Rapport du projet
iDoor

Sommaire

Introduction

Mettons-nous dans la peau d'une personne ayant plusieurs logements aux quatre coins de la France. Cette personne fait louer ces logements sur un site qu'on a tous probablement utilisé : AirBnB.
Se déplacer à chaque fois pour donner les clés du logement à tous ses clients, chaque jour serait beaucoup trop fastidieux voire impossible. C'est pour cela que l'idée de créer une clé digitale générée par le propriétaire de ces logements, utilisé par le locataire sur une application que tout le monde peut avoir ou a déjà, Discord. Cela veut dire qu'à partir d'un téléphone, d'une tablette, d'un ordinateur, d'une Smart TV, etc., nous pouvons ouvrir la porte de notre appartement. Cela implique plusieurs problèmes : une clé différente à chaque utilisation, une sécurité, une solution de secours, une clé possédant une date d'expiration correspondant au nombre de jours loués, etc.

I – Les outils/Le Matériel

a) Discord

Discord sera le début du circuit, il nous servira à :

- Générer la clé
- Supprimer la clé
- Vérifier la date d'expiration de la clé
- Créer une fiche utilisateur
- Vérifier la clé afin d'ouvrir la porte

c.f. II- Le Bot Discord

b) The Things Network

The Things Network sera le 2ème élément de notre circuit, il nous sera utile afin d'envoyer le paquet permettant l'ouverture de la porte en utilisant une Gateway lorsque la clé, vérifié par Discord, est valide.

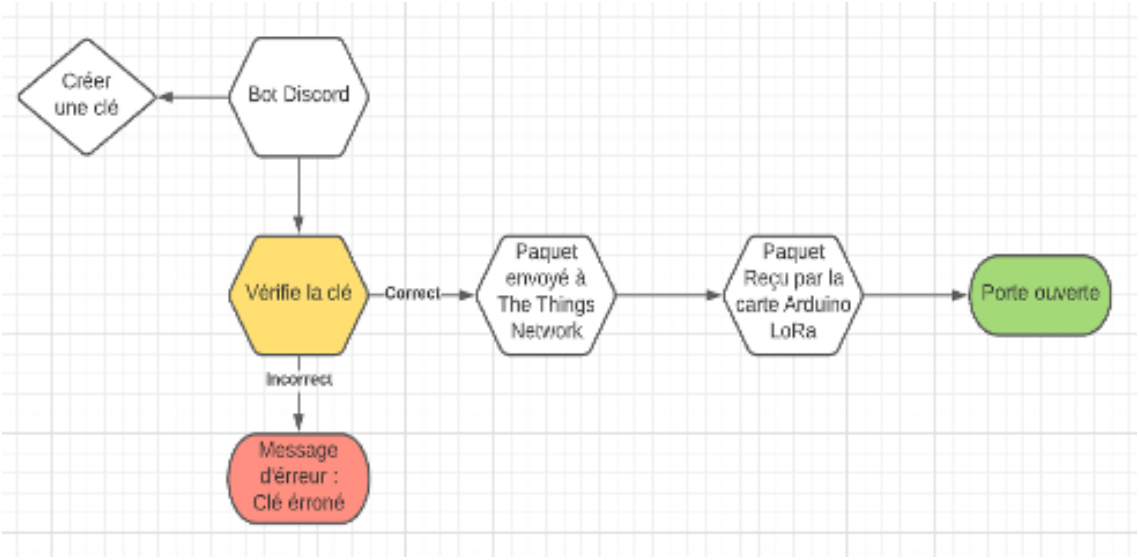
c.f. III-La Carte Arduino LoRa a) The Things Network

c) Carte LoRa

La carte LoRa sera le 3ème et dernier élément de notre circuit, il nous sera utile afin de recevoir le paquet émis par la Gateway The Things Network et permet l'alimentation et l'ouverture de la porte.

c.f. III-La Carte Arduino LoRa

d) Schéma



II – Le bot Discord

Avant de parler d'un bot Discord, le mieux serait de définir ce qu'est Discord. Discord est une alternative web à Skype créée en 2015. Elle permet la VoIP, Messagerie Instantanée, Partage de fichiers, Création de communautés, Création de serveurs, Partage d'écran en direct, Appels vidéo etc.

a) Qu’est-ce qu’un bot Discord et comment fonctionne-t-il ?

