Rapport

1 - Serveur Web

- On modifie l'example de base de la libraire
  Webserver , on modife l'adresse MAC et IP , puis on affiche un message personnalisé ou les données des capteurs sur le navigateur Web ;

2 - Serveur COAP

- CoAP - Constrained Application Protocol

COAP est un protocole de transfert Web optimisé pour les périphériques et réseaux contraints utilisés dans les réseaux de capteurs sans fil pour former l'Internet des objets.
On va mettre en place un serveur CoAP sur le MKR100 qui va transmettre des données à un client CoAP . Ces données seront extraites des capteurs de luminosité , humidité et température.

- **Client CoAP** :

- Installer le client sur la machine Ubuntu :

```
sudo apt install libcoap-1-0-bin
```

- **Serveur CoAP** :

- Code Arduino modifié à partir de l'exemple de base : ajout des fonctions callback pour la température , humidité , luminosité .
- Configurez l'Arduino pour se connecter à votre réseau Wifi ou ethernet .

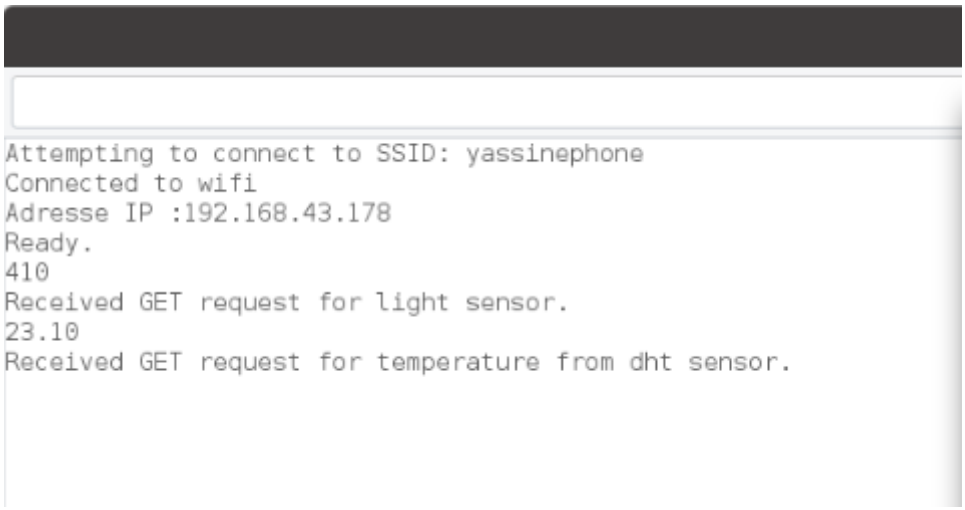
- **Procédure/Test**

- On lance le point d'accès Wifi , ou on connecte l'ethernet à l'arduino
- On se connecte à ce réseau puis on lance le code Arduino
- Le serveur CoAP commence , le code nous renvoie l'adresse IP du serveur
- Cette adresse IP sera utilisée pour récuperer les données (voir ci-dessous).
- Récupérer lee données du serveur on lance

```
// sudo coap-client coap://ADRESSE_IP_DU_SERVEUR_COAP/CALLBACK
sudo coap-client coap://192.168.43.178/temperature
sudo coap-client coap://192.168.43.178/luminosite
sudo coap-client coap://192.168.43.178/humidite
```

```
lambarki@vb-asus-yass:~$ sudo coap-client coap://192.168.43.178/luminosite
410 := Soir
lambarki@vb-asus-yass:~$ sudo coap-client coap://192.168.43.178/temperature
23
lambarki@vb-asus-yass:~$ sudo coap-client coap://192.168.43.178/temperature
```

- On remarque que effectivement le serveur a recu des requetes GET et leur a répondu avec les données nécessaires



3 - Proxy HTTP/CoAP

- On va mettre en place un proxy sur le raspberry .
- Le Raspberry envoie des requêtes HTTP sur le proxy et l'Arduino lui communique en retour les valeurs souhaitées . Le proxy récupère ces données grâce au code coaphttp.py et renvoie une réponse HTTP .
- Plusieurs configurations sont possibles , mais on a choisi de mettre en place , la configuration bonifiée :
  - Connexion entre PC et Raspberry en Ethernet
  - Connexion entre Raspberry et Arduino à l'aide du point d'accès WiFi du Rpi .

