Abdellah Ismaili Alaoui

Rapport « Nouveau usage de l'internet global »

But des TPs :

- TP1: Créer un serveur Web sur un Arduino ATMega
- TP2: Réaliser un serveur CoAP sur Arduino
- TP3: Développer un proxy HTTP/CoAP sur le raspberry
- TP4: Développer un serveur MQTT sur le Raspberry

Prérequis :

- Arduino 64 bits Linux
- Téléchargement du packet MKR100
- Inclusion des bibliothèques « coap.h », « dht. »
- Téléchargement de la libraire « AdafruitSensors.h », « CoAP-simple-library » et « WiFi101"

TP1: Créer un serveur Web sur un Arduino ATMega

Le but de ce TP est de tester l'Arduino ATMega. Grâge à l'exemple de librairie WebServer où j'ai rajouté l'adresse MAC du shield, on affcihe un message de test « hello world » sur le navigateur web.

TP2: Réaliser un serveur CoAP sur Arduino

Le but de cette partie est de réaliser un serveur CoAP sur Arduino pour qu'il puisse envoyer des données lorsque le client en demande. En fait, CoAP est un protocole basé sur une architecture Request/Response, basé sur le protocole UDP de la couche applicative du modèle OSI.

J'ai téléchargé un exemple de coapserver disponible sur Arduino sur lequel j'ai effectué les modifications nécessaires tel que mettre à jour les ports DHT et lumière et l'adresse MAC. J'ai également rajouté les différentes fonctions callback de température, luminosité et humidité qui se basent sur la bibliothèque DHT. J'ai également créé une adresse IP avec laquelle on pourra par la suite récupérer les valeurs lues sur les capteurs. J'ai rajouté cette dernière sur l'interface convenable avec la commande : sudo ip a add 192.168.1.4/24 dev enp2s0.

Enfin, sur un terminal on récupère les différentes valeurs avec la commande coap-client :

```
info@i005pc21:~$ coap-client coap://192.168.1.4/humidity
v:1 t:CON c:GET i:52ff {} [ ]
 humidity: 40%
info@i005pc21:~$ coap-client coap://192.168.1.4/temperature
v:1 t:CON c:GET i:974a {} [ ]
Temp: 25 °C
info@i005pc21:~$ coap-client coap://192.168.1.4/light
v:1 t:CON c:GET i:a237 {} [ ]
Il fait jour
```

TP3: Développer un proxy HTTP/CoAP sur le Raspberry

Le but de ce TP est de réaliser un proxy CoAP avec un Raspberry. Ce dernier, envoie des requêtes HTTP sur le proxy qui demande à l'Arduino en CoAP les valeurs souhaitées.
Arduino ,quant à lui, renvoie ces valeurs en CoAP. Le proxy récupère ces valeurs grâce au code python qui tourne sur le Rasbperry. La réponse finale est renvoyé en HTTP. On note que la connexion du Raspberry est effectuée par Ethernet.

Tout d'abord, il fallait installer Linux sur la carte SDI afin de l'intégrer sur le Raspberry pour configurer ce dernier. J'ai réalisé ceci depuis mon ordinateur personnel vu que les ordinateurs de la salle i005 ne disposent pas de lecteur SD. Ensuite, grâce à une interface, j'ai configuré le Raspberry. Sur un terminal, on lance la commande raspi-config

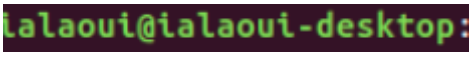
→ edit connexion → device eno → IPv4 → manual → add : 192.168.70.2/30 (adresse IP passerelle du Raspberry).

→ if-config → IPv6 : ignore → save.

Pour passer en ssh il fallait, lancer la commande : ssh <user_name>@address_IP
Dans ce cas : ssh ialaoui@192 .168.70.1

Afin de récupérer les valeurs des capteurs, j'ai eu recours à un code python. Ce dernier initialise l'adresse locale de l'Arduino "192.168.1.245". Il dispose d'une fonction get_values qui se charge de récupérer les valeurs des capteurs. La fonction main, quant à elle, renvoie les différentes valeurs sur l'adresse du Raspberry "192.168.70.2".