Discord est une application propriétaire, c'est-à-dire que le code n’est pas partagé au public mais, malgré tout, Discord on créé Discord.js.

Discord.js est une librairie permettant de communiquer avec Discord avec Node.js, un environnement exécutant du script JavaScript en Back-end, c’est-à-dire, hors d’un navigateur, donc, un serveur.

Tout d’abord, parlons de la manière dont les utilisateurs/profil de Discord sont gérés :

Chaque utilisateur a un ID et un Token automatiquement attribué. Ces deux composantes sont uniques et propre à chaque utilisateur. L’ID est l’identifiant de l’utilisateur qui ne peut être changé et qui est donc associé au compte, et le Token, la clé du compte de l’utilisateur qui peut être changé en modifiant le mot de passe.

Prenons l’exemple d’un utilisateur, j’ai nommé ElecNum#2021. Il a un ID qui lui est attribué. Admettons qu’il est dans mon serveur et qu’il n’est pas très gentil, j’ai donc décidé de le bannir du serveur. Lorsque je vais appuyer sur le bouton Bannir, Discord ne va pas comprendre « Bannir ElecNum#2021 » mais plutôt « Bannir ID » et c’est normal car, imaginons que cette personne aurait changé son pseudo, cela veut dire qu’il serai de nouveau capable de rejoindre mon serveur ? Non, vu que c’est pas le pseudo qui est pris en compte mais l’ID qui est inchangeable.

Maintenant, parlons du Token de ce ElecNum#2021. Vous avez surement tous déjà eu, lorsque vous voulez vous connecter sur un site quelconque, la case « Maintenir la connexion sur cet ordinateur » qui permet d'être toujours connecté même après fermeture du navigateur. Cela est dû au Token, une clé longue et complexe générée est sauvegardée sur votre navigateur (généralement, dans les cookies ou le LocalStorage) qui permet tout simplement de se connecter sans avoir à renseigner l'adresse mail et le mot de passe. C'est comme cela que Discord fonctionne également, quand vous vous connectez que ça soit sur un ordinateur, un téléphone, ou tout autre appareil, votre session reste active même après fermeture de l'application voire extinction de l'appareil.

Ce Token permet donc de se connecter au compte sans avoir à renseigner l'adresse email et le mot de passe.

Le fonctionnement d'un Bot est pareil qu'un utilisateur. Il a un pseudo, un ID et un Token. Nous utilisons son Token afin de se connecter au Bot et donc, de lui dire les instructions à suivre.

Pour créer un bot sur discord, il faut déjà créer une Application sur https://discord.com/developers , ensuite, via cette application, nous pouvons créer le Bot. Une fois le Bot crée et le pseudo choisi, nous avons le Token, cette clé qui nous permet donc de se connecter au Bot.

b) Comment est développé notre bot Discord ?

Nous avons choisi de développer notre bot en Node.JS.

Détaillons les étapes afin d'avoir un environnement convenable au développement du bot :

1- Créer un dossier nommé iDoor BotJS
2- Aller dans ce dossier et ouvrir un Terminal Shell
3- Exécuter la commande « npm install discord.js » (vu précédemment)
4- Exécuter la commande « npm install child-process » (nous allons voir pourquoi plus tard)
5- Installer un éditeur de code, nous avons choisi Microsoft Visual Studio

On y est, nous sommes maintenant prêts pour développer le bot.

Le code est retrouvable sur le lien Github suivant :

Pour démarrer le bot, faire « node index.js » dans le Terminal Shell.

Le code serait beaucoup trop long à expliquer en détail donc, nous allons expliquer le principe.

Nous avons un fichier initialiseur dit « index.js ». Il sert à démarrer le bot ainsi que charger tous les fichiers du répertoire. Dans le fichier index.js, nous l'avons indiqué que toutes les commandes seront dans le dossier « commandes », ainsi, chaque commande est exportée afin d'être exécutée par le bot. Nous avons également créé un fichier « config.json » où nous avons renseigné le Token qui permet de se connecter au Bot ainsi que le préfix à mettre avant chaque message afin d'appeler une commande. Le préfix choisi est « ! ».