Pré-requis :

- Flask est un micro framework de développement web écrit en Python.
- On va créer un serveur avec Flask et on va installer CoAPthon pour permettre au serveur de recevoir des requêtes CoAP .
- Installez pip

```
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py && python get-pip.py
```

- Installer Flask

```
pip install -U Flask
```

- Installer Coapthon

```
pip install -U Coapthon
```

- Remarque:
  - Si vous utilisez python 3 , il faut remplacer pip par pip3 et python par python3 .
- Enregister le code python :
  - Sur ce fichier on change l'adresse du serveur CoAP = 192.168.50.184
  - ainsi que le host = 10.42.0.203(rpi interface du côté http)

```
// Copier le code source python
// sur le ssh
cat > httpcoap.py
// coller sur le ssh CTRL-V
// + CTRL-C  pour sortir de l'editeur
cat httpcoap.py
// s'assurer que le fichier a bien été enregistré
```

Comment mettre en place le point d'accès Raspberry ?

- Au début j'ai utilisé un package RaspAp que je déconseille fortement à cause de son fichier de désinstallation qui est défectueux et qui m'a coûté un formatage du Raspberry la veille du rendu .
- Par contre , je conseille l'utilisation de Hostapd et Dnsmasq pour une telle pratique .

```
// Comme avant tout installation
sudo apt-get update && sudo apt-get upgrade
// Install hostapd
sudo apt-get install hostapd
// Install dnsmasq
sudo apt-get install dnsmasq
// Disable AP while installation
sudo systemctl disable hostapd
sudo systemctl disable dnsmasq
```

- Maintenant on va configurer Hostapd

```
// Créer le fichier
sudo nano /etc/hostapd/hostapd.conf
// et y ajouter les paramètres suivants du AP
interface=wlan0
driver=nl80211
ssid=RPiHotSpot
```

```
hw_mode=g
channel=6
wmm_enabled=0
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=1234567890
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

- Màj des changements

```
sudo nano /etc/default/hostapd
// et Changer:
#demon_conf=""
à
demon_conf="/etc/hostapd/hostapd.conf"
```

- Configuration de dnsmasq

```
// Créer le fichier pour permettre au Pi
// d'agir comme un routeur et assigner les adresses
sudo nano /etc/dnsmasq.conf
// copy-paste
#RPiHotspot Config
#stop DNSmasq from using resolv.conf
no-resolv
#Interface to use
interface=wlan0
bind-interfaces
dhcp-range=10.0.0.3,10.0.0.20,12h
```

Cf : https://www.raspberryconnect.com/61-uncategorised-1/148-rpi3-auto-wifi-hotspot-if-no-internet-oldscript

Remarque importante
Il est important de passer swicther facilement le Rpi entre point d'accès et client wifi .
Pour cela :

- Passage de AP -> client

```
// Ouvrir ce fichier
sudo nano /etc/default/hostapd
```

```
// Commenter cette ligne
demon_conf="/etc/hostapd/hostapd.conf"
// Désactiver le AP
sudo systemctl disable hostapd
sudo systemctl disable dnsmasq
// Enregistrer les modif
sudo reboot
```

- Passage de Client -> AP

