

Rapport

1 - Serveur Web

- On modifie l'exemple de base de la librairie Webserver , on modifie 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 récupérer les données (voir ci-dessous).
- Récupérer les 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 reçu des requêtes GET et leur a répondu avec les données nécessaires

```
Attempting to connect to SSID: yassinephone
Connected to wifi
Adresse IP :192.168.43.178
Ready.
410
Received GET request for light sensor.
23.10
Received GET request for temperature from dht sensor.
```

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 [coaphhttp.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 .
- Enregistrer 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.raspberrypi.com/connect/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 déjà 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

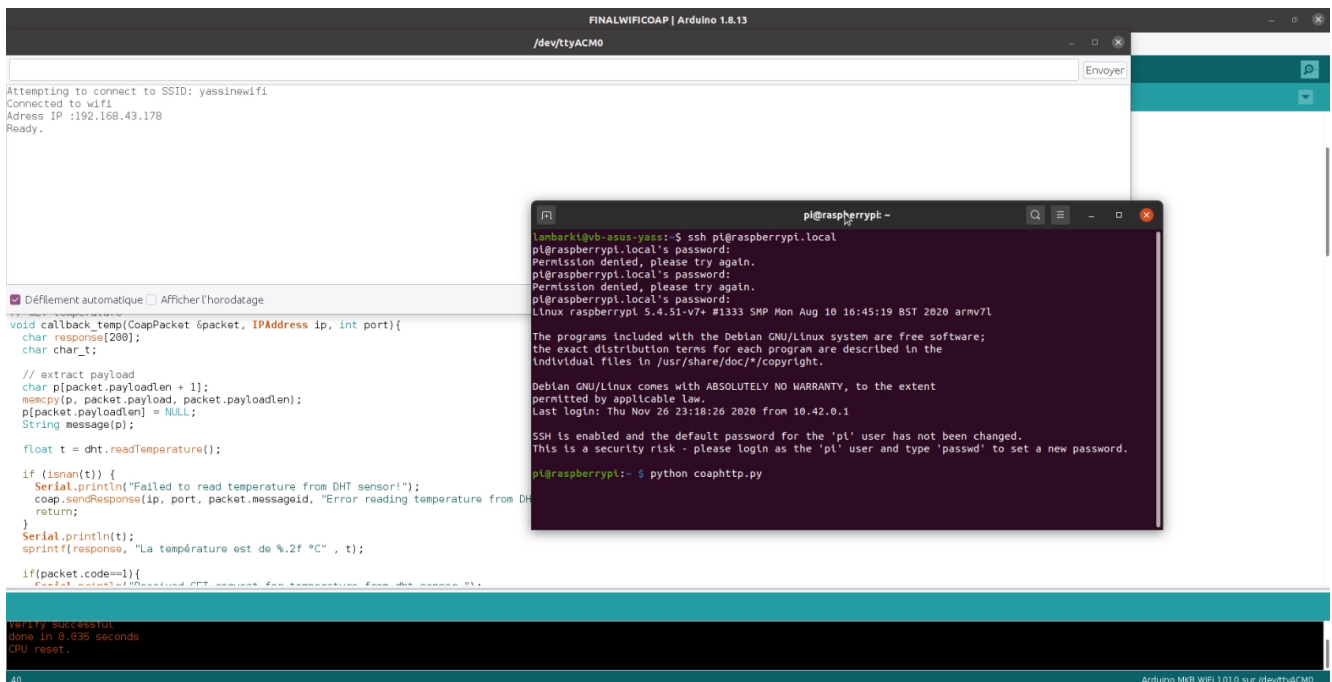
```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

Puis ajouter les settings du Wifi

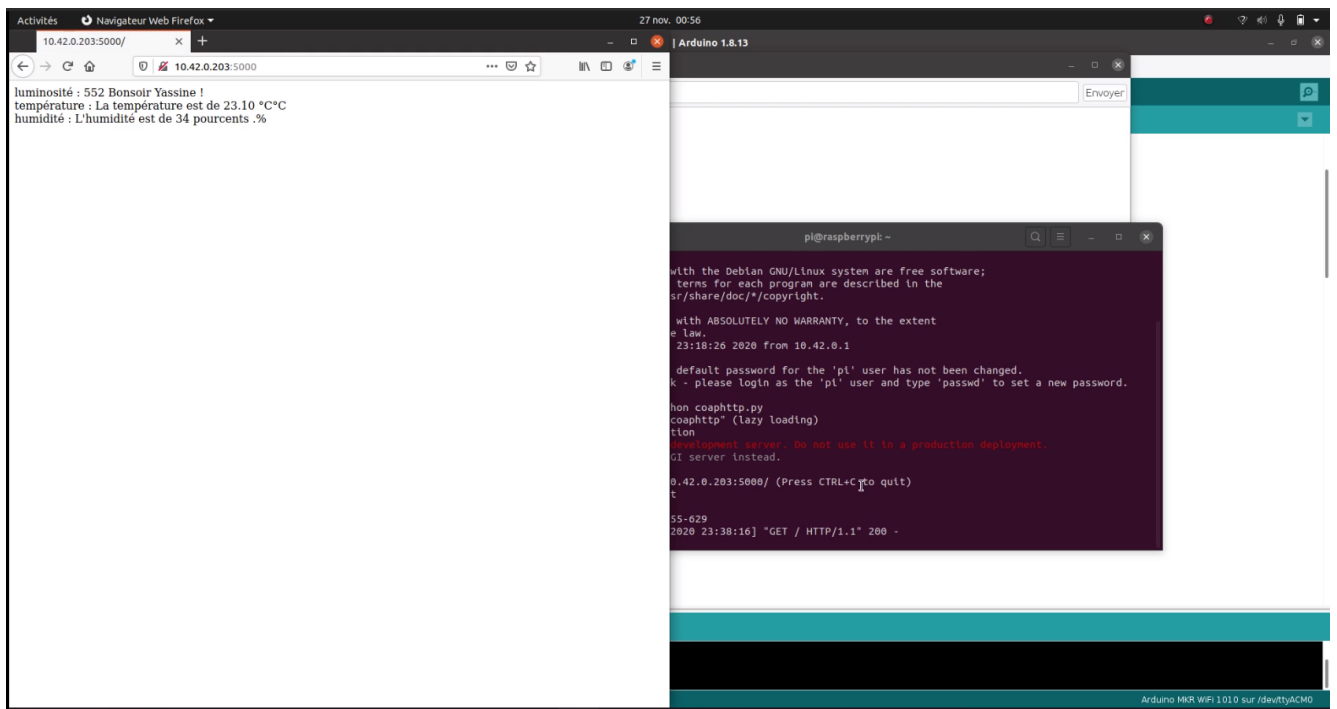
```
country=fr
update_config=1
ctrl_interface=/var/run/wpa_supplicant