Chaque fichier dans le dossier « commandes » correspond à une commande. Nous avons donc 6 commandes :

- Key
- Deletekey
- Sendkey
- Open
- Status
- Help

Nous allons détailler chaque commande.

Key

La commande key permet de créer une fiche utilisateur avec la clé. Elle se compose de la manière suivante :

!key {durée} {utilisateur}

La durée se présente sous la forme : chiffre+lettre. Les lettres possibles sont « d » (pour days), « w » (pour weeks) et « m » (pour months). Si nous voulons générer une clé pour 9 semaines, ça serait donc 9w.

La commande génère une clé aléatoire de 64 caractères pouvant contenir des lettres majuscules, des lettres minuscules et des chiffres. Elle crée un fichier utilisateur nommé par l'id de l'utilisateur renseigné sous la forme « id.json » avec comme contenu :

Clé (généré), Temps (renseigné), Date (de création), Propriétaire (renseigné)

Afin de pouvoir faire des calculs avec la date de création (comme rajouter le temps renseigné à cette dernière), nous allons utilisé une fonction disponible dans JavaScript nommé Date.now() qui nous retourne le nombre de millisecondes

écoulées depuis le 1er Janvier 1970 (correspondant au temps 0 en informatique).

Ainsi, si nous exécutons la commande : « !key 9w @Gearz#2021 » le 11/05/2021 à 01:45:30, cela nous créer le fichier 341297132243910676.json avec comme contenu :

{"cle":"eHrs0JTxVYuzCZsYHW6sBBRAHmzq4NHU4maQZFTb235D6fd32V3zUVVAZZszNvW8","temps":"9w","date":1620690330684,"proprietaire":"341297132243910676"}

Deletekey

La commande deletekey permet de supprimer la fiche utilisateur et donc, la clé. Elle se compose de la manière suivante :

!deletekey {utilisateur}

Il n'y a pas une fonction permettant de supprimer un fichier en JavaScript sans ajouté de module, c'est pour cela que nous avons utilisé le module « child-process » nous permettant d'exécuter des commandes dans le terminal directement via le code JavaScript.

Ainsi, afin de supprimer le fichier de l'utilisateur renseigné, nous vérifions d'abord si une fiche utilisateur est créée, si oui, nous la supprimons avec la commande dans le terminal « rm -f id.json ».

rm est la commande permettant de supprimer un fichier (remove)
-f force sa suppression même si le fichier n'est pas vide
id.json correspond à la fiche utilisateur créée avec !key

Sendkey

La commande sendkey permet d'envoyer à nouveau la même clé en message privé. Elle se compose de la manière suivante:

!sendkey

Open

La commande clé permet d'ouvrir la porte en renseignant la bonne clé. Elle se compose de la manière suivante :

!open {clé}

Premièrement, la commande vérifie si la date de validité de la clé est toujours bonne, pour cela, nous convertissons la durée renseignée dans la commande !key en milliseconde :

- 1d = 86400000ms
- 1w = 86400000ms * 7
- 1m = 86400000ms * 30

Ensuite, nous l'ajoutons à la date de création de la clé et nous vérifions si le résultat est supérieur à la date de l'exécution de la commande !open. Si oui, alors, la clé est expirée et ainsi, le fichier est supprimé et la porte reste fermée. Sinon, elle vérifie si la clé est juste. Si oui, un paquet de 10 bytes est envoyé à The Things Network. Sinon, un message d'erreur est envoyé signalant une clé erronée.

Status

La commande status permet d'obtenir les informations de sa fiche utilisateur. Elle se compose de la manière suivante :

!status

Help

La commande help permet d'obtenir la liste de toute les commandes. Elle se compose de la manière suivante :

!help

Les commandes en rouges sont exécutables uniquement par le propriétaire des biens et non les autres utilisateurs.

c- Sécurité

Notre projet peut contenir 3 failles :

- Sois, le token est trouvé et donc, la personne peut se connecter au bot.
- Sois, le compte discord de l'utilisateur, qui est l'unique personne à pouvoir utiliser la clé, a été piraté
- Sois, la machine/le serveur où fonctionne le bot est piraté.

