

PROJET ESP32 - SmartLock - NET 4104

DAUDE Tristan
GUILLEMET Samuel
SAFON Clément
COSTEL Alexandre

Enseignant responsable : Rémy Grünblatt

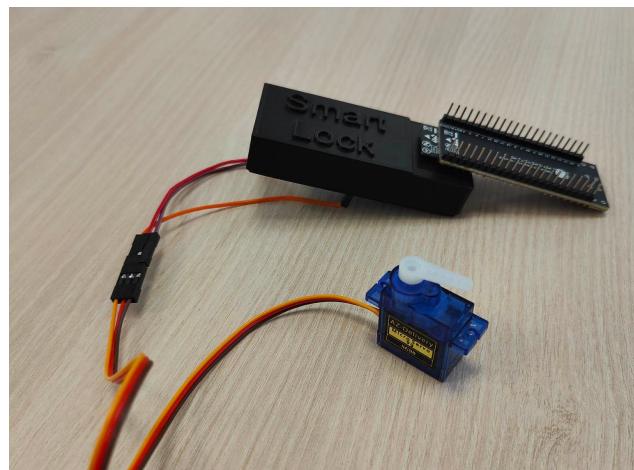
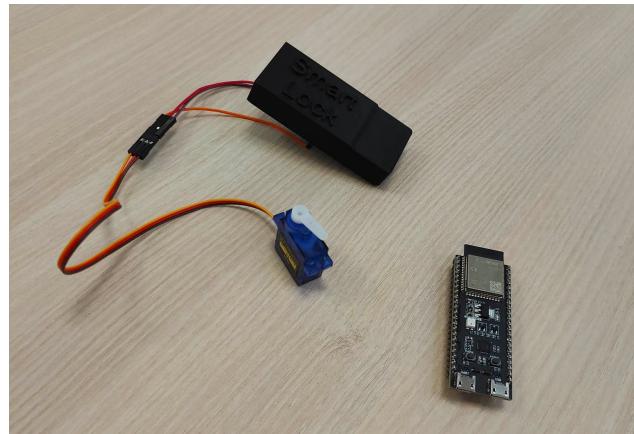


Table des matières

1) Introduction	3
2) Développement	3
3) Interprétation des résultats	5
4) Manuel utilisateur	6

Introduction

Notre projet consiste à développer un système de porte connectée grâce à plusieurs appareils Bluetooth ESP32. L'idée serait de créer un système clé/serrure en utilisant des ESP32 fonctionnant en Bluetooth BLE. Le déverrouillage de la porte s'effectuerait de la même manière qu'avec badge classique NFC. Pour cela nous utiliserons une whitelist d'adresses MAC ainsi qu'un processus d'authentification classique type challenge.

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

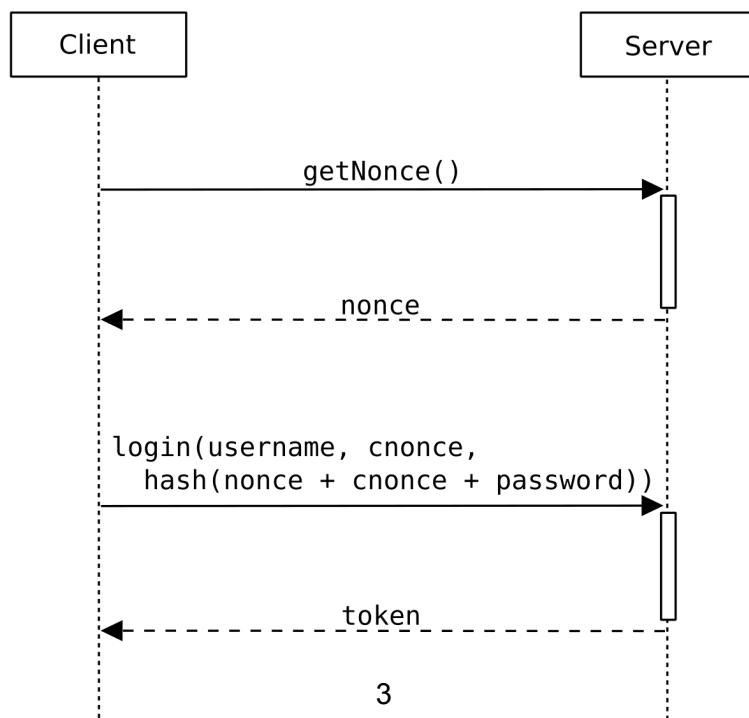
Dans ce document vous trouverez la conception, l'explication du fonctionnement ainsi qu'un manuel utilisateur.

Développement

Le projet est décomposé en 2 parties, le code de ESP32 qui correspond à la clé et le code de l'ESP32 qui correspond à la serrure qui actionne l'ouverture ou la fermeture de la porte.

