**Lycée Chevalier de Saint Georges**

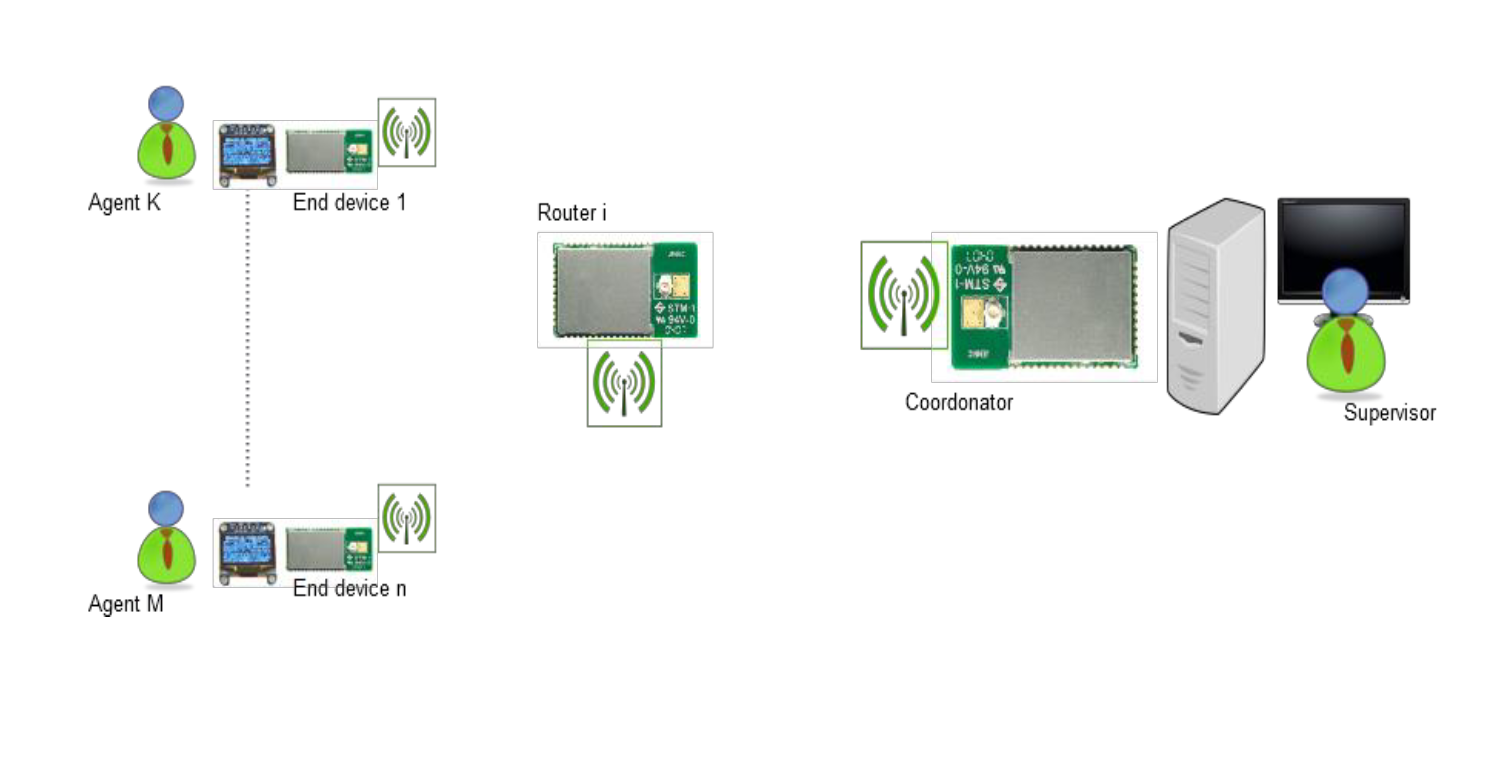
**Rapport de Projet**

**BTS SNIR**

Pager mobile de taches

Bordelais Axell





**Sommaire**

I) Présentation du thème

II) Description des différentes étapes du projet

III) Détail du développement du projet

III-1) Qu’est-ce que l’ESP 8266

III-2) Programmer l'ESP8266 avec l'IDE Arduino

III-3) Configuration de l'IDE Arduino

III-4) Les avantages du Websocket par rapport à l’API REST HTTP classique

# 

III-5) LIBRAIRIE WIFIMANAGER

III-6) Avantages de la librairie WiFiManager

III-7) Installer la librairie WiFiManager dans l’IDE Arduino

# III-8) Comment utiliser la librairie

PRESENTATION DU THEME

**1)INTRODUCTION :**

Dans ce projet du pager mobile de tache, le groupe du projet est constitué de trois personne qui sont Raphael Junior, Christophe Yanis et moi-même Bordelais Axell.