Dans le premier cas, un token est une clé très complexe pouvant dépasser les 100 caractères et possédant des caractères spéciaux. Les possibilités de combinaisons se comptent par milliards et cela prendrait plusieurs dizaines d'années avec une seule machine afin de trouver le token par la manière Bruteforce (en testant toutes les combinaisons possibles). Ensuite, la personne ne peut récupérer aucune information et ne peut ouvrir la porte car elle n'aura pas accès aux codes du bot du fait que le code n'est pas hébergé dans le bot mais dans la machine/le serveur. Il pourra juste utiliser le bot comme si c'était le sien en utilisant son propre code.

Dans le deuxième cas, il sera demandé aux utilisateurs de vérifier leur mail et d'activer la double authentification qui reste,

pour l’instant, l’une des méthodes les plus sécurisées afin d’authentifier une personne. Il sera stipulé que toutes personnes sont tenues responsables de la protection de son compte. S’il y a un problème, le propriétaire des biens peut supprimer la clé à tout moment et donc, retirer l’accès au compte piraté.

Dans le troisième cas, la personne aura accès à toutes les données, mais il est quasi impossible que cela se produise car avant d’essayer d’accéder à la machine, il faut trouver la machine et il n’y a aucun moyen de le savoir. C’est comme si quelqu’un a les outils pour crocheter la serrure de votre appartement sans connaître votre adresse. Ensuite, généralement, pour héberger un serveur de façon permanente (24h/24 7j/7), nous utilisons un hébergeur et dans ce cas-là, ce sont eux qui garantissent la sécurité des machines (comme OVH.com par exemple).

III – La carte Arduino LoRa

a) The Things Network

1/ Comment ça marche ?
The Things Network est un réseau LoRaWAN communautaire et open source pour l'IoT (Internet of Things).
Ici, il va servir de passerelle entre Discord et la Gateway.

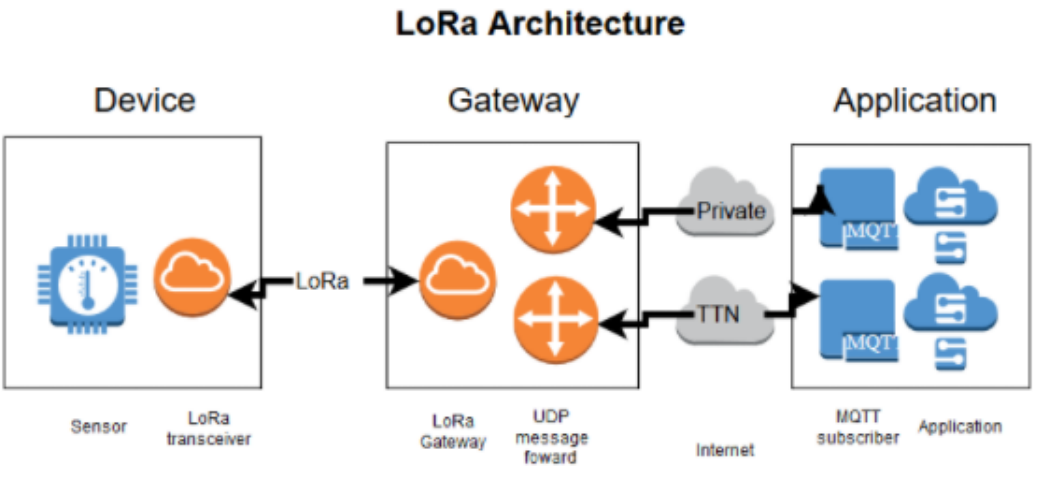
2/ Ca place dans le projet.
Ici, il va servir de passerelle entre Discord et la Carte arduino.

