PROJET ESP32 - SMartLock - NET 4104

DAUDE Tristan
GUILLEMET Samuel
SAFON Clement
COSTEL Alexandre

Enseignant responsable : Rémy Grünblatt

Table des matières

1 Introduction

Notre projet consiste à conceptualiser et développer un appareil Bluetooth ESP32 pour ouvrir une porte grâce à une clé sans fil. L'idée serait de créer une paire clé-porte sur le principe d'un système clé/serrure en utilisant des ESP32 fonctionnant en Bluetooth. Au travers d'un protocole réseau utilisant l'adresse mac de la clé et un processus d'authentification basé sur un nonce.

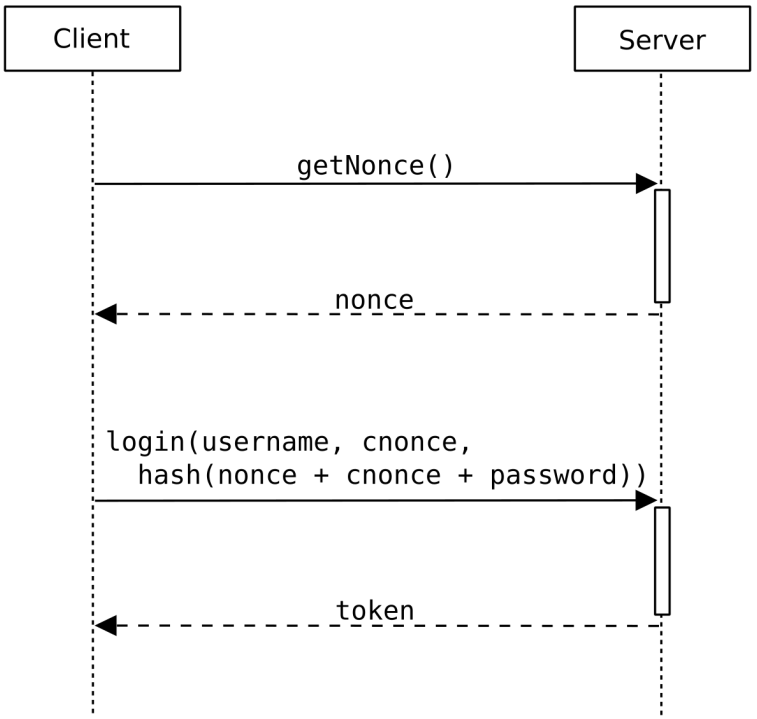
La création de cette application passera par l'utilisation de bibliothèques microPython, le portage du langage Python sur microcontrôleur. Les principaux objectifs seront l'authenticité de la clé et la disponibilité du service.

Ce projet réalisé dans le cadre du module NET 4104 à Telecom SudParis s'inscrit dans la démarche de développer les compétences informatiques essentielles à la carrière d'ingénieur du numérique. De ce fait, dans ce présent document vous trouverez les conceptions et le code de l'application (via un projet github) ainsi qu'un manuel utilisateur.