Mes deux professeurs sont monsieur Monteil et monsieur Barreau.

Le système à vocation à être utilisé à l’hôtel Salako du Gosier afin de pouvoir joindre des employés « mobiles » présents sur le site (comme par exemple du personnel d'entretien) et de leurs envoyer de cours message texte. Le système se compose d'une base fixe « le coordonateur », reliée à un ordinateur, où une IHM permettra à un Superviseur de composer le texte à envoyer vers un ou plusieurs agents « End device », en passant éventuellement par un ou plusieurs routeurs. Un « bip sonore » sur un récepteur de poche porté par la personne que l'on cherche à appeler lui indiquera l’arrivée du message de service. La personne aura ainsi les nouvelles consignes qui lui sont assignées.

J’avais pour objectif de réaliser différentes fonctions, c’est-à-dire indiquer couverture ondes radios à l’agent, indiquer charge batterie à agent, capter les ondes radios.

Pour la couverture ondes radios, j’ai décidé d’utiliser un module wifi par conseil des professeur. Ce module wifi est l’ESP 8266.

Qu’est-ce que l’ESP 8266



L'ESP8266 est un module Wi-Fi qui peut être utilisée de deux manières :

**-associé à un autre microcontrôleur** (par exemple un Arduino), l'ESP8266 s'occupera juste de la partie "communication Wi-Fi". L'Arduino enverra des commandes à l'ESP8266 (commandes du style "connecte-toi à tel réseau Wifi", "envoie tel message à tel serveur", etc.). En utilisant l'ESP8266 de cette manière, on est très proche de ce que l'on pouvait faire avec un Shield Wi-Fi

-L'ESP8266 peut aussi être **utilisé de manière totalement autonome**, en exécutant lui-même des applications que vous aurez programmées. Là, ça devient vraiment puissant. L'ESP8266 ne va pas se contenter d'ajouter des fonctionnalités Wi-Fi à votre Arduino, il va exécuter lui-même votre programme Arduino, tout en prenant en charge la partie Wi-Fi. Vous n'uploaderez plus votre programme sur l'Arduino, mais directement sur l'ESP8266. Bon, comme nous le verrons par la suite, il y a quelques limitations quand même, notamment liées au nombre réduit de GPIOs.

Les caractéristiques communes à toutes les versions :

* Tension de fonctionnement: **3.3v**
* Fréquence CPU: 80 MHz
* RAM: 64Ko pour le code, 96Ko pour les données
* Wi-Fi: b/g/n, WEP ou WPA/WPA2
* Antenne: intégrée
* Consommation: entre 60mA et 200mA en fonctionnement normal, quelques dizaines de µA en veille

L'ESP8266 peut s'utiliser de différentes façons. À sa sortie, la puce ne pouvait être utilisée que comme module Wi-Fi "périphérique", associé à autre microcontrôleur. Fin 2014, Espressif (le constructeur à l'origine du module) a mis à disposition de la communauté un SDK, qui a rendu possible l'utilisation de l'ESP8266 comme microcontrolleur totalement autonôme.

Dans ce projet, j’ai décidé de programmé l’ESP8266 à l’aide de l’IDE arduino.