Le principe est plutôt simple, dans un premier temps, on vérifie si l'adresse mac de l'objet Bluetooth correspond à une des adresse autorisées, si c'est le cas, on attend que celui-ci soit suffisamment proche en regardant la puissance de réception (RSSI). Enfin, une fois la clé suffisamment proche, on établit une connexion et l'authentification de la clé commence.

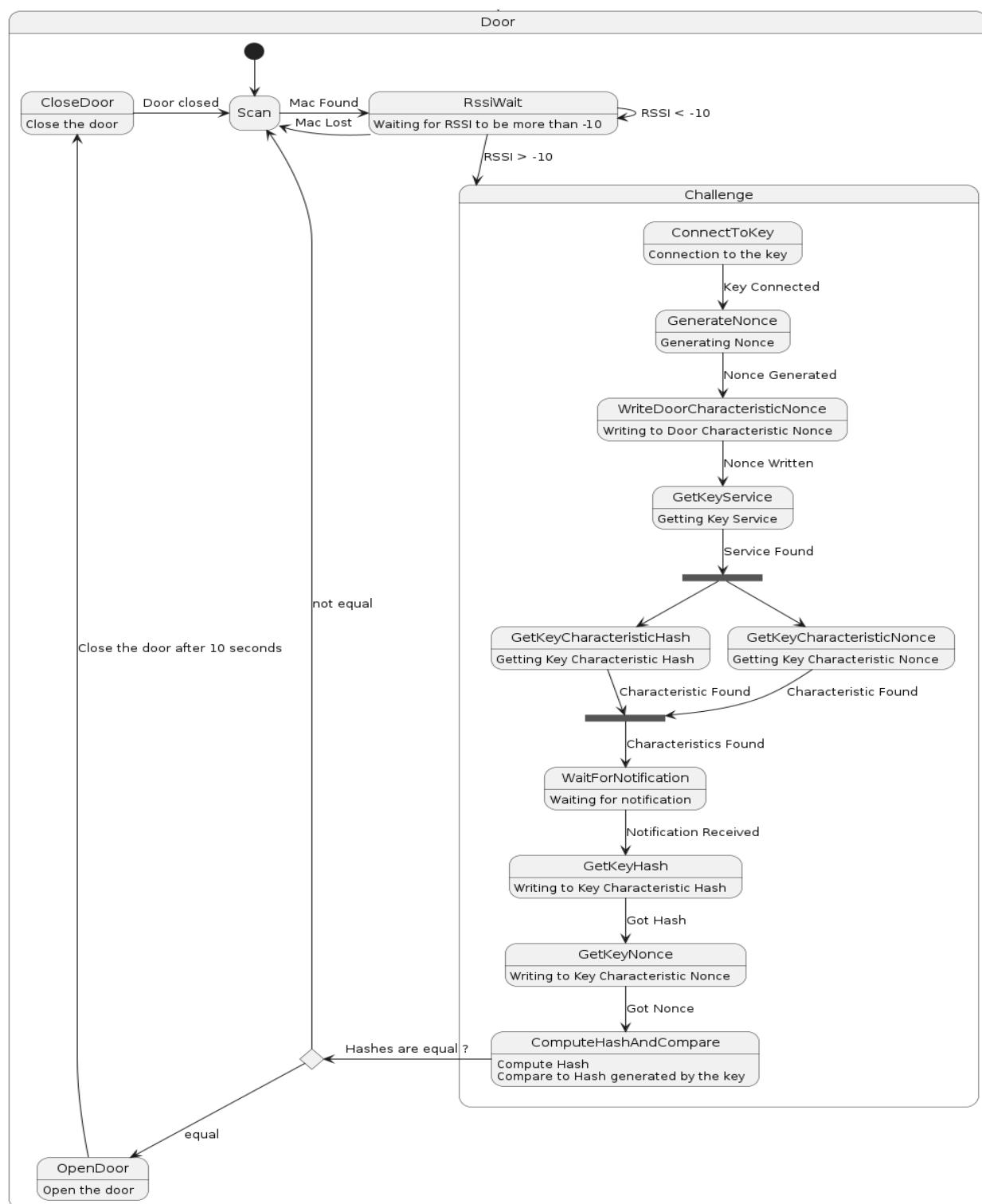
L'authentification est de type challenge. Chaque ESP32 possède une clé secrète (PSK) et il va falloir que la porte vérifie que ces deux clés sont identiques. Pour cela on se base sur l'algorithme ci-dessous (le client représente la clé tandis que le serveur représente la porte).