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

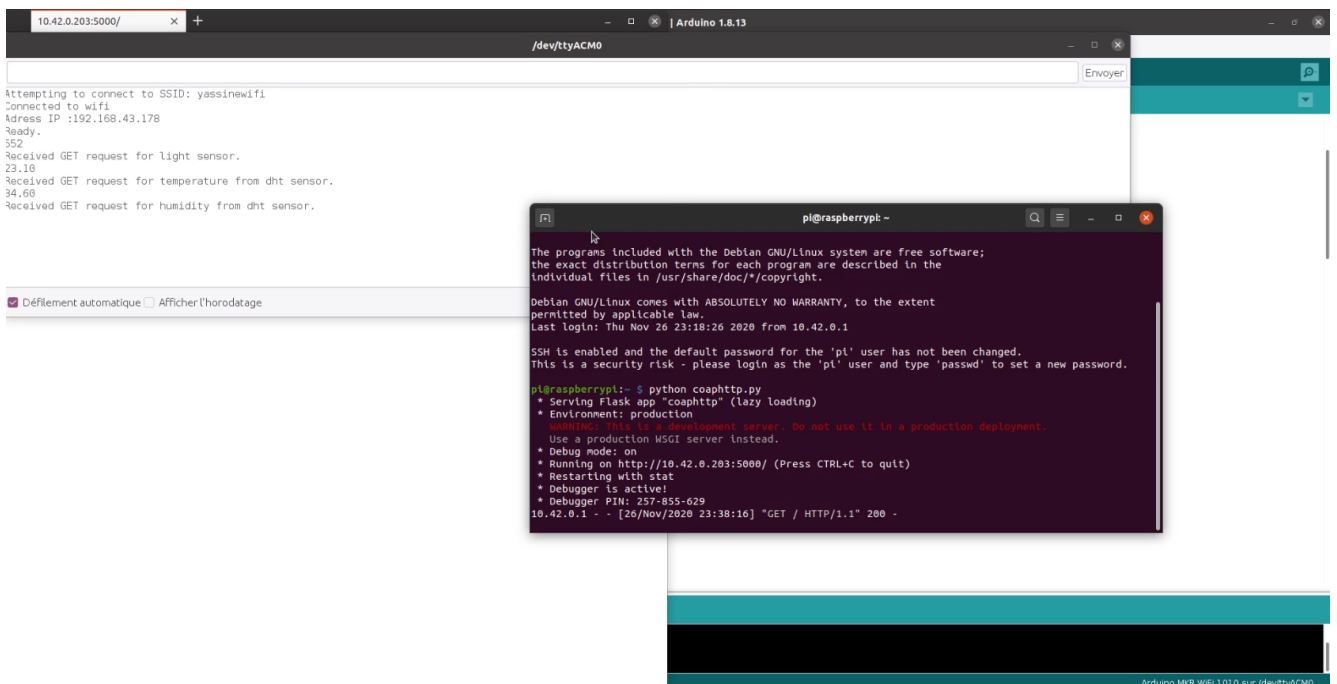
- **Test**
- On configure le Rpi en tant que point d'accès puis on reboot .
- On lance le code Arduino , le serveur CoAP se connecte à ce réseau .



- 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 reconnaît pas le serveur cf.(who has ...?))

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Espressi_d4:a5:94	Broadcast	ARP	42	ARP Announcement for 192.168.50.184
2	1.000203	192.168.50.184	192.168.50.10	DNS	74	Standard query 0x8716 A 0.pool.ntp.org
3	1.000393	192.168.50.184	192.168.50.10	DNS	74	Standard query 0x8716 A 0.pool.ntp.org
4	2.000024	192.168.50.184	192.168.50.10	DNS	74	Standard query 0x8716 A 0.pool.ntp.org
5	3.999954	192.168.50.184	192.168.50.10	DNS	74	Standard query 0x8716 A 0.pool.ntp.org
6	6.639198	192.168.50.10	192.168.50.184	CoAP	57	CON, MID:15152, GET, /luminosite
7	6.751773	192.168.50.184	192.168.50.10	CoAP	72	ACK, MID:15152, 2.05 Content (text/plain)
8	7.760744	192.168.50.10	192.168.50.184	CoAP	58	CON, MID:30501, GET, /temperature
9	7.985741	192.168.50.184	192.168.50.10	CoAP	82	ACK, MID:30501, 2.05 Content (text/plain)
10	8.994519	192.168.50.10	192.168.50.184	CoAP	55	CON, MID:10634, GET, /humidite
11	9.209455	192.168.50.184	192.168.50.10	CoAP	83	ACK, MID:10634, 2.05 Content (text/plain)
12	11.680452	Raspberr_25:69:b8	Espressi_d4:a5:94	ARP	42	Who has 192.168.50.184? Tell 192.168.50.10
13	11.871713	Espressi_d4:a5:94	Raspberr_25:69:b8	ARP	42	192.168.50.184 is at 4c:11:ae:d4:a5:94
14	13.084243	192.168.50.10	192.168.50.184	CoAP	57	CON, MID:62619, GET, /luminosite
15	13.305367	192.168.50.184	192.168.50.10	CoAP	72	ACK, MID:62619, 2.05 Content (text/plain)
16	14.314224	192.168.50.10	192.168.50.184	CoAP	58	CON, MID:43391, GET, /temperature
17	14.538979	192.168.50.184	192.168.50.10	CoAP	82	ACK, MID:43391, 2.05 Content (text/plain)
18	15.549134	192.168.50.10	192.168.50.184	CoAP	55	CON, MID:13859, GET, /humidite
19	15.762970	192.168.50.184	192.168.50.10	CoAP	83	ACK, MID:13859, 2.05 Content (text/plain)

