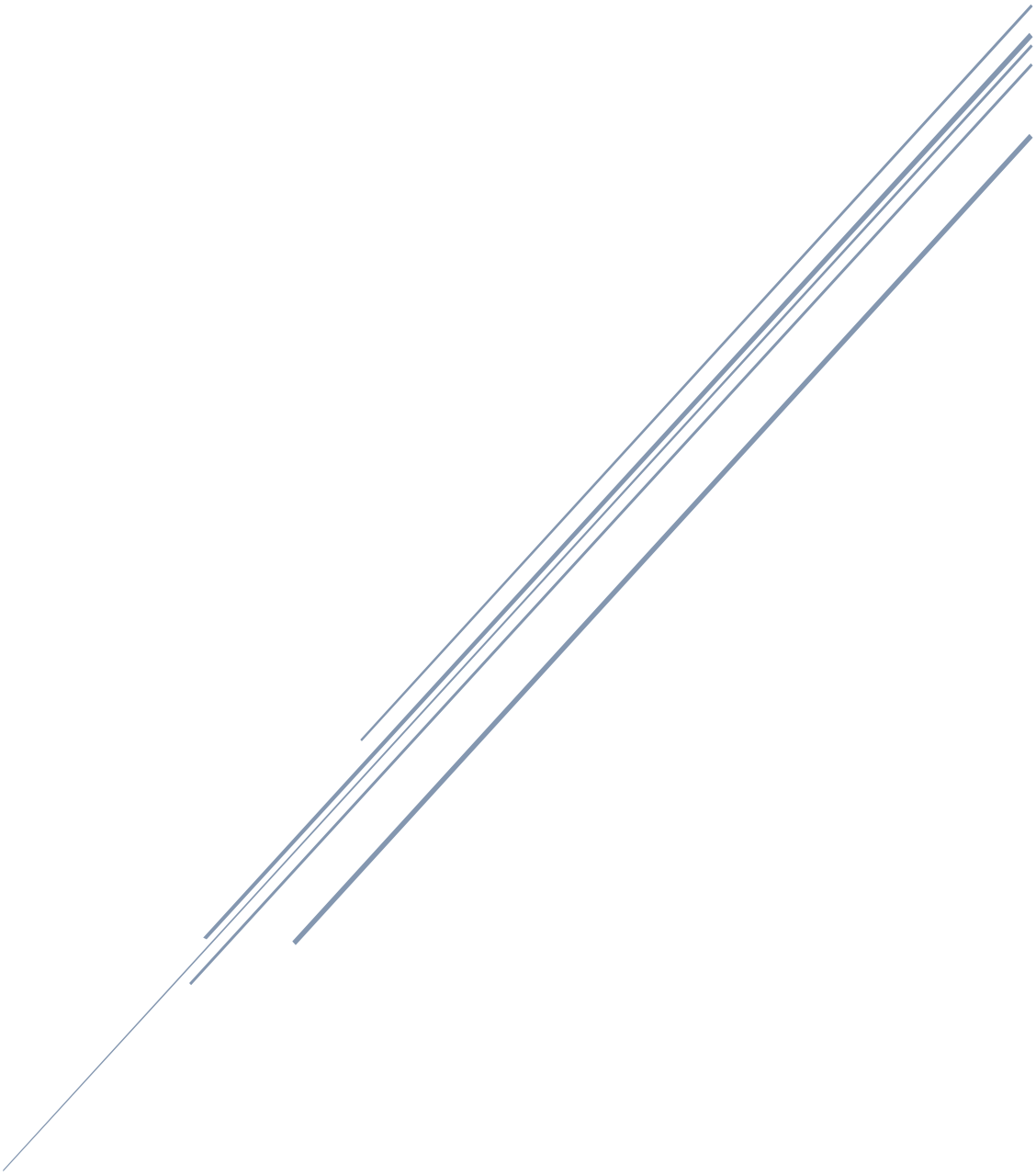
SYSTEME DE RADIATEUR CONNECTÉ

TP IOT 18/03/2019



UNIVERSITE DE NICE SOPHIA, MIAGE
Master 1 – Mike AUBENAS – SOFIAN CHAIBI

1 Installation

Afin de pouvoir utiliser notre projet il vous sera nécessaire d'installer quelques outils et de modifier quelques éléments.

1.1 Node JS

Pour utiliser la partie server de notre application il est nécessaire pour vous d'installer node js. Pour ce faire rendez – vous sur le site dessous et effectuer le tutoriel (dans la langue de Shakespeare) jusqu'à l'étape 8 (pour windows).

https://www.guru99.com/download-install-node-js.html

Pour vérifier la bonne installation lancez une invite de commande (cmd) et taper « node -v »

```
E:\>node -v
v10.13.0
```

Il est possible que le système nécessite un redémarrage.