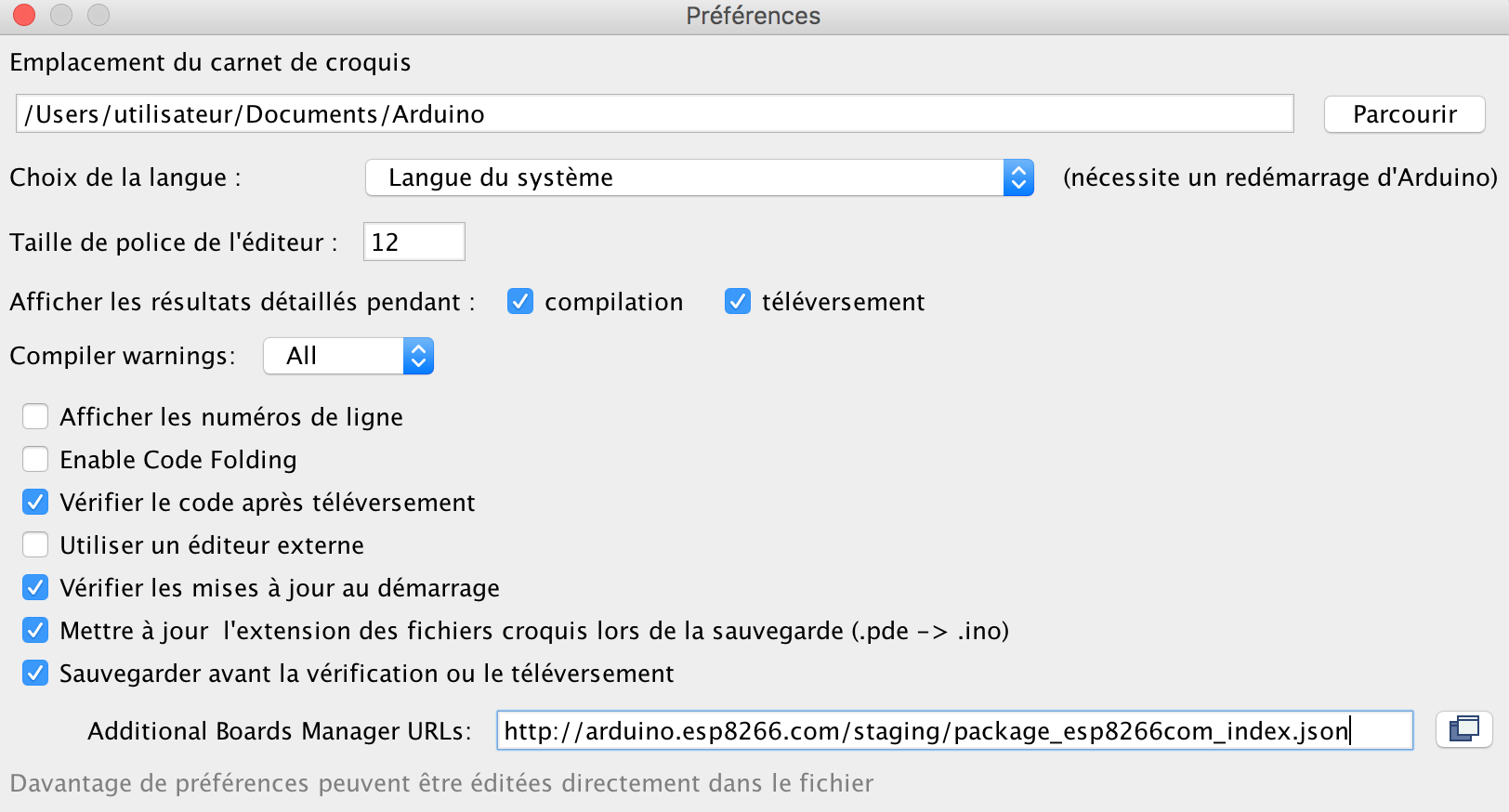
Programmer l'ESP8266 avec l'IDE Arduino

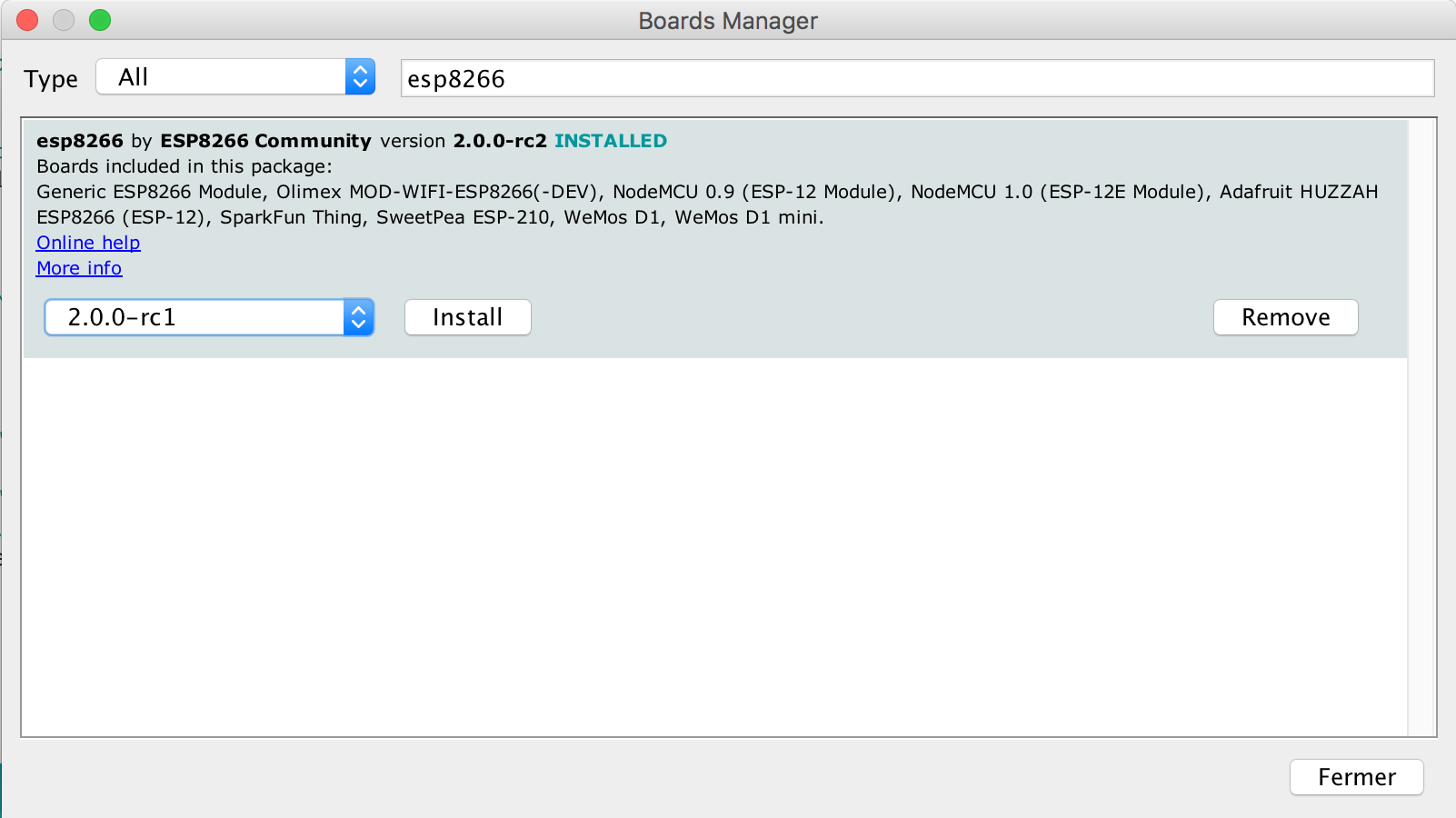
L'objectif est d'utiliser l'ESP8266 comme microcontrolleur principal, et non comme un composant périphérique. Nos capteurs y seront directement connectés et toute la partie logicielle y sera exécutée.

Nous verrons dans un premier temps comment configurer l'IDE Arduino pour qu'il considère l'ESP8266 comme un "type de carte" à part entière

