 °C 76.28 °F      Heat index: 24.42 °C
Humidity: 50.00 %      Temperature: 24.60 °C 76.28 °F      Heat index: 24.42 °C
Humidity: 50.00 %      Temperature: 24.60 °C 76.28 °F      Heat index: 24.42 °C
Humidity: 50.00 %      Temperature: 24.60 °C 76.28 °F      Heat index: 24.42 °C
]
[ Défilement automatique [ Afficher l'horodatage
Nouvelle ligne 115200 baud Effacer la sortie
```

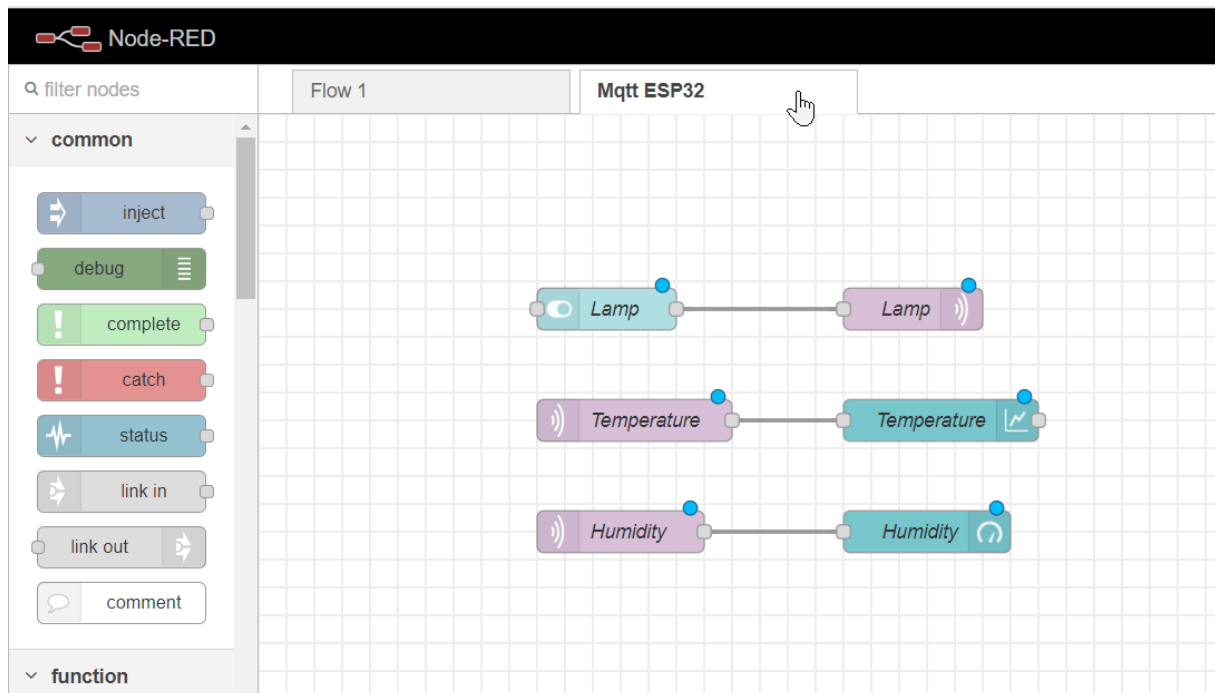
La figure suivante montre le schématique à réaliser pour mettre en place le protocole MQTT. Pour faire gagner du temps ce schématique est disponible sur le GitHub sous le nom « **flows_mqtt.json** » et disponible depuis ce lien : https://github.com/charlesladzro/Protocole-MQTT_COAP/raw/main/flows_mqtt.zip. Il suffit de le télécharger et l'importer dans votre node-red local.



Tout d'abord, on extrait du fichier zip qu'on vient de telecharger le fichier « **flows_mqtt.json** ». Pour l'importer dans node-red, suivant la figure ci-dessus, on clique sur les trois traits qui sont à côté de « Deploy » puis sur Import ; ensuite sur « Select a file to import ». Une fois le fichier choisit, on clique sur Import. (Voir figure ci-dessous).