```
// Ouvrir ce fichier
sudo nano /etc/default/hostapd
// Décommenter cette ligne
#demon_conf="/etc/hostapd/hostapd.conf"
// Activer le AP
sudo systemctl enable hostapd
sudo systemctl enable dnsmasq
// Enregistrer les modif
sudo reboot
```

Comment s'assurer que le point d'accès Raspberry marche bien évidemment ?

Si jamais , cela ne marche pas , comment pourrait-on connecter le Rpi et l'arduino ?

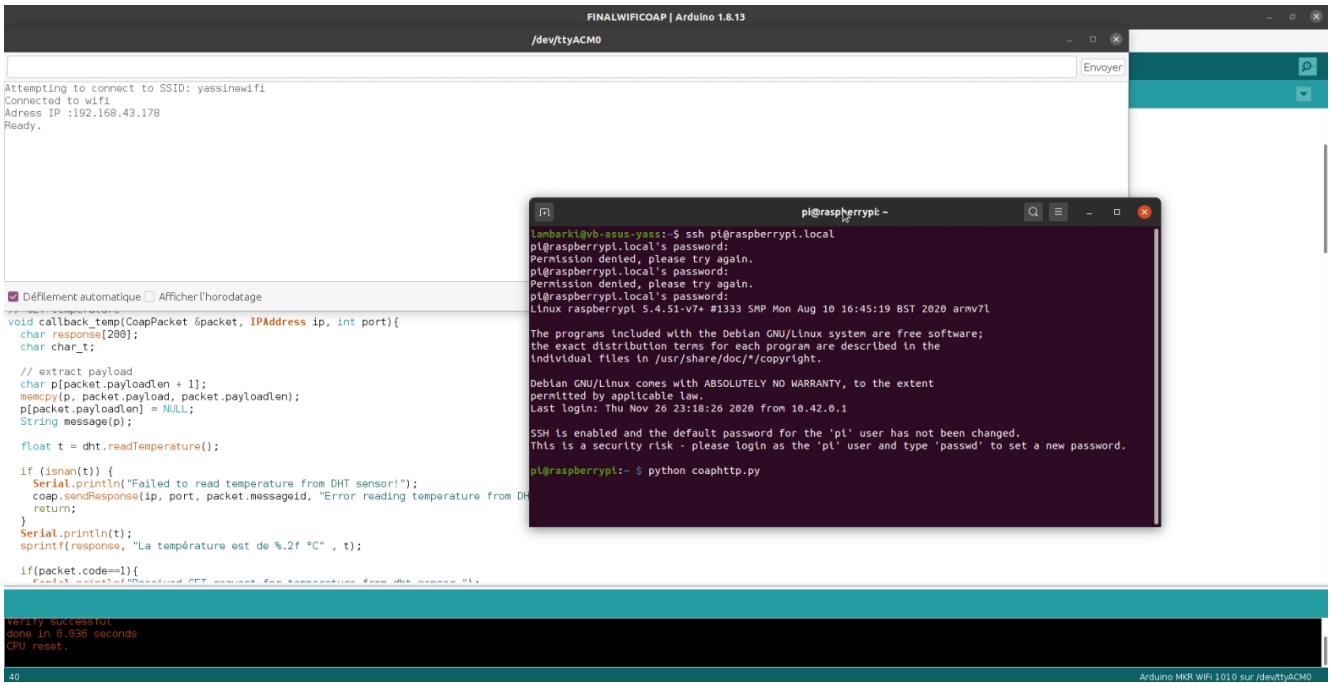
Pour lier l'Arduino et le Rpi , on peut les connecter à notre point d'accès Wifi depuis le smartphone .
Pour connecter l'Arduino à un Wifi , cela a été déja vu; cf le tp d'avant .

Pour connecter le Raspberry à un Wifi . Il suffit de rentrer la carte SD de celui-ci , et de créer un fichier wpa_supplicant.conf

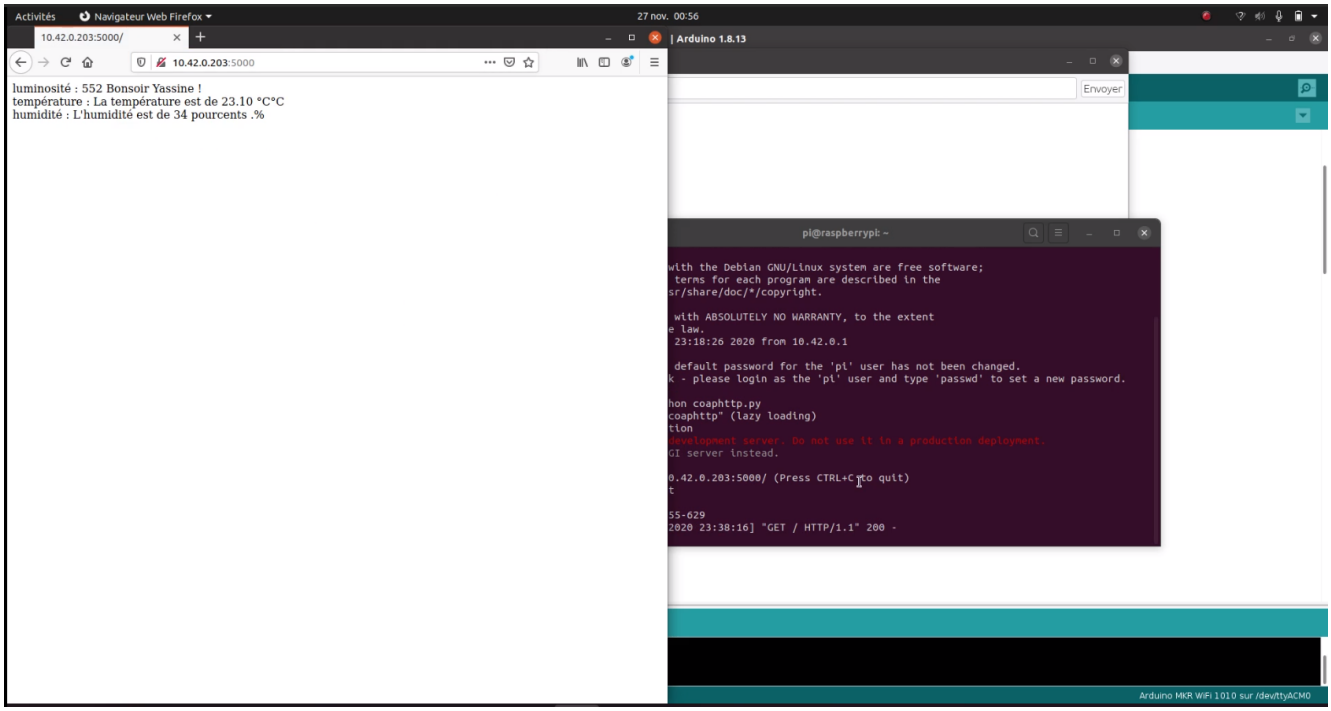
```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