> Frame 6: 57 bytes on wire (456 bits), 57 bytes captured (456 bits)
 > Ethernet II, Src: Raspberr_25:69:b8 (b8:27:eb:25:69:b8), Dst: Espressi_d4:a5:94 (4c:11:ae:d4:a5:94)
 > Internet Protocol Version 4, Src: 192.168.50.10, Dst: 192.168.50.184
 > User Datagram Protocol, Src Port: 55651, Dst Port: 5683
 > Constrained Application Protocol, Confirmable, GET, MID:15152

```

0000 4c 11 ae d4 a5 94 b8 27 eb 25 69 b8 08 00 45 00 L.....' %i---E-
0010 00 2b dc ee 40 00 40 11 77 c0 c0 a8 32 0a c0 a8 +-...@.@- w---2...
0020 32 b8 d9 63 16 33 00 17 d7 b3 40 01 3b 30 ba 6c 2--c-3-- --@;0-1
0030 75 6d 69 6e 6f 73 69 74 65 uminosit e

```

• L'échange HTTP aussi:

No.	Time	Source	Destination	Protocol	Length	Info
15	6.008392	10.42.0.203	10.42.0.1	TCP	66	5000 → 39400 [ACK] Seq=1 Ack=383 Win=64896 Len=0 TSval=2710609599 TSecr=3235500062
16	9.545159	10.42.0.203	10.42.0.1	TCP	83	5000 → 39400 [PSH, ACK] Seq=1 Ack=383 Win=64896 Len=17 TSval=2710613136 TSecr=3235500062 [TCP segment
17	9.545797	10.42.0.203	10.42.0.1	HTTP	346	HTTP/1.0 200 OK (text/html)
18	9.546757	10.42.0.1	10.42.0.203	TCP	66	39400 → 5000 [ACK] Seq=383 Ack=18 Win=64256 Len=0 TSval=3235503601 TSecr=2710613136
19	9.547370	10.42.0.1	10.42.0.203	TCP	66	39400 → 5000 [FIN, ACK] Seq=383 Ack=299 Win=64128 Len=0 TSval=3235503601 TSecr=2710613137
20	9.547425	10.42.0.203	10.42.0.1	TCP	66	5000 → 39400 [ACK] Seq=299 Ack=384 Win=64896 Len=0 TSval=2710613138 TSecr=3235503601
21	13.548287	10.42.0.1	10.42.0.203	TCP	74	39408 → 5000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3235507602 TSecr=0 WS=128
22	13.548394	10.42.0.203	10.42.0.1	TCP	74	5000 → 39408 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=2710617139 TSecr=3235
23	13.549220	10.42.0.1	10.42.0.203	TCP	66	39408 → 5000 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3235507603 TSecr=2710617139
24	13.549471	10.42.0.1	10.42.0.203	HTTP	448	GET / HTTP/1.1
25	13.549530	10.42.0.203	10.42.0.1	TCP	66	5000 → 39408 [ACK] Seq=1 Ack=383 Win=64896 Len=0 TSval=2710617140 TSecr=3235507603
26	17.118856	10.42.0.203	10.42.0.1	TCP	83	5000 → 39408 [PSH, ACK] Seq=1 Ack=383 Win=64896 Len=17 TSval=2710620710 TSecr=3235507603 [TCP segment
27	17.119618	10.42.0.203	10.42.0.1	HTTP	346	HTTP/1.0 200 OK (text/html)
28	17.120075	10.42.0.1	10.42.0.203	TCP	66	39408 → 5000 [ACK] Seq=383 Ack=18 Win=64256 Len=0 TSval=3235511174 TSecr=2710620710