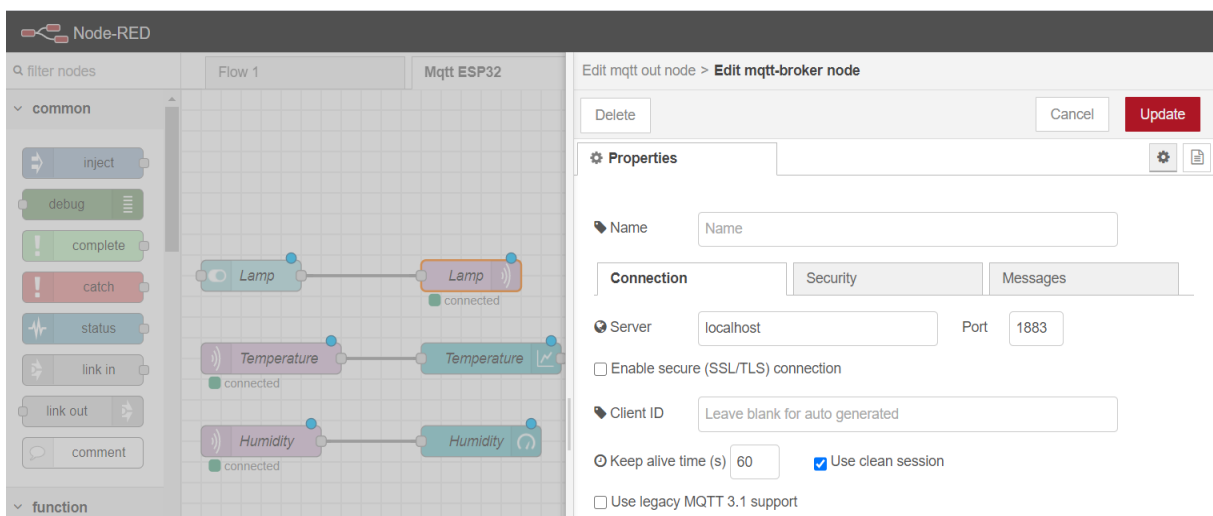
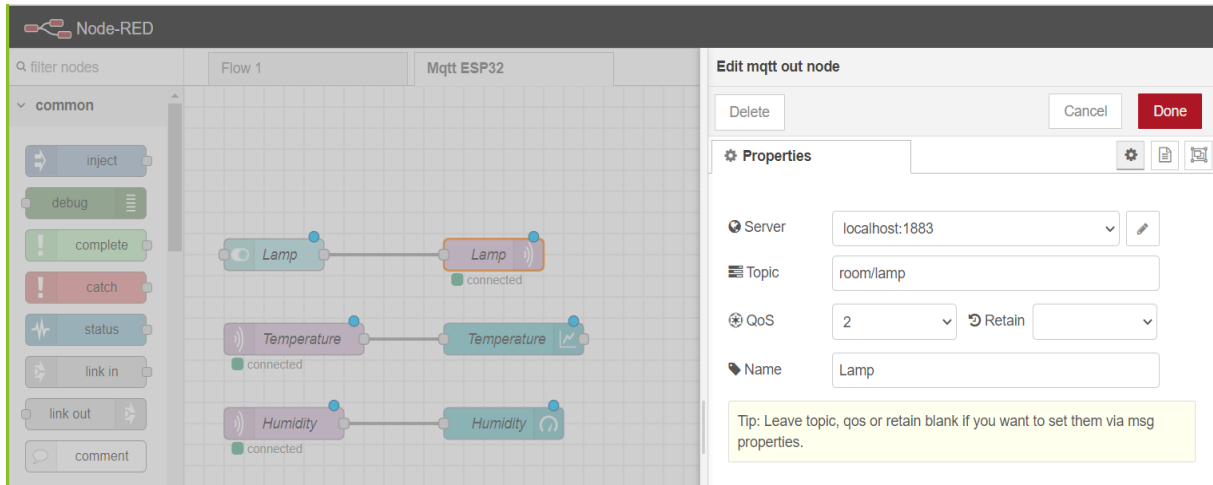


Après l'importation on a donc la figure suivante :



Il faudra aussi préciser l'adresse IP du broker ainsi que le port de communication sur les émetteurs et récepteurs MQTT (les blocs en violet – Lamp, Temperature, Humidity). Dans notre cas, le broker est localhost et sur le port 1883

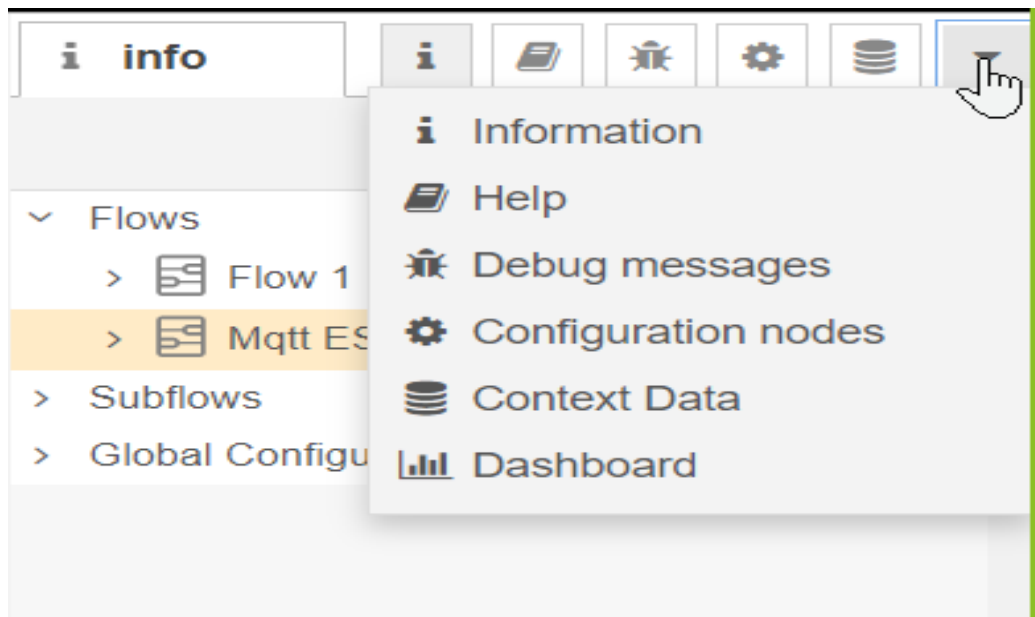