Puis ajouter les settings du Wifi

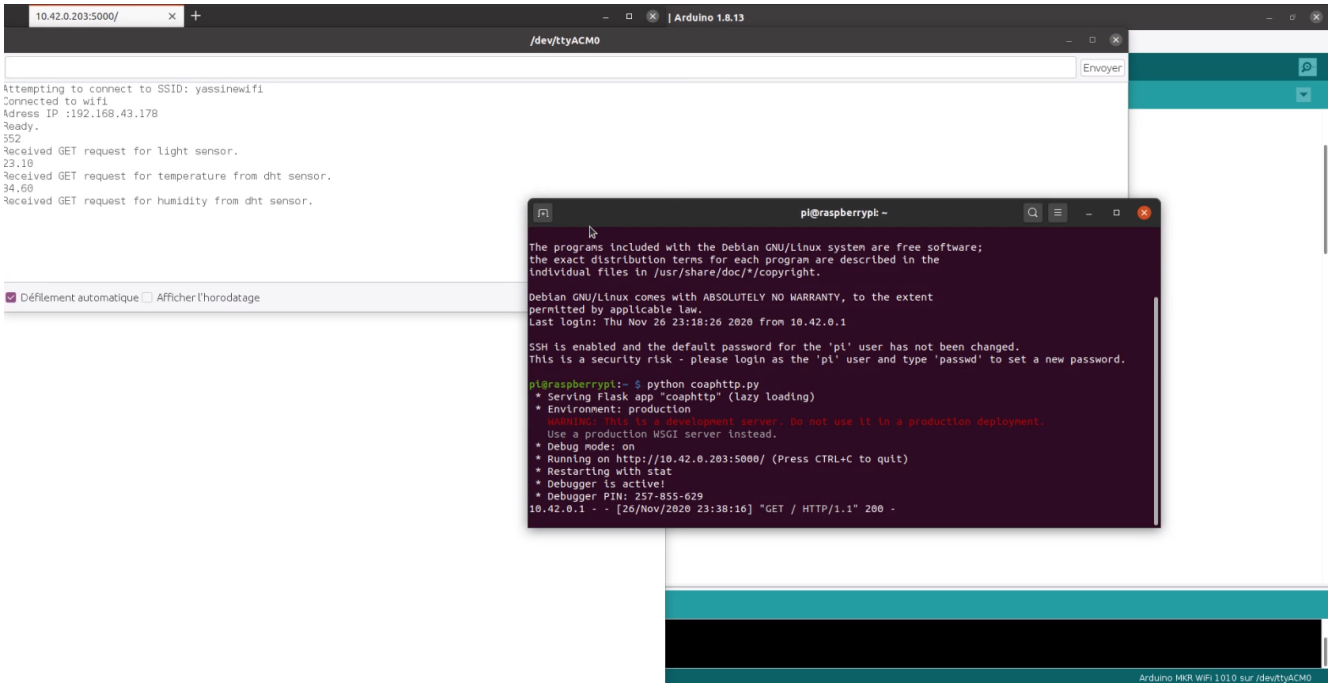
```
country=fr
update_config=1
ctrl_interface=/var/run/wpa_supplicant

network={
 scan_ssid=1
 ssid="nomDeLaBox"
 psk="cléDeSécurité"
}
```