Ensuite, j'ai configuré la connexion WiFi sur le Rasbperry, pour cela il fallait modifier les deux fichiers en ssh :



➔ */etc/network/interfaces/*
auto wlan0 (nom de l'interface wifi que j'ai récupéré avec la commande ifconfig)
iface wlan0 inet dhcp
wpa-conf /etc/wpa_supplicant/wpa_supplicant_conf

➔ */etc/wpa_supplicant/wpa_supplicant_conf*
country=FR
network={
ssid= « iPhone de Abdellah » (nom de la connexion wifi partagée par mon tél)
psk= « 12345678 » (le mot de passe de cette dernière)
}

Afin de tester, j'ai redémarrer le Rasbperry et fait un ping vers google (8.8.8.8), les packets étaient bien reçus.

Afin de faire tourner correctement ce code python, il m'a fallu installer Pip, Flask et Coapthon :

- pip : → curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
  → python get-pip.py
- Flask : pip install -U Flask
- Coapthon : pip install -U Coapthon

Enfin, on lance le code python avec la commande : python coaphttp.py

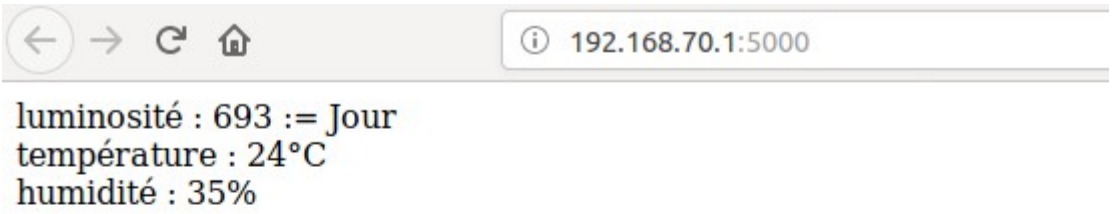
```
 * Support:        https://ubuntu.com/advantage

624 paquets peuvent être mis à jour.
369 mises à jour de sécurité.

New release '18.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Mon Nov 26 14:02:02 2018 from 192.168.70.2
ialaoui@ialaoui-desktop:~$ python coaphttp.py
 * Serving Flask app "coaphttp" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://192.168.70.1:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 270-840-611
192.168.70.2 - - [26/Nov/2018 14:05:50] "GET / HTTP/1.1" 200 -
192.168.70.2 - - [26/Nov/2018 14:05:50] "GET /favicon.ico HTTP/1.1" 404 -
```

Dans un navigateur Web, on va sur l'adresse http://192.168.70.1:5000 , on obtient le résultat suivant :



En parallèle, sur le moniteur d'Arduino, on visualise la connexion au wifi ainsi que la récupération des différentes valeurs :