Pour ce faire, l'on fait un double clic sur l'outil ajouté et éditer l'adresse IP en cliquant sur le petit crayon situé sur la ligne de **Server**. (Voir les deux figures suivantes)



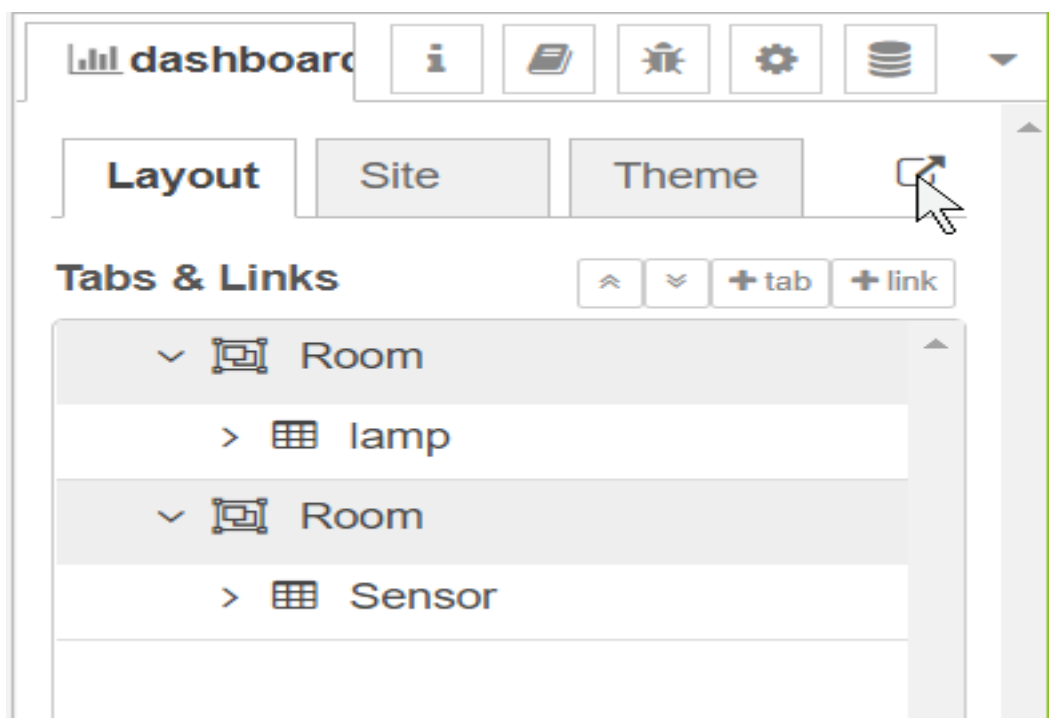
Après mise en place des outils nécessaires, il faut déployer l'application en appuyant sur le bouton rouge en haut à gauche de l'application.



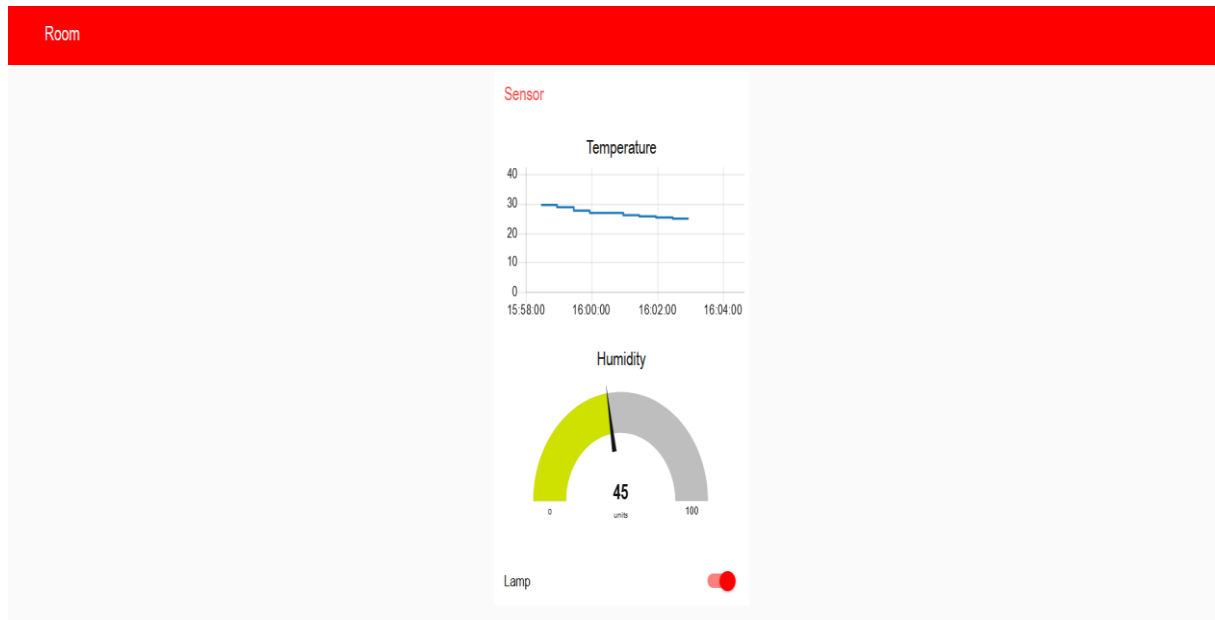
Pour visualiser les résultats graphique ou numérique cliquez la petite flèche en haut à droite situé sur la ligne d'**Info** puis sur Dashboard (voir figure ci-dessous)



Ensuite cliquez sur le petit carré avec la flèche qui est à gauche (voir figure suivante)



Sur l'interface, on voit bien l'affichage de la température de l'humidité qui sont récupérés par la communication en uplink mais aussi un bouton qui permet de gérer la communication downlink pour allumer ou éteindre la led de la carte ESP32.