- **Test**
- On configure le Rpi en tant que point d'acceès puis on reboot .
- On lance le code Arduino , le serveur CoAP se connecte à ce réseau .
- 

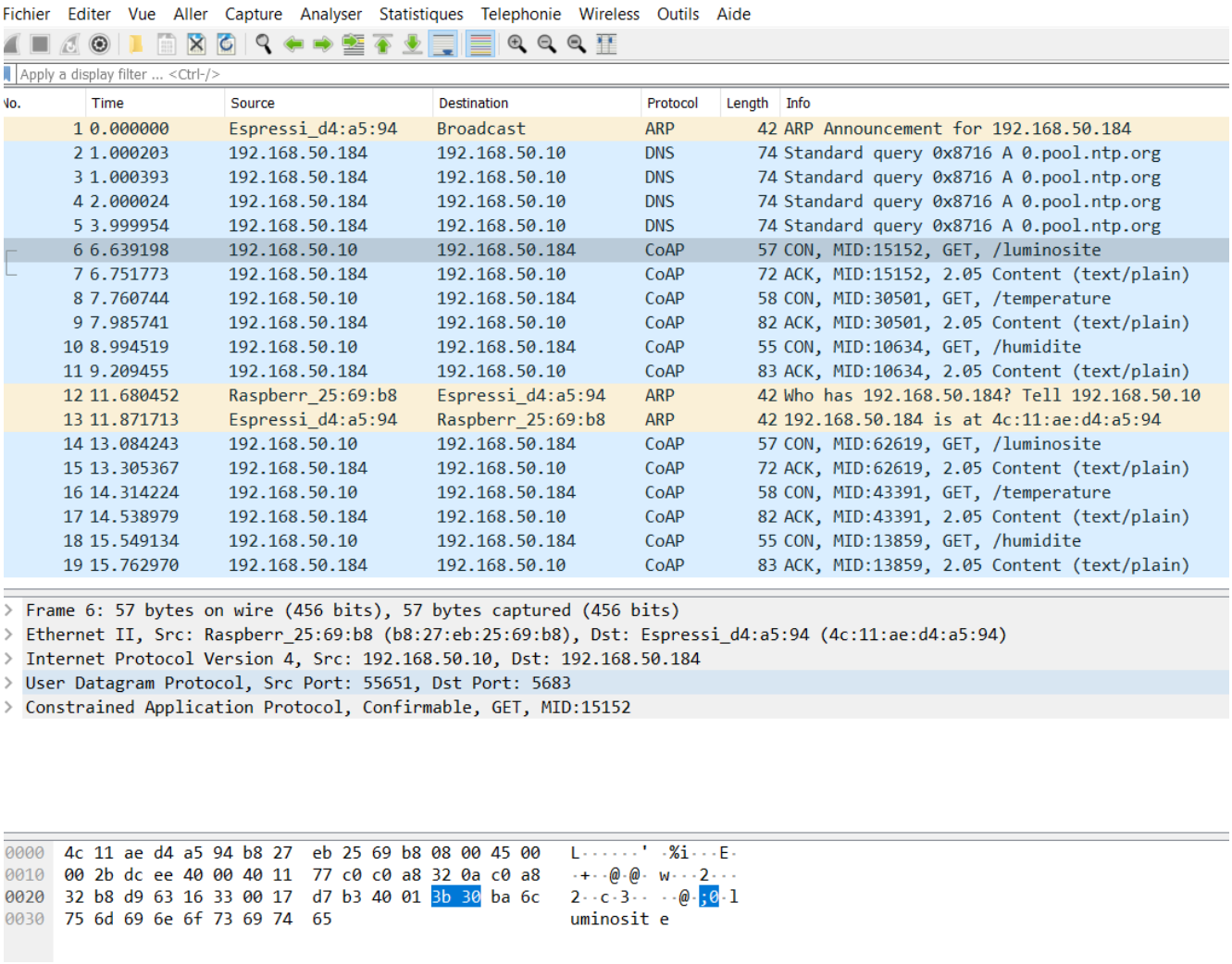
- On lance le code Python , et on se connecte à l'adresse affichée

