Documentation technique

Projet : IOTProject	**Client :**

Versions du document

Date	Version	Emetteur
26/12/2017	V1R1	Benjamin VIRGO
29/12/2017	V1R2	Benjamin VIRGO
05/01/2018	V1R3	Benjamin VIRGO
08/01/2018	V1R4	Benjamin VIRGO
26/01/2018	V1R5	Benjamin VIRGO

Liste de diffusion

Destinataire	Contact
Nicolas Laurio	nicolas.laurio@southside-interactive.com

Sommaire

1. Introduction

1.1. Objectif du document

La présente documentation technique a pour but de servir de support d'aide à la compréhension du projet dans sa globalité.

1.2. Fonctionnement du document

1.2.1. Acronymes et définitions

Pour des questions de facilité, des acronymes ou abréviations seront utilisés :

- MQTT : (MQ Telemetry Transport) est un protocole de messagerie publish-subscribe basé sur le protocole TCP/IP (*Wikipédia*).
- RPi : Raspberry PI.
- ESP8266 : micro-contrôleur
- TTS : Text-to-Speech, texte vers vocal
- Relais : Commutateur piloter à l'aide d'un éléctro-aimant.

1.3. Présentation du projet

IOTProject est un projet réalisé dans le cadre du cursus EPITECH.
Le choix du projet est libre, seul le thème de l'IOT est à respecter. Il nous est imposé d'utiliser :

- Le protocole MQTT
- Un micro-contrôleur et/ou un RPi
- Node-Red (ou équivalent) pour établir des scénarios

Il nous est paru évident qu'une solution domotique est le sujet parfait pour répondre à ces critères.

Nous proposons donc une solution de gestion de tous les appareils électriques domestiques simple d'utilisation et à moindre coup.

2. Mosquitto

Un broker MQTT est installé sur le RPi afin d'interfacer les différents appareils.
Le broker choisi est Mosquitto, pour l'installer taper la commande suivante dans un terminal du RPi :
sudo apt-get install mosquitto

Pour lancer le broker, exécuter la commande :
mosquitto

Dans un premier terminal taper la commande suivant :
mosquitto_sub –h localhost –t test

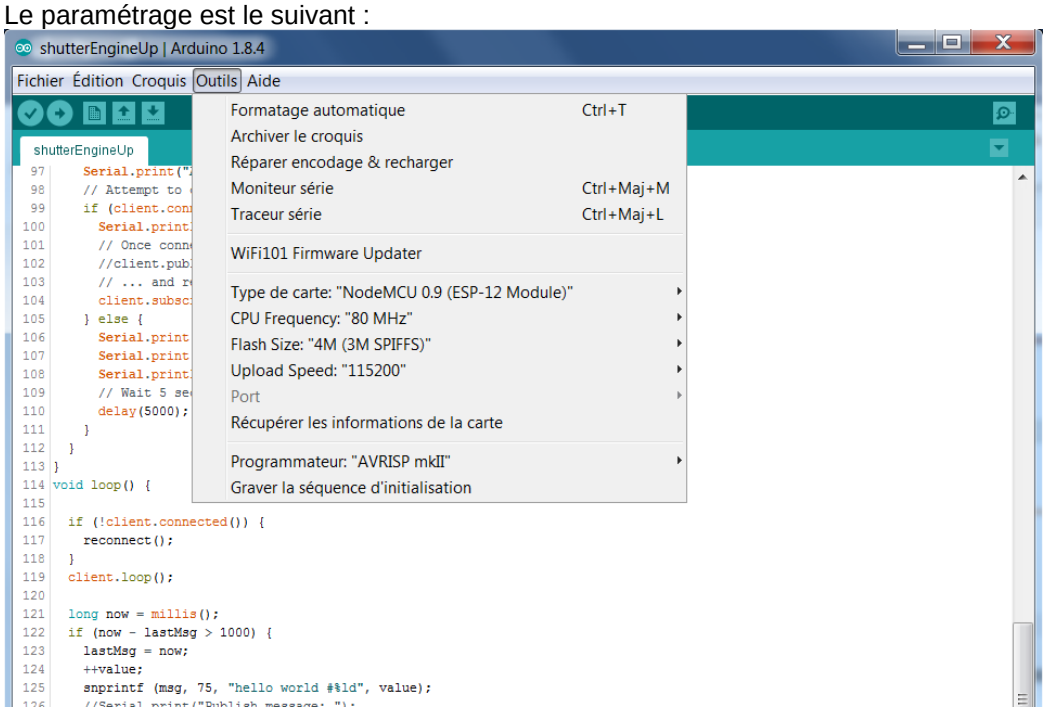
Puis dans un second :
mosquitto_pub –h localhost –t test –m ''hello world''