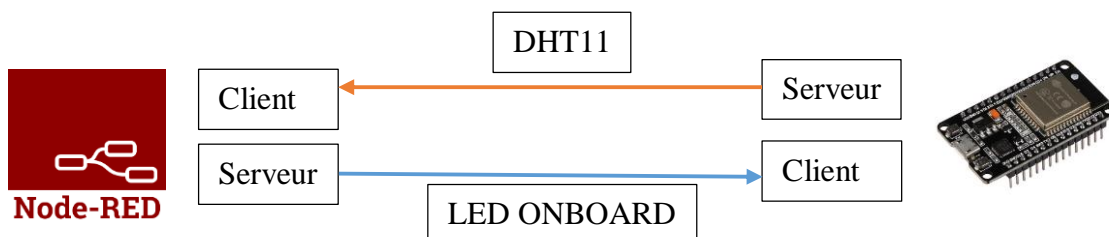
6. Protocole COAP

a. L'objectif de la mise en place du protocole

L'objectif est de montrer 2 démos qui permettent de vérifier la connexion bidirectionnelle :

- Dans le sens montant (uplink) remonter et afficher sur un graphique la température et l'humidité par l'intermédiaire du capteur DHT11.
- Dans le sens descendant (downlink) activer une LED depuis une interface web.

Ce protocole est basé sur une architecture client-serveur. Ce qui fait que l'ESP32 et le node-red seront tous deux à la fois un serveur coap et un client coap (voir figure suivante).



b. Programmation de la carte ESP32

- **Installation de la bibliothèque CoAP-simple-library**

La bibliothèque **CoAP-simple-library** (<https://github.com/hirotakaster/CoAP-simple-library>) permet de créer un serveur ou un client avec la carte ESP32 ou Arduino.

- 1) Télécharger la bibliothèque **CoAP-simple-library** en utilisant ce lien : <https://github.com/hirotakaster/CoAP-simple-library/archive/master.zip> . Vous devriez avoir un dossier .zip dans vos téléchargements
- 2) Décompressez le dossier .zip et vous devriez obtenir le dossier CoAP-simple-library-master
- 3) Renommez votre dossier de CoAP-simple-library-master en CoAP-simple-library
- 4) Déplacez le dossier CoAP-simple-library-master vers votre dossier de bibliothèque d'installation Arduino IDE
- 5) Ensuite, rouvrez votre IDE Arduino

- **Importation du code arduino à exécuter**

Le code arduino à téléverser dans l'ESP32 est disponible sur le GitHub sous le nom de « **esp32_coap.ino** » avec lien suivant : https://github.com/charlesladzro/Protocole-MQTT_COAP/raw/main/esp32_coap.zip . Après avoir télécharger le code, extraire le fichier « **esp32_coap.ino** » et ouvrez le dans le logiciel Arduino. Nous allons faire quelques changements.