29	17.120921	10.42.0.1	10.42.0.203	TCP	66	39408 → 5000 [FIN, ACK] Seq=383 Ack=299 Win=64128 Len=0 TSval=3235511175 TSecr=2710620711
30	17.120989	10.42.0.203	10.42.0.1	TCP	66	5000 → 39408 [ACK] Seq=299 Ack=384 Win=64896 Len=0 TSval=2710620712 TSecr=3235511175
31	27.161614	10.42.0.1	10.42.0.203	TCP	74	39418 → 5000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=3235521215 TSecr=0 WS=128
32	27.161776	10.42.0.203	10.42.0.1	TCP	74	5000 → 39418 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TSval=2710630753 TSecr=3235
33	27.162595	10.42.0.1	10.42.0.203	TCP	66	39418 → 5000 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3235521217 TSecr=2710630753
34	27.162846	10.42.0.1	10.42.0.203	HTTP	448	GET / HTTP/1.1
35	27.162984	10.42.0.203	10.42.0.1	TCP	66	5000 → 39418 [ACK] Seq=1 Ack=383 Win=64896 Len=0 TSval=2710630754 TSecr=3235521217
36	30.840945	10.42.0.203	10.42.0.1	TCP	83	5000 → 39418 [PSH, ACK] Seq=1 Ack=383 Win=64896 Len=17 TSval=2710634432 TSecr=3235521217 [TCP segment
37	30.841630	10.42.0.203	10.42.0.1	HTTP	346	HTTP/1.0 200 OK (text/html)
38	30.841586	10.42.0.1	10.42.0.203	TCP	66	39418 → 5000 [ACK] Seq=383 Ack=18 Win=64256 Len=0 TSval=3235524895 TSecr=2710634432
39	30.843057	10.42.0.1	10.42.0.203	TCP	66	39418 → 5000 [FIN, ACK] Seq=383 Ack=299 Win=64128 Len=0 TSval=3235524897 TSecr=2710634433
40	30.843133	10.42.0.203	10.42.0.1	TCP	66	5000 → 39418 [ACK] Seq=299 Ack=384 Win=64896 Len=0 TSval=2710634434 TSecr=3235524897

> Frame 24: 448 bytes on wire (3584 bits), 448 bytes captured (3584 bits)
 > Ethernet II, Src: SpeedDra_a0:1e:1d (00:13:b:a0:1e:1d), Dst: Raspberr_70:3c:ed (b8:27:eb:70:3c:ed)
 > Internet Protocol Version 4, Src: 10.42.0.1, Dst: 10.42.0.203
 > Transmission Control Protocol, Src Port: 39408, Dst Port: 5000, Seq: 1, Ack: 1, Len: 382
 > Hypertext Transfer Protocol

```

0000 b8 27 eb 70 3c ed 00 13 3b a0 1e 1d 08 00 45 00 .-p<--- ;-----E-
0010 01 b2 3e f0 40 00 40 06 e5 36 0a 2a 00 01 0a 2a -->@.@-6.*...*
0020 00 cb 99 f0 13 88 d5 01 dd e4 9a 7e 4d dd 80 18 .....-...M...
0030 01 f6 40 e7 00 00 01 01 08 0a c0 d9 ed 93 a1 90 ..@-.....
0040 bc 33 47 45 54 20 2f 20 48 54 54 50 2f 31 2e 31 -3GET / HTTP/1.1

```

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
```

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

- Installer Node-RED Dashboard

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

- Redémarrer pour que l'installation soit prise en compte