2 Développement

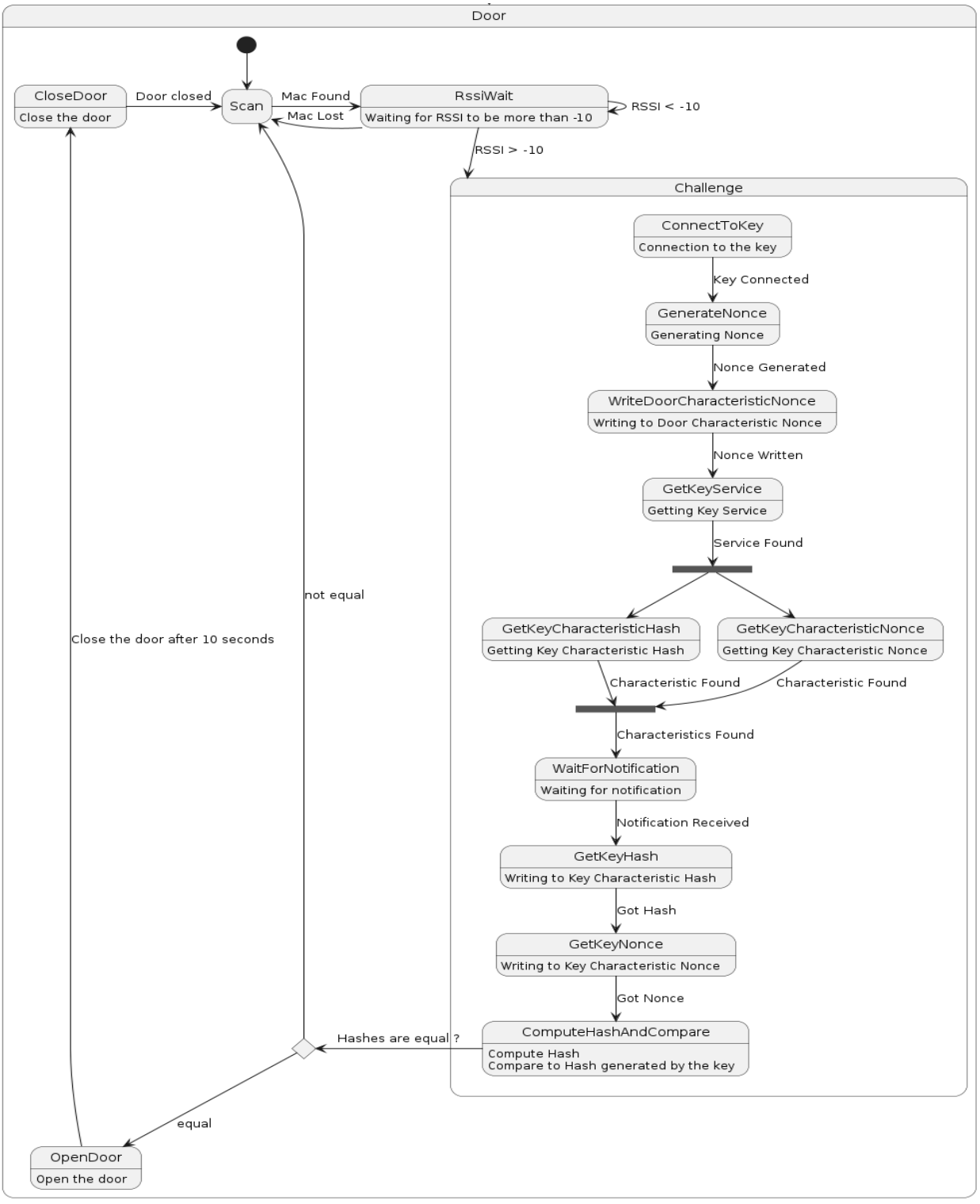
Le projet est décomposé en 2 parties, le code pour la clé et le code pour le microcontrôleur qui actionne l'ouverture ou la fermeture de la porte. La structure du fonctionnement de la porte est présentée plus bas.

Le principe est plutôt simple, vérifier si l'adresse mac de l'objet Bluetooth correspond à une des adresse autorisées, si ou attendre que celle-ci soit suffisamment proche en regardant la puissance de réception. Puis, une fois celle-ci assez proche, établir une connexion et tester la "clé" avec un processus d'identification par nonce.