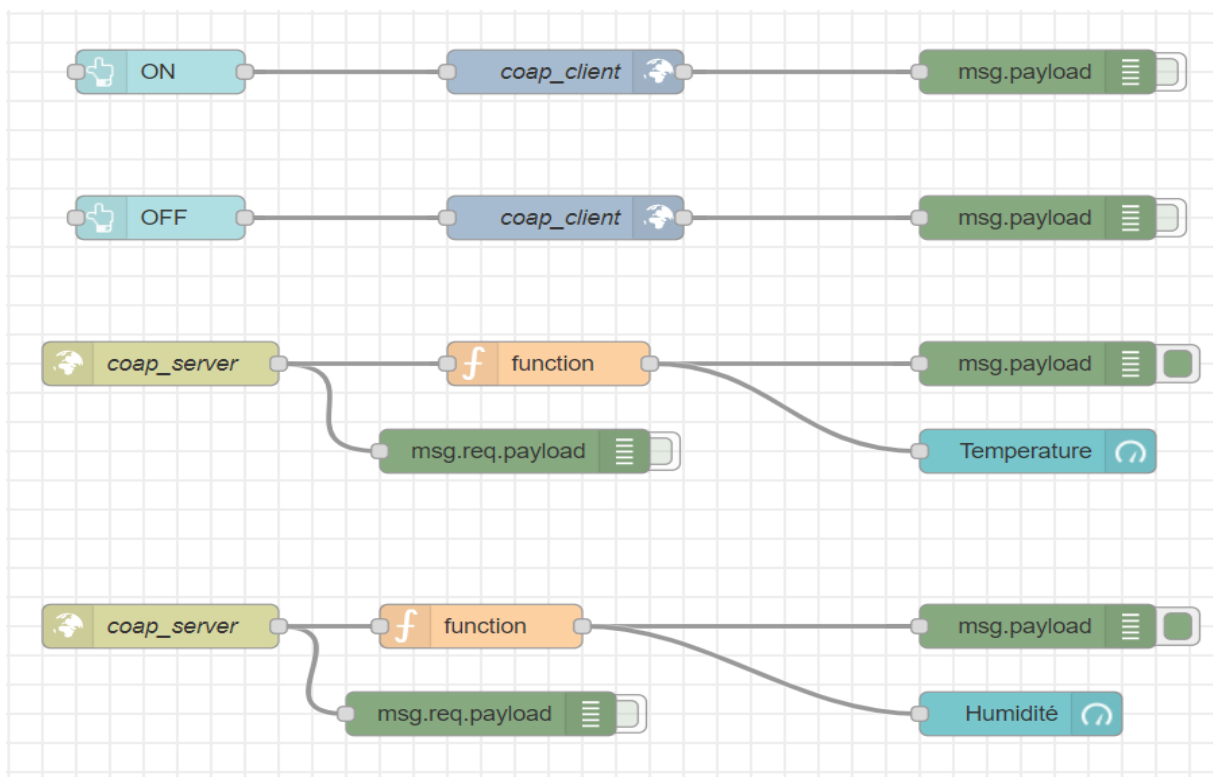
```
// Initialize DHT sensor.  
DHT dht(DHTPin, DHTTYPE);  
  
const char* ssid      = "YOUR SSID";  
const char* password = "YOUR PASSWORD";  
  
// Node-red coap server  
IPAddress node_red_coap_server_ip = IPAddress(192,168,43,139);  
  
// Node-red coap port  
IPAddress node_red_coap_server_port = 5683;
```

- Changer « YOUR SSID » par le nom de votre wifi et « YOUR PASSWORD » par votre mot de passe
- Changer aussi « 192,168,43,139 » par l'adresse IP de votre ordinateur sur laquelle node-red est installé.