Pour les autres systèmes d'exploitation suivait le tutoriel suivant.

https://www.ostechnix.com/install-node-js-linux/

1.2 L'archive

Après avoir télécharger l'archive et dézippez la, placé vous a la racine du projet dans « lieuDeDezippe/serveur ».

Une fois fait il va vous falloir modifier quelque variable pour pouvoir certaines fonctions dépendent de votre Ip ou de votre wifi chose qu'il est préférable de garder pour vous !

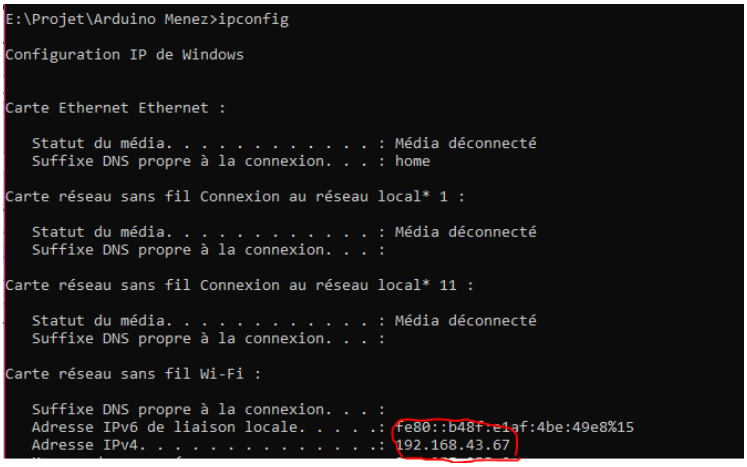
1.3 Modification « serveur.js »

```
var mqtt = require('mqtt')
var client  = mqtt.connect('mqtt://10.154.125.62') // mettre votre adresse IP  "mqtt://xxx.xxx.xxx.xxx"
```

Ici il vous suffit de placer le vôtre IP comme indiqué.

Pour avoir votre IP :

- Ouvrez un terminal
- Taper « ipconfig »



1.4 Modification « mqtt_full.uo »

Pour ce fichier il nécessaire aussi de modifier les adresse IP. En effet pour le celui en ligne était trop instable il a donc été nécessaire de passer sur un serveur perso.

Il faut donc modifier la ligne suivante :

```
/*===== MQTT broker/server and TOPICS ========*/
const char* mqtt_server = "192.168.1.11"; /* ip perso car broker en ligne trop instable */
// address a changer au dessus
```

Ainsi que vos identifiant de connexion internet.

```
void connect_wifi() {
  //const char* ssid = "Téléphone Mi";
  //const char *password= "966ca0b0556b";
  const char* ssid ="Livebox-CBB4";
  const char* password="19445155EADA3345D4D33267DC";
```

1.5 Modification « WebControl2.uo »

Ici de même que pour le service mqtt il est nécessaire de modifier le fichier.

Tout d'abord votre wifi :

```
/*===== Variable Wifi  ========*/
const char* ssid ="Livebox-CBB4";
const char* pass="19445155EADA3345D4D33267DC"; // ne pirater pas ma box svp :p
```

Ensuite votre IP :

```
String ipServeur ="192.168.1.11"; // adress a changer ici
unsigned long previousMillis = 0;
```

Et finalement votre IP (encore) :

```
IPAddress server(192,168,1,11); // address du serveur
```

Attention : il ne s'agit pas d'une chaine mais de chiffres séparés par des virgule

1.6 Lancement du serveur

Une fois tout modifié il est grand temps de lancer le serveur toujours placé à la racine de serveur.

Lancé un cmd à cet emplacement, puis faite « node serveur.js ».

```
E:\Projet\Arduino Menez\serveur>node serveur.js
Serveur crée
```

Une fois cela effectué allez sur votre navigateur préféré.

Taper dans le navigateur cela (les x étant votre adresseIp) « xxx.xxx.xxx .xxx :8080/webSocket.html »

10.154.125.62:8080/webSocket.html

Une fois cela effectuer vous pouvez transférer sur vos/votre esp32 les différents programmes.

2 Choix Technique

Pour ce projet, nous avons décidé de compartiment la solution en deux parties bien distinctes.

Une partie de communication qui passe par le MQTT et une partie qui passe l'utilisation de socket.

Tout d'abord, pour pouvoir effectuer un lien entre notre esp et une page web de contrôle, il a fallu trouver une solution pour permettre de le faire communiquer sur le même réseau.