- 

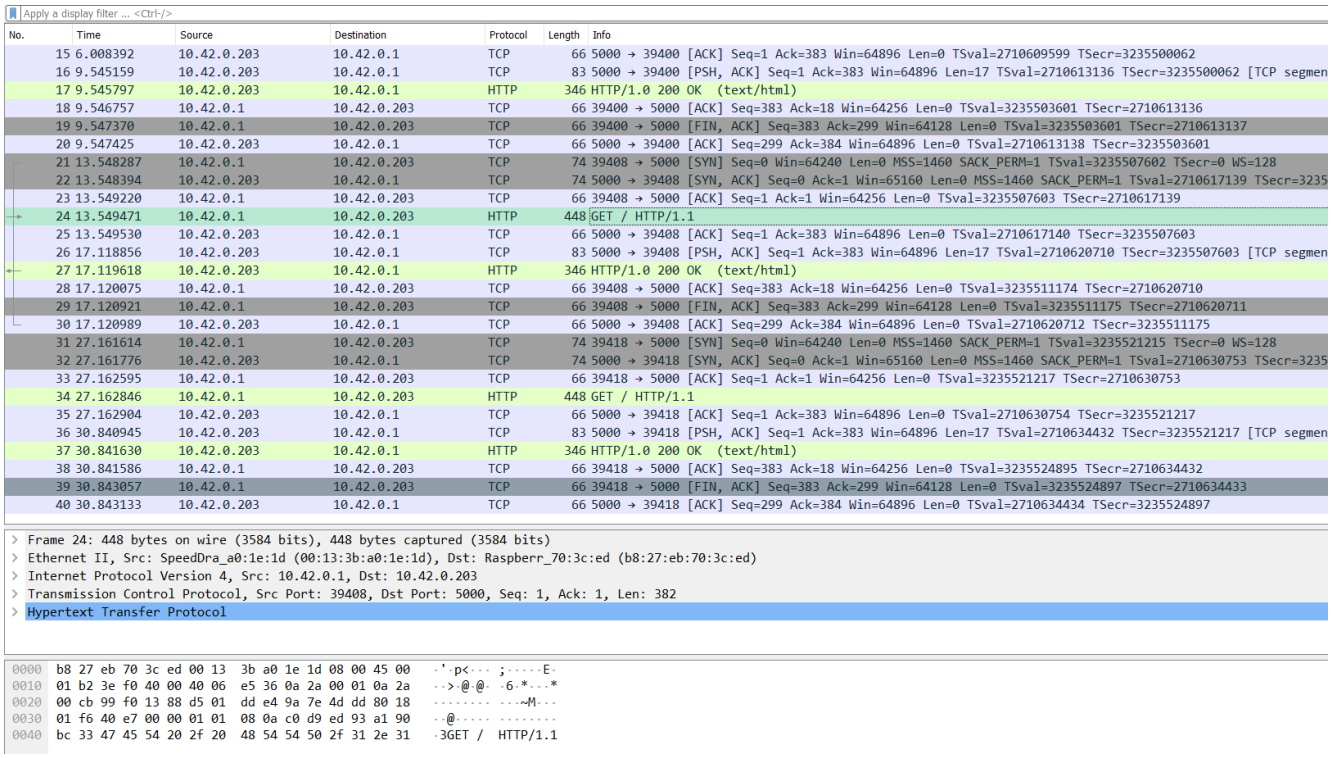
- Côté Arduino :

- 

- Une capture Wireshark montre les requêtes CoAP échangées et les données demandées : (au début le rpi ne reconnait pas le serveur cf.(who has ...?))