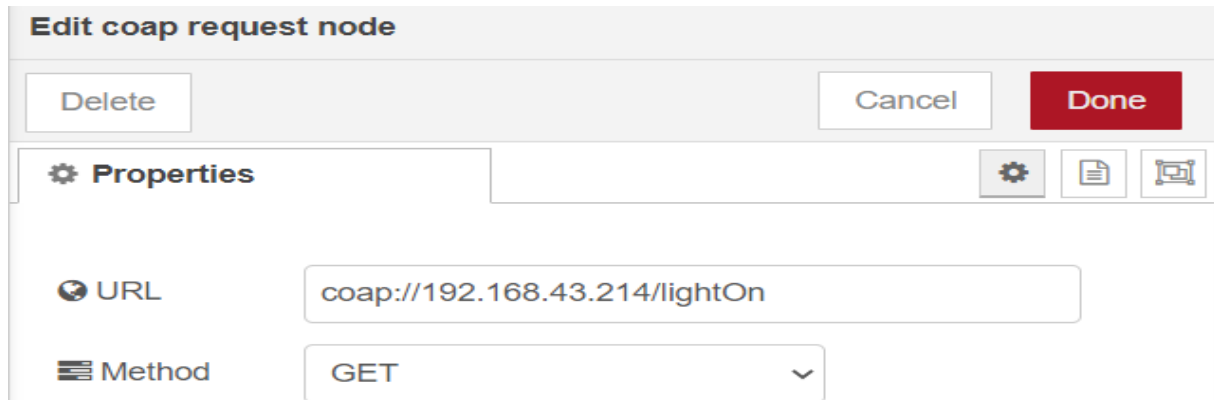
Après ces changements, compiler et téléverser le programme sur l'esp32, vous pouvez maintenant aller configurer le node-red.

c. Réalisation de l'IHM dans node-red

La figure suivante montre le schématique à réaliser pour mettre en place Coap. Pour faire gagner du temps ce schématique est disponible sur le GitHub sous le nom « **flows_coap.json** » et disponible depuis ce lien : https://github.com/charlesladzro/Protocole-MQTT_COAP/raw/main/flows_coap.zip . Il suffit de le télécharger et l'importer dans votre node-red local. (Confère le point 5.e sur la page 13)



Il faut impérativement que votre ordinateur et votre ESP32 soit dans le même réseau wifi. En double-cliquant sur le nœud « **coap_client** », cette fenêtre s'ouvre et il faudra changer l'adresse IP « 192.168.43.214 » par l'adresse IP qu'à votre ESP32 (voir figure suivante).



Edit coap request node