Pour ce faire, nous avons choisi le Node js car il s'agit d'une solution rapide pour la mise en place. De plus, la simplicité d'utilisation nous a permis de concentrer plus la partie communication des esp 32.

Le serveur Node est donc effectué en Node js et contient un mécanisme de WebSocket afin de communiquer avec la page et un des deux esp32. Le mécanisme est très simple lors du lancement du serveur celui-ci se met en attente de requête cliente, pendant ce temps au lancement de l'esp et/ou de la page chacune émette un socket avec un message permettant de les identifier et de les sauvegarder. Cette sauvegarde des sockets permet au serveur de pouvoir librement, lorsqu'il en besoins de communiquer des informations que ce soit à la page web ou a l'esp.

De plus, notre serveur contient aussi un client mqtt permet de surveiller et publié sur les topiques associés à l'esp 32.

La configuration WebControl2, est en fait basé sur un défi personnel que nous voulions illustrer. Il s'agit de client web, qui fonction par l'utilisation des webSocket (api spécifiée plus bas). En effet, ce script permet communiquer à intervalle régulier des informations recueilli par l'esp (température, luminosité) au serveur via l'envoi de socket. Bien sûr, il s'agit d'une communication réciproque, le serveur peut aussi envoyer des ordres a l'esp comme l'allumage de la lumière (#chauffage).

De plus, en commun avec le programme mqtt, l'esp en webSocket possède aussi une gestion d'erreur (#connection) et de température trop élevée.

En conclusion, ceci était un bon challenge pour nous et nous avons trouvé qu'effectue ce programme était plus simple et moins dense que le programme mqtt. Cependant, il possède ces limites et ne permet pas une utilisation sur batterie de contrôleur qui serait assez complexe a gérer au niveau du serveur.La configuration mqtt_full, est quant à lui basé sur le tp effectué lors du cours IOT, il s'agit en effet de permettre la communication via notre serveur et l'esp. Pour ce faire, nous utilisons un système d'abonnement et de poste sur des topics permettant ainsi pour l'esp32 et le serveur de communiqué entre eux des informations. Nous avons utilisé un broker (point relais, serveur) dans un premier temps un broker en ligne, mais de nombreuses restrictions et panne nous ont obligé de passé sur un broker sur notre propre réseau.

Concernant la page de contrôle web, nous voulions partir sur quelque chose de complétement diffèrent en react natif permettant le développement d'application cependant suite a un accident technique de notre part (perte de l'intégralité du projet, oui, on sait, on n'est pas doué …), nous somme revenue sur un simple page web indiquant les deux salles et permettant l'allumage et l'arrêt du

chauffage (modélisé par les led) avec des graphismes quelque peu « épuré ». La page web ne fait que recevoir les données émises par le serveur et les affiche ou envois des instructions via les webSocket.

WorkFlow des esp32

- MQTT :
    - Connection au wifi
    - Paramétrage des divers pins de réceptions de données
    - Connection au server mqtt
    - Inscription au topic à surveiller
    - Surveillance du topic et action si notification
    - Boucle toutes les 10 secondes :
        - Récupérations de luminosité
            - If lumière basse (=nuit)
                - Deep sleep de 30 secondes pourrait (juste pour le test mais égale économie d'énergie)
        - Récupérations de la température
            - If temp>25 :
                - Température anormale (=feu), message sur topic alerte
        - Envois des différentes informations sur leurs topics respectif
- Web Socket :
    - Connection au wifi
    - Paramétrage des divers pins de réceptions de données
    - Connection au server webSocket
    - Surveillance de socketRecus
        - If socket reçus action en fonction de l'ordre émis
    - Boucle toutes les 5 secondes :
        - Récupérations de luminosité
        - Envois luminosité
        - Récupérations de la température
            - If temp>25 :
                - Température anormale (=feu), message sur topic alerte
        - Envois température

La led rouge situé en pin 18 et indicateur permettant de signaler une erreur de connexion wifi ou serveur sur les deux esp32.
Workflow Serveur

- Mise en place du serveur mqtt
- Inscription au topic nécessitant surveillance
- Mise en place du serveur web
- Attente de socket venant du serveur et de la page web
- Sauvegarde des sockets afin de pouvoir leur répondre à tout instant
- Attente de socket / notification mqtt
    - If socket reçus action en de l'emetteur
        - Web = transmission to esp32
        - Esp = transmission to Web
        - MQTT = transmission to Web

- arduinoWebSockets: https://github.com/Links2004/arduinoWebSockets/

- WebSocket: https://www.npmjs.com/package/websocket

Conclusion

Merci d’avoir lu ce rapport navré pour l’interface web qui n’est pas a la hauteur et reflétant du travail effectuer pour ce projet.