## **Configuration de l'IDE Arduino**

Il faudra télécharger l'IDE Arduino sur le site officiel





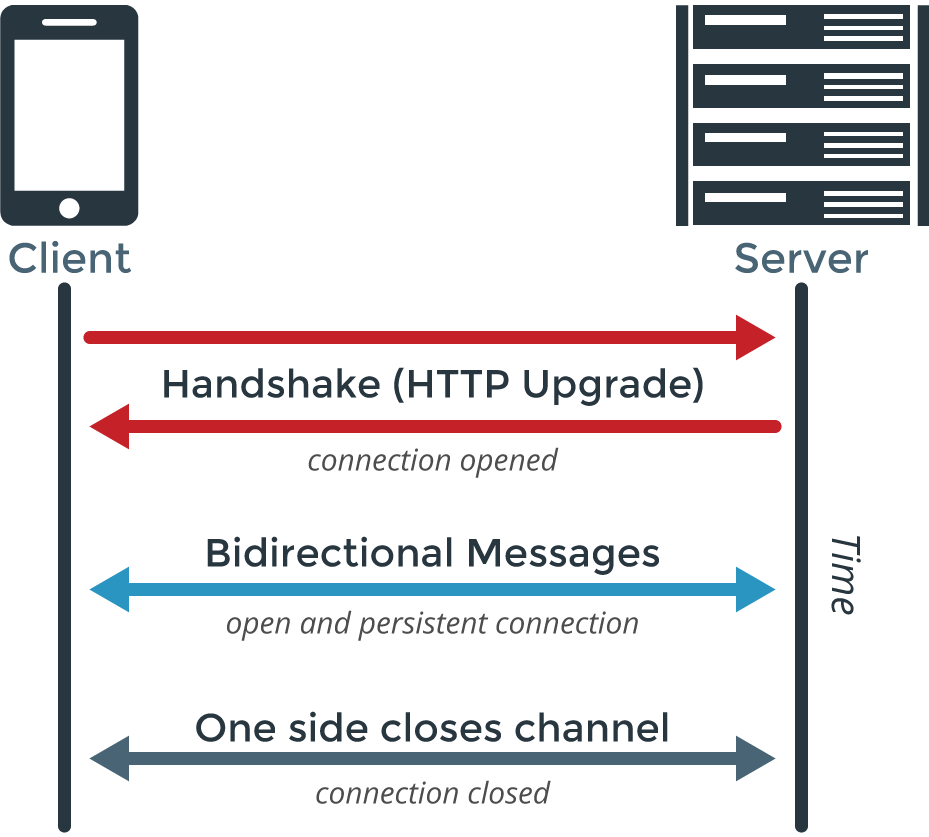
**COMMUNICATION WEBSOCKET AVEC UN ESP8266**

**Le Websocket est un protocole de communication beaucoup plus rapide que le protocole REST qui utilise des requêtes HTTP classiques. Le Websocket permet d’ouvrir un canal de communication bi-directionnel entre deux appareils. Dans le cas présent, ce sera entre un ES8266**

# Les avantages du Websocket par rapport à l’API REST HTTP classique

# Le websocket a été élaboré pour les applications qui nécessitent des réponses rapide ou interactives. Le HTTP a été élaboré à la préhistoire du Web (par le CERN de Genève). Le protocole HTTP est employé pour faire fonctionner les sites internets mais également les applications mobiles (par exemple). Les API REST sont également basées sur l’HTTP. L’HTTP n’est pas adapté aux applications qui nécessitent des réponses rapides ou interactives. En effet, à chaque fois que le client fait une requête au serveur, on doit ouvrir une connexion, attendre la réponse du serveur puis refermer la connexion ce qui est consommateur de ressources et prend du temps de traitement.

Le Websocket vise à résoudre ces problèmes. Le Websocket ouvre un tunnel de communication entre deux appareils. Ce tunnel reste ouvert jusqu’à ce que le client se déconnecte. A n’importe quel moment le client peut envoyer des messages (JSON, binaire, texte…) et vis versa.



* **Bi-directionnel**: le protocole HTTP est unidirectionnel, c’est à dire que le client envoi une requête à laquelle le serveur répond ensuite. Le client consomme ensuite la réponse et ainsi de suite. WebSocket est un protocole bidirectionnel dans lequel il n’y a pas de modèles de message prédéfinis tels que demande / réponse. Le client ou le serveur peut envoyer un message à l’autre partie.
* **Full-duplex**: serveur et client peuvent s’envoyer des messages à n’importe quel moment indépendamment des traitements en cours.
* **Connexion TCP unique**: Généralement, une nouvelle connexion TCP est lancée pour une requête HTTP et se termine après la réception de la réponse. Une nouvelle connexion TCP doit être établie pour une autre requête / réponse HTTP. Avec le WebSocket, le client et le serveur communiquent sur la même connexion TCP jusqu’à ce que le client ou le serveur ferme la connexion.
* **Léger** : le Websocket se concentre sur l’essentiel contrairement à l’HTTP qui embarque de nombreuses informations à chaque question / réponse

Après plusieurs recherches et quelques consultations avec les professeurs, j’ai remarqué qu’il y avait une solution plus adaptée pour pouvoir résoudre ce problème.

Cette solution a été la librairie wifimanager.

**LIBRAIRIE WIFIMANAGER**

**La librairie WiFiManager permet de gérer très simplement la connexion Wi-Fi dans vos projets ESP8266**. La connexion WiFi est très bien prise en charge par de nombreuses librairies, par exemple Blynk, Cayenne, Homie, ESP Easy. Pour des projets plus simples ou qui sortent du cadre de ces librairies, vous devez gérer vous même la connexion au réseau Wi-Fi. La librairie WiFiManager démarre l’ESP8266 en mode point d’accès (mode AP) qui dispose alors d’une fenêtre de configuration à un réseau WiFi. Il est également possible de saisir d’autres paramètres (serveur MQTT, Token Blynk…) en même temps que les paramètres WiFi.