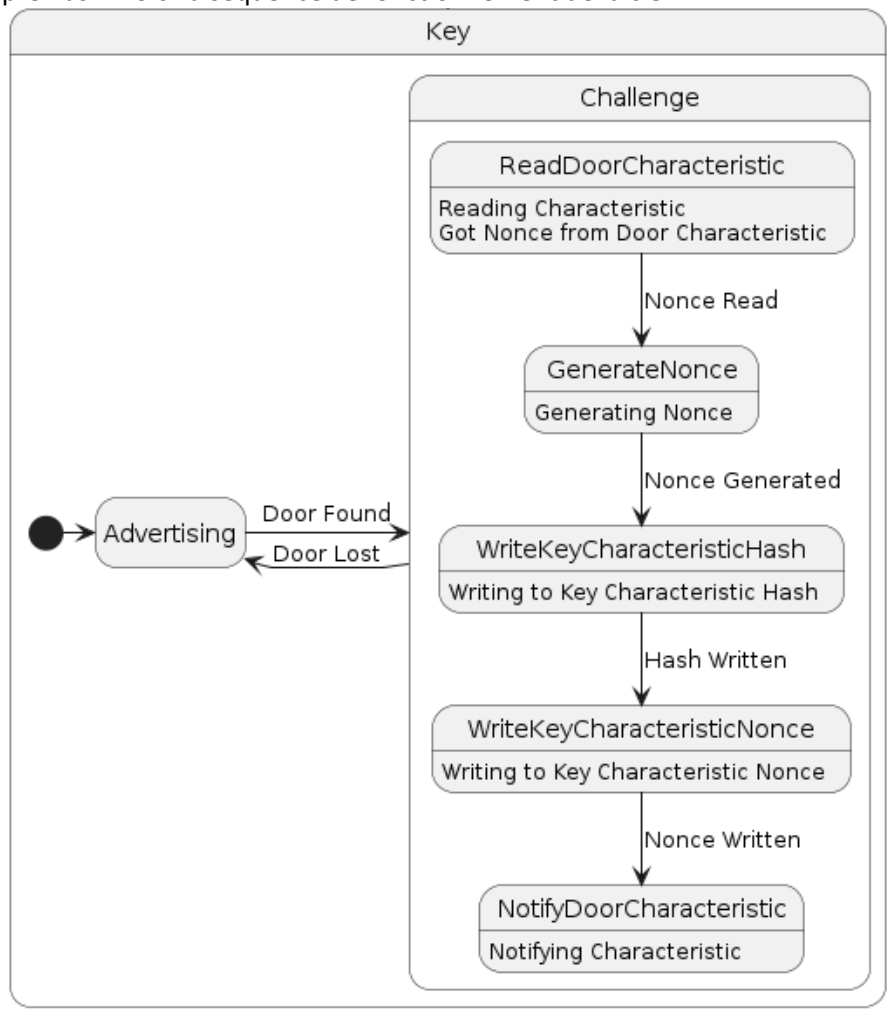


Où le client est la "clé" à tester et le serveur est le microcontrolleur qui s'occupe de la porte. La porte génère un nonce, puis l'envoie et attends de recevoir un hash généré par la clé en réponse, puis la porte compute un hash à partir du mot de passe prédéterminé et des deux nonces et enfin, la porte compare le hash reçu et celui-ci hash pour savoir si la clé à utiliser le bon mot de passe pour générer le hash. Si c'est le cas, la porte s'ouvre puis se referme après 10 secondes. Si ce n'est pas le cas, il ne se passe rien. L'utilisation des nonce pour la communication entre le paire clé/porte permet de se protéger des replays attack car à chaque nouvelle ouverture de porte, les nonces changent.

Pour illustrer le fonctionnement de la porte, voice sa séquence de fonctionnement:



L'autre partie du projet consistait à créer la clé. La clé annonce sa présence régulièrement (toutes les 250ms) et récupère le nonce de la porte avant de générer son hash avec le mot de passe et un autre nonce qu'elle génère. Le hash généré est alors envoyé à la porte pour qu'elle compare avec son propre hash. Voici la séquence de fonctionnement de la clé:



Enfin pour regarder plus en profondeur le code utilisé il faut se réferrer au github: https://github.com/SamuelGuillemet/NET4104-ESP32-Project

3 Interprétation des résultats

L'application marche parfaitement comme l'illustre la vidéo sur le github. Les principales difficultés sont venues de la taille limitée des données échangées par défaut entre la clé et la porte (4 octets par défaut contre 8 octets pour les hash échangés).

La disponibilité du service est assurée par des scans réguliers de la porte, toutes les 30 millisecondes et par des annonces régulières de la clé toutes les 250 millisecondes.

L'authenticité est assurée par l'utilisation d'un mot de passe commun entre la clé et la porte et par l'utilisation d'un hash avec des nonces pour éviter les écoutes et les replays attacks.

4 Manuel Utilisateur

Sous Linux:
sudo chmod a+rw /dev/ttyUSB0

Prérequis:

Téléchargez et installez la dernière version de Python 3.10

Téléchargez et installez le dernier bac pour l'ESP32S3 d'Espressif - (https://micropython.org/resources/firmware/GENERIC_S3-20220618-v1.19.1.bin)

Installez le logiciel ESP32S3 MicroPython

pip install esptool

esptool.py --chip esp32s3 --port /dev/ttyUSB0 erase_flash

esptool.py --chip esp32s3 --port /dev/ttyUSB0 write_flash -z 0x1000 GENERIC_S3-20220618-v1.19.1.bin

Installer mpremote:

pip install mpremote

Installer aioble micropython-lib aioble:

mpremote connect /dev/ttyUSB0 mip install aioble

Tester l'installation:

mpremote connect /dev/ttyUSB0 run ./src/test/test.py

Installer hashlib micropython-lib hashlib:

mpremote connect /dev/ttyUSB0 mip install hashlib

Une fois cela-fait, il suffit de porter le code vers les deux ESP-32 (l'un pour la clé, l'autre pour la porte) et de rapprocher la paire pour s'authentifier et pour actionner un servomoteur qui permettrait d'ouvrir une porte.