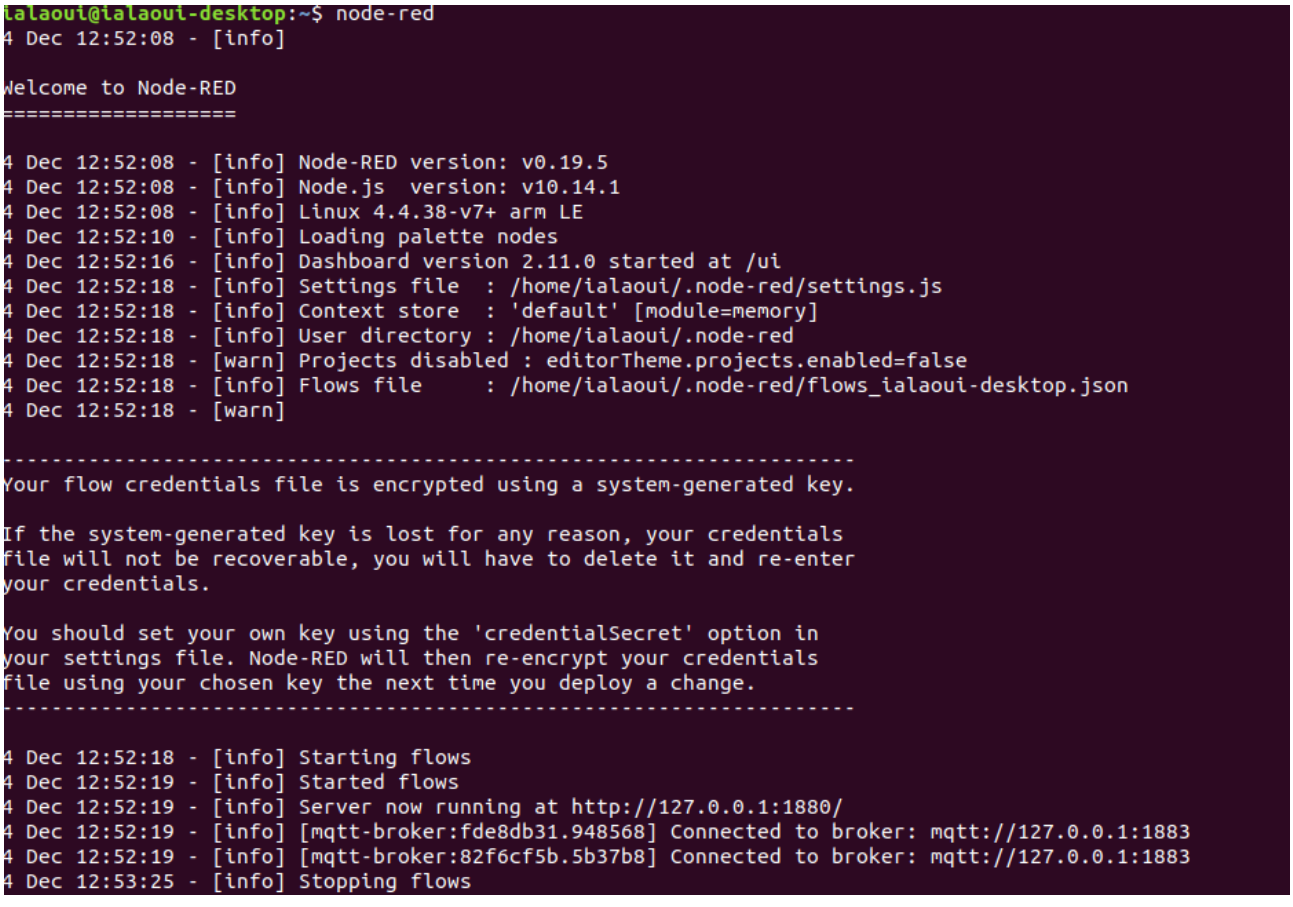
TP4: Développer un serveur MQTT sur le Raspberry

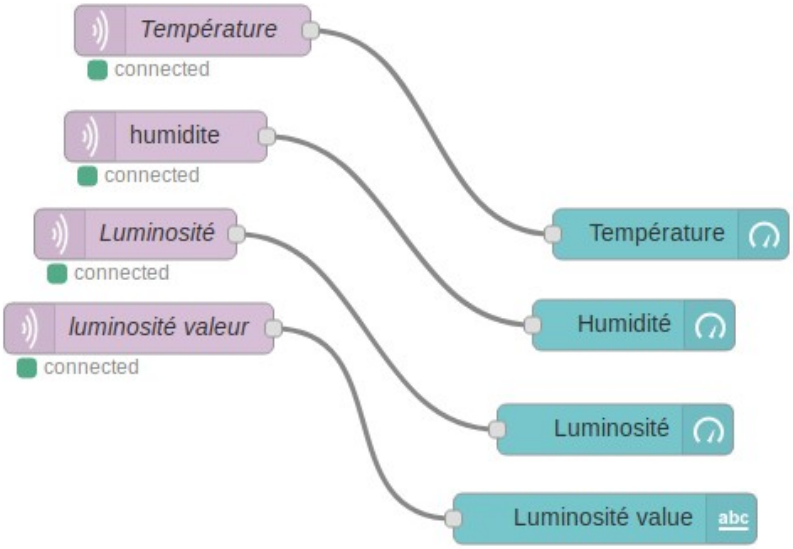
Contrairement à HTTP ou CoAP (requête/réposne) utilise une architetcure publisher/subscriber. Ici, le publisher est l'arduino (client), le subsriber est l'ordinateur (client) et le broker (serveur) est le Raspberry qui se charge de gérer les topics, relayer ses derniers auprès des souscrivants et relayer l'information auprès des clients.