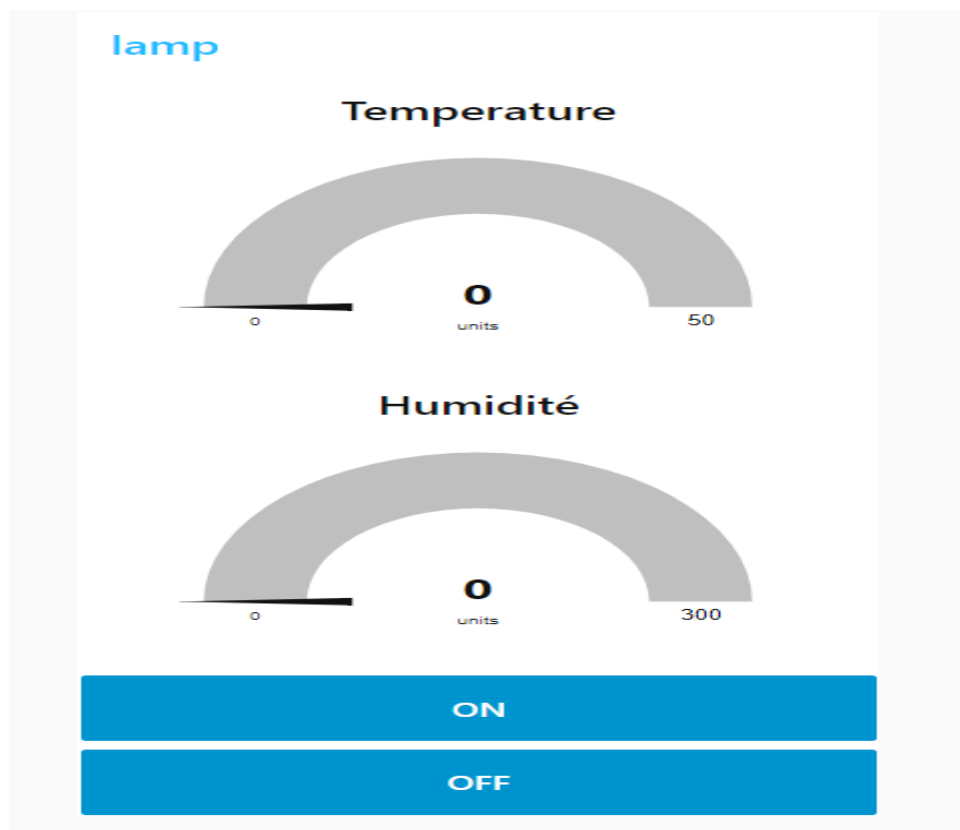
Delete Cancel Done

Properties

URL coap://192.168.43.214/lightOn

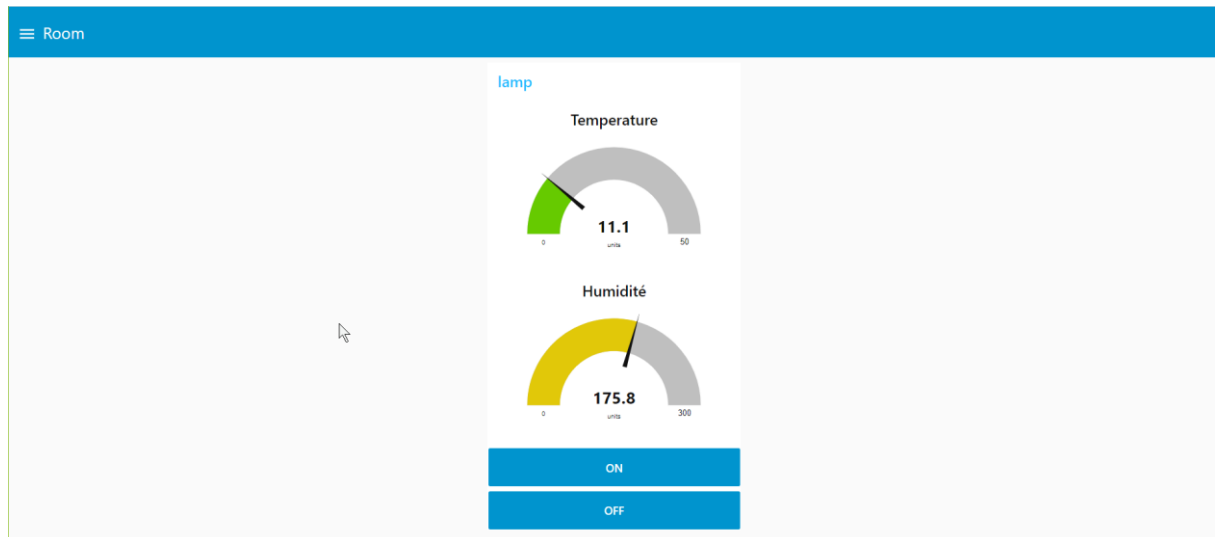
Method GET

En appuyant sur « Deploy », et en ouvrant le Dashboard, on a la figure suivante :



Le bouton ON permet d'allumer le led onboard de l'ESP32, le bouton OFF permet de l'éteindre. Et les jauges permettent de visualiser les valeurs de la température et de l'humidité envoyé par la DHT11.

Sur l'interface, on voit bien l'affichage de la température de l'humidité qui sont récupérés par la communication en uplink mais aussi les boutons qui permettent de gérer la communication downlink pour allumer ou éteindre la led de la carte ESP32.



Nous sommes arrivés à la fin de notre tutoriel qui explique la mise en place du protocole MQTT et COAP. Le lien du GitHub du projet complet est donc le suivant :

https://github.com/charlesladzro/Protocole-MQTT_COAP