- L'échange HTTP aussi:



4 - Serveur MQTT

- **MQTT** est un protocole très léger et donc tout aussi rapide qui a révolutionné le monde de l'IoT .

- Quand on parle MQTT, on parle de connexion **client-serveur avec abonnement**. Concrètement les équipements connectés publient et/ou s'abonnent à un **topic** qui référence les messages et les communique aux abonnés.

Installer un broker Mosquitto sur le Raspberry Pi

```
Se connecter en SSH
// Installer mosquitto
sudo apt-get install mosquitto
// Mise à jour et paquets manquants
sudo apt update && sudo apt upgrade
// Vérifier que mosquitto est tout fonctionnel
systemctl status mosquitto
// Vous pouvez aussi installer un clien pour faire des tests
sudo apt-get install mosquitto-clients
```

- C'est terminé, votre **serveur MQTT** local est **en place et prêt** à collecter les messages

Note additionnelle :

- Pour ajouter plus sécurité à votre broker MQTT ,il est possible de lui ajouter une authentification

```
sudo mosquitto_passwd -c /etc/mosquitto/passwd NOM_UTILISATEUR
```

- Interdiser l'accès anonyme au broker en ajoutant les lignes suivantes au fichier /etc/mosquitto/mosquitto.conf

```
sudo nano /etc/mosquitto/mosquitto.conf
allow_anonymous false
password_file /etc/mosquitto/passwd
```

- Puis redémarrer le serveur pour que les paramètres soient pris en compte

```


systemctl restart mosquitto
```

Installer Node-RED

- Node-RED est un puissant outil open source pour créer des applications Internet des objets (IoT) dans le but de simplifier le composant de programmation. Il utilise une programmation visuelle qui vous permet de connecter des blocs de code, appelés nœuds, ensemble pour effectuer une tâche.
- Installer Node-RED

```
pi@raspberry:~ $ bash <(curl -sL https://raw.githubusercontent.com/node-red/raspbian-deb-package/master/resources/update-nodejs-and-nodered)
```

- Installer Node-RED Dashboard

```
pi@raspberry:~ $ node-red-stop
pi@raspberry:~ $ cd ~/.node-red
pi@raspberry:~/.node-red $ npm install node-red-dashboard
```