Si le message « hello world » apparaît dans le premier terminal, le broker est alors actif et fonctionnel.

3. ESP8266

3.1.Sources

Les programmes installés sur les ESP8266 sont développés sous Arduino IDE.
Le paramétrage est le suivant :



Par compatibilité avec les cartes « bas de gammes » commandées, le type de carte renseigné doit être « ESP-12 ».

Pour ouvrir et fermer le volet roulant, deux sources sont nécéssaires : « shutterEngineUp » et « shutterEngineDown ».

Les deux sources sont très similaires, seules quelques lignes diffères.
Ainsi cette partie du document ne traitera qu'un seul des deux sources, prenons en exemple le programme d'ouverture du volet roulant.

shutterEngineUp :

Le code se décompose en trois parties majeures :

- La connexion au réseau Wi-Fi
- La connexion au broker MQTT
- La boucle principale du programme

3.1.1. Connexion au Wi-Fi

void setup_wifi() :

Pour la connexion au réseau Wi-Fi nous utilisons la librairie « ESP8266WiFi.h ».
Les éléments de connexion sont déclarés en tant que variables globales.
Le programme ne se déroulera seulement si l'ESP8266 arrive à se connecter au réseau.

3.1.2. Connexion au broker MQTT

La connexion au broker MQTT se fait par le biais de la librairie : « PubSubClient.h ».

void reconnect() :

Cette fonction n'est appelée que si l'ESP8266 perd la connexion au broker MQTT.
Dans ce cas, le programme ne se déroulera uniquement si la connexion est raccrochée.

Une fois la connexion effectuée, le programme « écoutera » sur le topic « 1floor/shutterEngineUp » grâce à la méthode : **void subscribe(char *topic)**.

void callback(char *topic, byte *payload, unsigned int lenght) :

Une fonction « callback » est appelée à chaque message reçu. Elle prend en paramètre le topic sur lequel le message est reçu, le message reçu et la longueur du message.
C'est dans cette fonction que le message reçu est traité afin de déterminer l'état dans lequel doit se trouver le contact à l'intérieur du relais.
Le message « 2 » signifie que le relais doit être activé et donc le contact fermé : le courant passe.
Tandis que le message « 1 » signifie que le relais doit être désactivé et donc le contact ouvert : le courant ne passe plus.

3.1.3. Boucle principale

void loop() :

Cette fonction est appelée en boucle tant que l'ESP8266 est alimenté en courant.
En fonction de l'état du relais, un message sera publié sur le topic « esp1State » à l'aide de la méthode **void publish(char *topic, const char *payload).**

4. Application

4.1. Application Android

L'application Android est développée en Javascript à l'aide du framework Cordova.
Le code source de l'application se trouve dans le fichier « index2.js », le code HTML dans « index2.html » et les styles CSS dans « style.css ».

Le code Javascript se décompose en quatre parties majeures :

- Gestion de la connexion au broker MQTT
- Pilotage des relais
- TTS et Speech recognition
- Fonction MQTT

Toute la gestion du MQTT se fait par le biais du plugin « cordova-plugin-mqtt ».
Ce plugin adapte la librairie Java « Paho » au Javascript.

La partie TTS est gérée au travers du plugin « cordova-plugin-tts ». Elle permet d'avertir l'utilisateur de tout type de message ou d'erreur sans passer par des pop-ups intempestives pouvant gêner l'expérience utilisateur.
La reconnaissance vocale est embarquée dans le plugin « cordova-plugin-speechrecognition ».