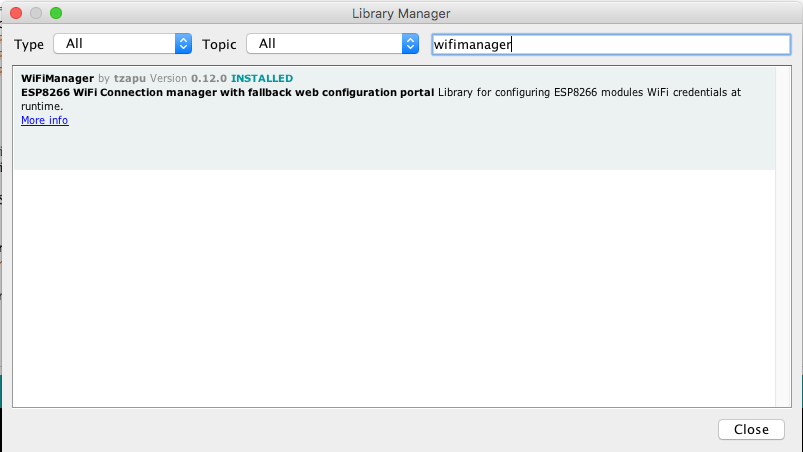
# Avantages de la librairie WiFiManager

Il peut aussi être très intéressant de ne pas embarquer d’identifiant et de mot de passe à un réseau Wi-Fi dans le code Arduino du projet. Par exemple, l’objet devra se connecter sur un réseau dont vous ne connaissez pas les identifiants au moment du développement. Les identifiants peuvent changer. par exemple, dans un établissement scolaire, les identifiants et mot de passe sont changés à chaque rentrée. Le point d’accès a été remplacé et le nouveau mot de passe est différent.

En évitant de coder “en dure” l’identifiant et le mot de passe, vous n’aurez pas besoin de repasser par l’IDE Arduino en cas de changement de mot de passe, de matériel ou de point d’accès.

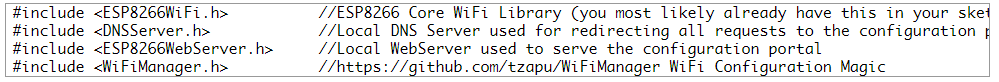
# Installer la librairie WiFiManager dans l’IDE Arduino

La librairie WiFiManager est directement disponible depuis le gestionnaire de bibliothèque de l’IDE Arduino.