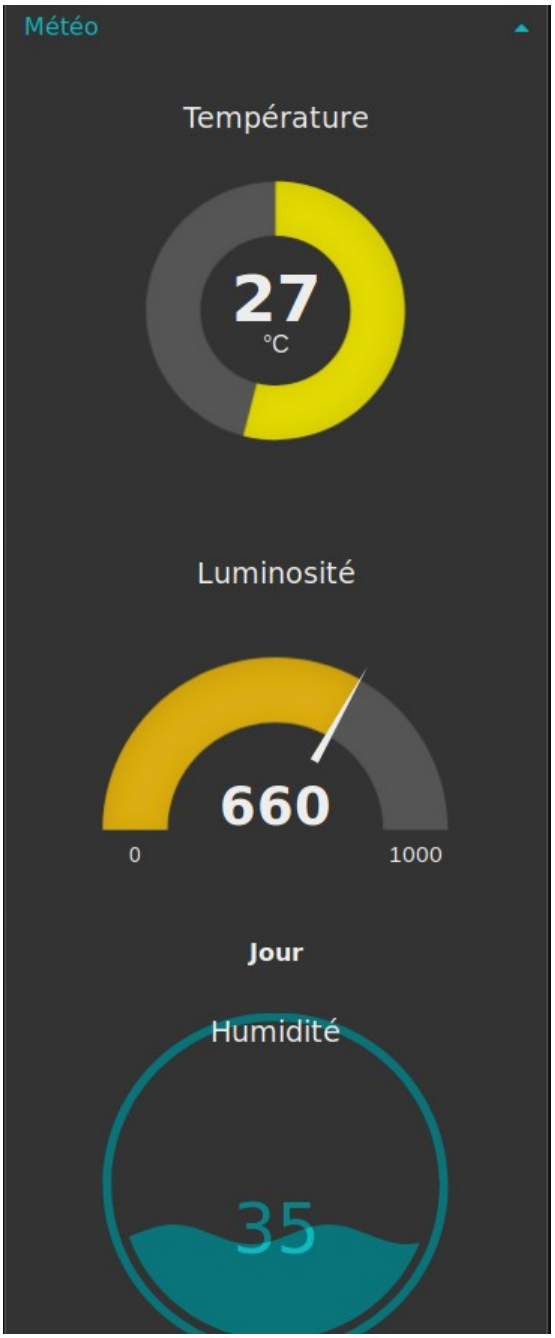
En se basant sur les configurations du TP d'avant, j'ai commencé par télécharger mosquitto sur le Rasbperry avec la commande : sudo apt-get install mosquitto-clients après avoir effectuer d'éventuelles mise-à-jour avec la commande : sudo apt-get update. Pour se rassurer que le téléchargement a bien été mené, on lance la commande : service mosquitto status.

Comme précédemment, à partir d'un exemple mqtt où j'ai rajouté les différentes fonctions callback et mis à jour le mqtt_server qui correspond à l'adresse IP du Raspberry sur mon téléphone : « 172.20.10.9/8».

On compte afficher les résultats sur un dashboard, c'est pour cela que j'ai installeé le node-red-dashboard sur le Raspberry en suivant le tutoriel https://nodered.org/docs/hardware/raspberrypi. Avec la commande node-red on lance l'interface. Sur l'adresse « http://192.168.70.1:1880/ » on accède au node-red-dashboard où j'ai construit le dashboard, que j'ai nommé « météo », sur lequel apparaissent les différentes valeurs.




Température
connected
humidite
connected
Luminosité
connected
luminosité valeur
connected
Température
Humidité
Luminosité
Luminosité value
abc


Météo
Température
27
°C
Luminosité
660
0
1000
Jour
Humidité
35