```
pi@raspberrypi:~ $ 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

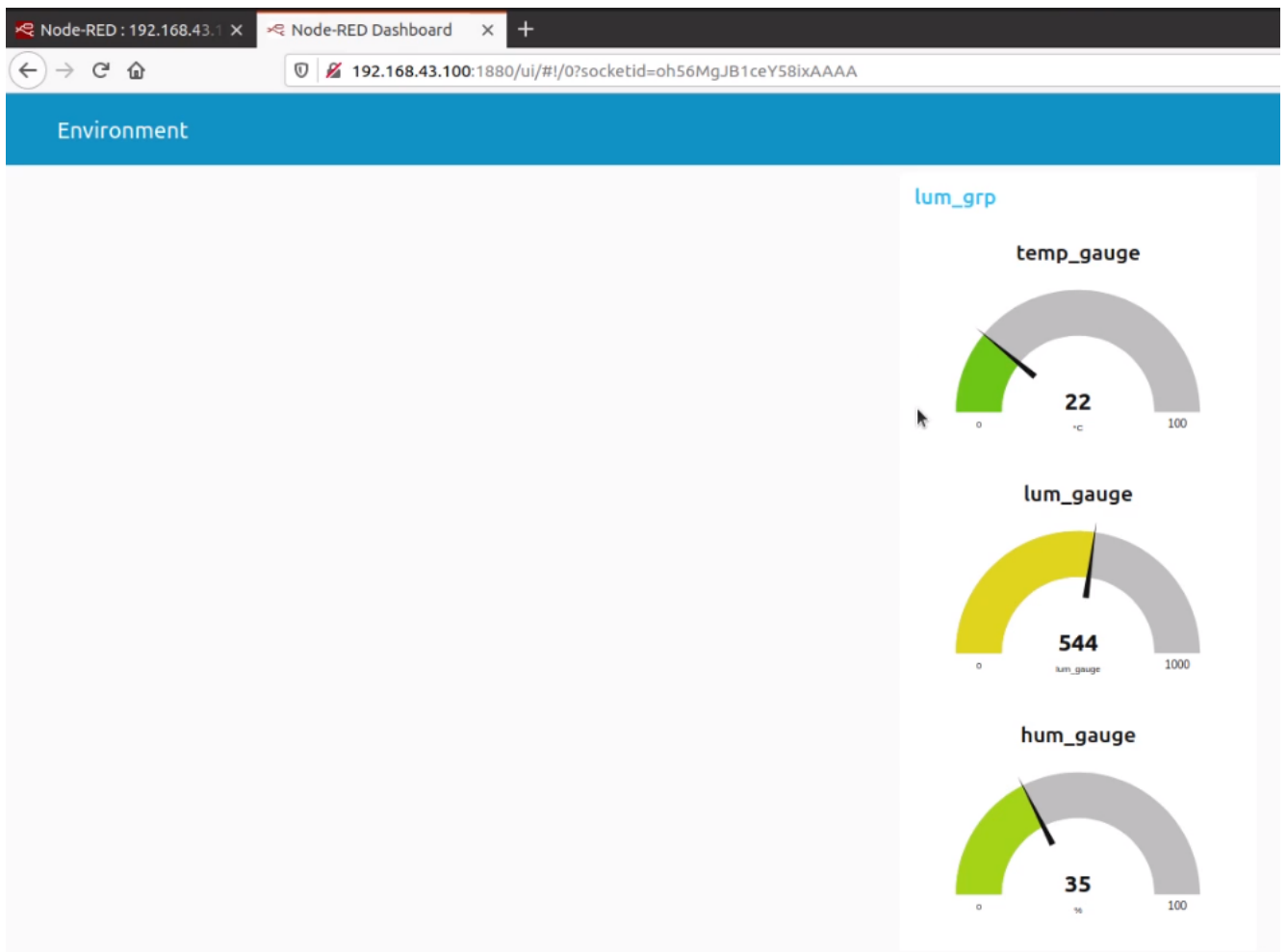
SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set a new password.

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