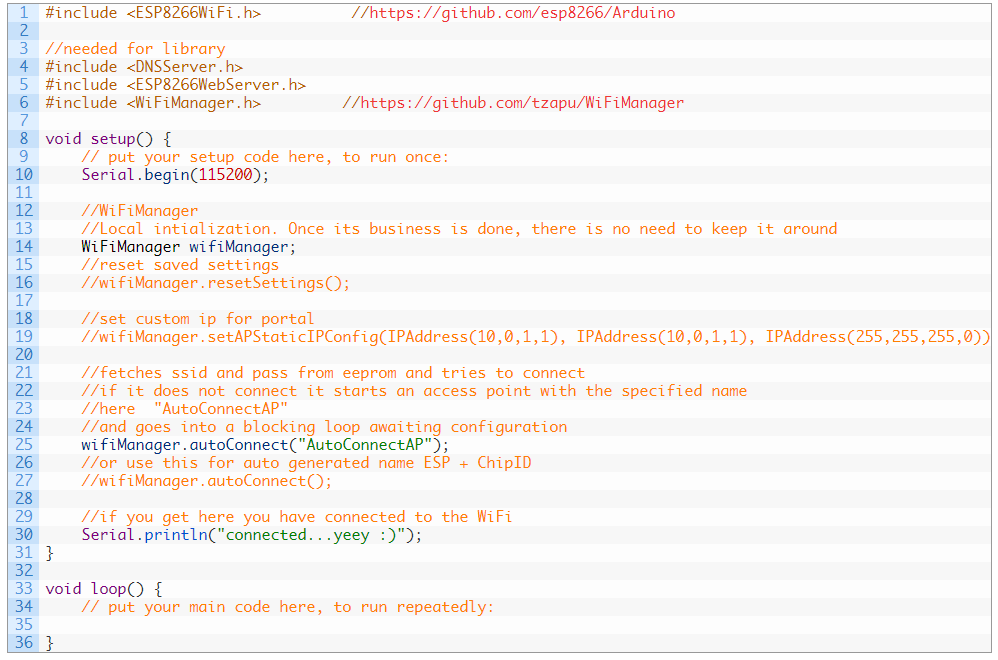


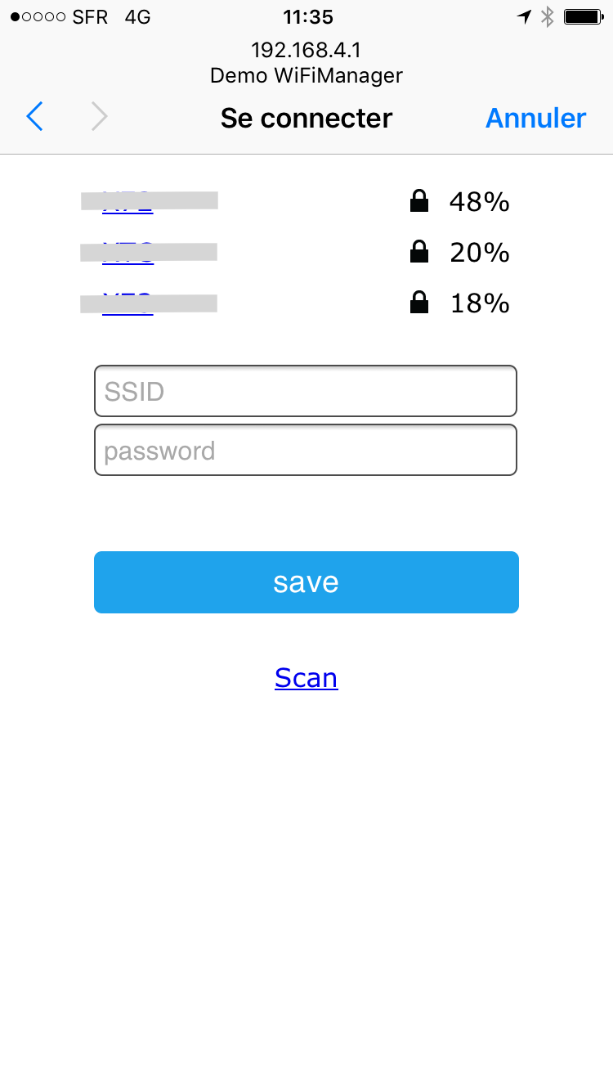
# Comment utiliser la librairie

La librairie WiFiManager s’exécute au démarrage de l’ESP8266. Lorsque l’ESP démarre, la librairie tente de se connecter au dernier réseau connu ou au réseau indiqué dans le code



Donc, si aucun réseau WiFi n’a été configuré ou si le réseau WiFi est inaccessible, un point d’accès est créé. Voici un programme de base que vous pouvez utiliser pour tester la librairie.





MQQT : (Message Queuing Telemetry Transport) est un protocole de messagerie [publish-subscribe](https://fr.wikipedia.org/wiki/Publish-subscribe" \o "Publish-subscribe) basé sur le protocole [TCP/IP](https://fr.wikipedia.org/wiki/TCP/IP).

De très nombreuses bibliothèques sont disponibles pour programmer des clients MQTT, pour la plupart des langages (C, C++, Java, JavaScript, PHP, Python, etc) et sur la plupart des plates-formes (GNU/Linux, Windows, iOS, Android, Arduino…).

Dans le monde réel, de nombreux projets mettent en œuvre MQTT :

* Facebook Messenger : Facebook a utilisé des aspects de MQTT dans Facebook Messenger

**MANUEL D’UTILISATION**

1. Installation de la librairie WifiManager

Cette librairie est disponible sur l’IDE Arduino

1. Configuration de cette librairie dans l’IDE Arduino

# 3.Installation de MQTT : ESP8266-WIFIMANAGER-MQTT

# Cet exemple combine le client standard "Knollery" avec WiFiManager afin de pouvoir facilement

# configurer les paramètres haut débit et MQTT sur l’ESP8266 via une page Web HTML.

Une fois les paramètres enregistrés, l'ESP8266 peut envoyer un message MQTT via votre haut débit à un courtier MQTT protégé par mot de votre choix - soit localement sur votre réseau, soit sur un réseau externe