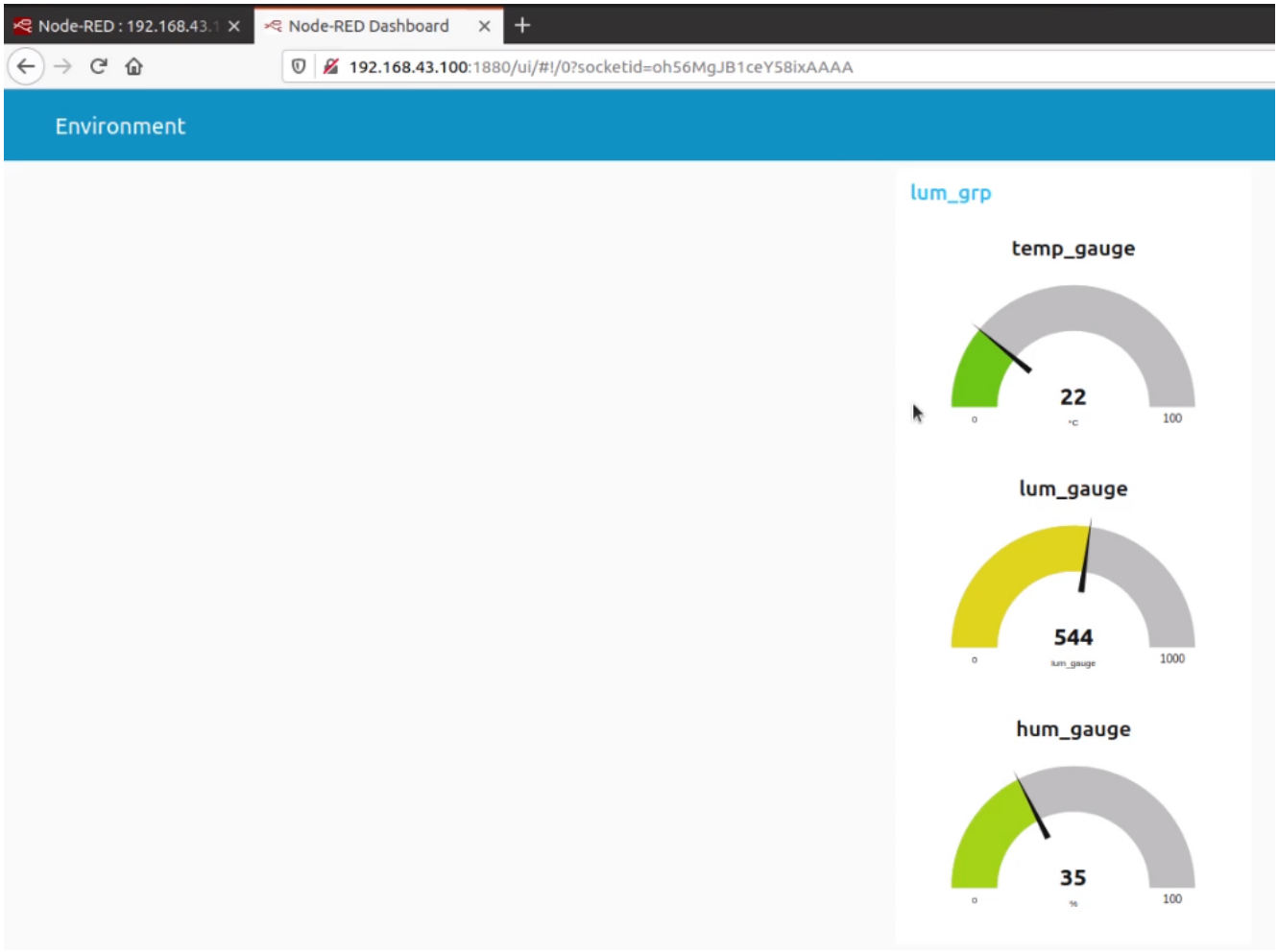
- Redémarrer pour que l'installation soit prise en compte

```
pi@raspberry:~ $ sudo reboot
```

- Tester l'installation en lançant node-red et en ouvrant l'adresse suivante dans un navigateur :
  `* http://VOTE_RASPBERRYPI_ADRESSE:1880`
  (Cette adresse sera affiché quand vous lancez node-red ou node-red-start)
- Sur cette interface graphique , créer les noeuds MQTT selon les données qu'on souhaite afficher , humidité , luminosité ou température.
- Pour connecter ces noeuds au serveur mqtt , on vous conseille de suivre ce tutoriel : https://medium.com/@varuldcube100/display-temperature-and-humidity-sensor-data-in-node-red-dashboard-using-esp8266-and-mqtt-node-da8b49cdc33b
- **Test**
  1- On établit le serveur MQTT sur Arduino et on se connecte en SSH au Raspberry . Puis , on active le broker , et on lance node-red-start

```
Linux raspberrypi 4.19.75-v7+ #1270 SMP Tue Sep 24 18:45:11 BST 2019 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Nov 25 18:17:15 2020

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set a new password.

pi@raspberrypi:~ $ node-red
25 Nov 18:19:19 - [info]

Welcome to Node-RED
===================

25 Nov 18:19:20 - [info] Node-RED version: v1.2.5
25 Nov 18:19:20 - [info] Node.js  version: v12.19.1
25 Nov 18:19:20 - [info] Linux 4.19.75-v7+ arm LE
25 Nov 18:19:21 - [info] Loading palette nodes
```

  2- On ouvre l'adresse IP affichée par node-red , on voit bien que les données sont affichées en Real-Time



3- On remarque ci-dessous que les données sont bien envoyées depuis le serveur MQTT

```
Attempting to connect to SSID: yassinewifi
Connected to wifi
Adress IP :192.168.43.178
Attempting MQTT connection...connected
550
43.90
21.10
559
36.90
21.20
```

Défilement automatique Afficher l'horodatage