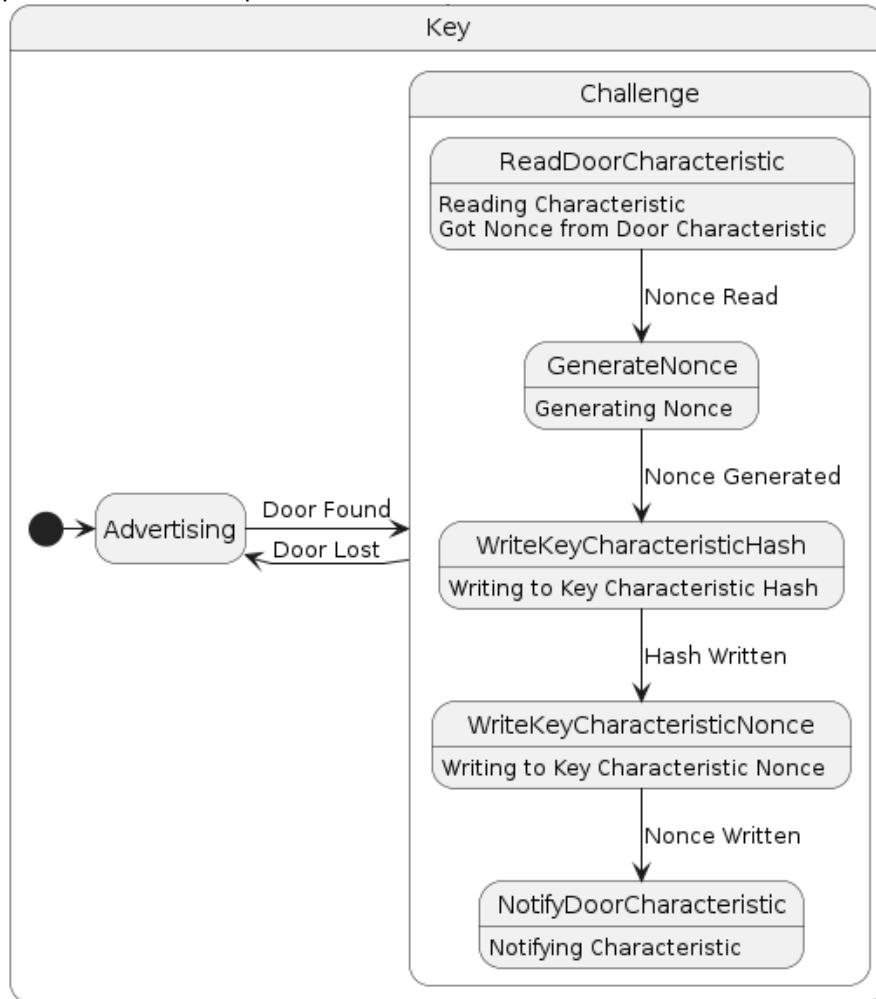
Dans notre projet, pour échanger les différentes données entre les deux ESP32 nous allons utiliser des Services (et Caractéristiques) basés sur le modèle GATT. Les étapes du protocole sont donc les étapes suivantes. D'abord, la porte génère un nonce (nombre aléatoire à usage unique), puis l'envoie à la clé. (Cette étape d'envoie est remplacée par l'écriture de ce nonce sur une caractéristique de la porte et la clé va venir lire cette caractéristique). Puis la serrure attend de recevoir un hash et un second nonce généré par la clé. (De la même manière ici on va venir lire certaines caractéristiques de la clé). La serrure va elle aussi calculer son hash de cette manière **hash = sha1(nonce + key + cnonce)** et le comparer au hash “reçu” de la clé. Si les deux hash sont égaux, la porte s'ouvre puis se referme après 10 secondes. Si ce n'est pas le cas, il ne se passe rien.

L'utilisation de cette méthode d'authentification entre le paire clé/porte permet de se protéger replay attack car à chaque nouvelle ouverture de porte, les nonces changent.

Pour illustrer le fonctionnement de la porte, voici 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 référer au github:
<https://github.com/SamuelGuillemet/NET4104-ESP32-Project>

Interprétation des résultats

L'application marche parfaitement comme l'illustre la vidéo sur le github. Les principales difficultés sont venues de l'échange de données entre les ESP32. Nous ne savions pas quoi utiliser avant de comprendre comment nous pourrions utiliser le modèle GATT pour échanger des données entre les deux appareils. De plus, nous avons aussi été confrontés à la taille limitée des données inscrites dans une caractéristique (4 octets par défaut contre 20 octets pour les hash sha1).

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.

Manuel Utilisateur

Vous trouverez le détail complet de l'utilisation via les différents markdown du projet.

<https://github.com/SamuelGuillemet/NET4104-ESP32-Project/blob/main/docs/installation.md>