Lorsque le bot Discord va valider l'ouverture de la serrure, il va commander à The Things Network l'envoi de paquet vers la Gateway.
Cela peut être 2 ou 10 bits peu importe quand la validité du code est directement vérifier par le bot Discord, l'important c'est l'envoi d'un signal permettant l'ouverture.
Il suffit d'indiquer dans le code arduino l'id du dossier The Things Network ou le signal va être envoyé.
"Csf" puis "rek'sai"

b) Qu'est-ce que LoRa Arduino et comment fonctionne-t-il ?

Il s'agit d'une carte de communication LoraWan .
La technologie Lorawan est une technologie sans fils qui permet de transmettre à bas débit des informations sur une distance importante avec une puissance faible.

Voici un schéma de son fonctionnement :



source: things4u.github.io

Dans notre projet de serrure connectée on va utiliser cette outil un peu différemment, ici nul besoin de capteur et d'envoyer un

Signale de la carte au serveur d’application.
En effet on a simplement besoin que la carte Arduino reçoive
Un signal afin de permettre l’ouverture de la serrure en mettant en route le servomoteur.

La programmation de la carte se fait en C.

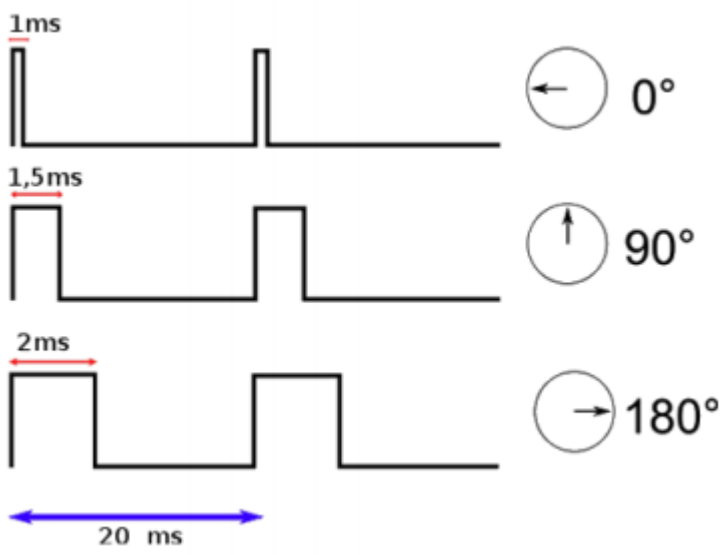
Voici la partie du code qui va permettre au moteur de démarrer après la réception d’un signal (ici 10 bits)

```
if (LMIC.dataLen) {
  Serial.print(F("Received "));
  Serial.print(LMIC.dataLen);
  if (LMIC.dataLen==10) {
      myservo.write(90);
      delay(5000);
      myservo.write(0);
  }
```

On a également l’option d’un bouton poussoir pour l’ouverture de la serrure (imaginons le cas ou on veut simplement sortir d’un bâtiment inutile de repasser par Discord)

Le Servo-moteur :

La carte Arduino va envoyer un signal de type PWM
On va pouvoir venir jouer avec la durée à l'état haut du signal
Afin de modifier l’angle de rotation du moteur.
Sachant que le maximum est de 180.



Pour ce projet l'angle est programmé à 90° la durée à l'état haut est donc de 1.5ms. La fréquence elle sera fixé à 20 ms

c) Branchement du moteur :

2 câbles pour l'alimentation

- Le noir pour la masse (GND)
- Le rouge pour l'alimentation positive (V_bus)

Et un câble pour l'entrée du signal de commande en orange.

Conclusion

Ce projet nous a permis d'acquérir beaucoup de compétences, à la fois en développement avec JavaScript, NodeJS, JSON, Unix et C. Il nous a également permis, dans un système d'éducation qui nous a appris à toujours travailler pour nous, à mettre des connaissances en commun, à travailler pour avancer ensemble. Concernant le projet, il est fonctionnel mais non abouti. Si nous devrions l'utiliser à des fins commerciales, nous devons tout d'abord beaucoup plus se pencher sur l'aspect sécurité du projet. Mais, on peut donc se poser la

question, est ce que tout projet est infaillible ou existe-t’il forcément une faille?