NET4104 - Rapport de Projet

Inès Kacer, Thomas Perel,
Sarah Ramdani, Julien Vankrinkelen

Sommaire

Abstract.

Le BLE (Bluetooth Low Energy) est une technologie qui consomme très peu d'énergie. Cet avantage considérable dans le domaine de l'IoT fait du BLE la technologie de communication principale déployée sur des objets connectés tels que les montres connectées, téléphones, distributeurs de nourritures…

Contexte

Proposition 1: contrôle d'accès.

Dans un contexte d'entreprise qui possède des accès physique sensibles, notre projet pourrait être utilisé pour mesurer la distance entre un employé et la porte d'accès afin de s'assurer que celui-ci est à une distance sûre avant de lui autoriser l'accès.

Proposition 2: aide aux malvoyants

Notre projet pourrait aider des personnes malvoyantes à naviguer dans des environnements inconnus. En effet, en détectant la distance entre son téléphone portable et des balises placées dans l'environnement, le système peut fournir des instructions audio en temps réel pour guider la personne en question.

Description du projet

Le concept du projet est assez simple. Nous souhaitons, à partir d'une adresse MAC donnée, récupérer la distance à laquelle l'appareil associée se trouve de notre récepteur en exploitant la puissance reçue. ‘

Les étapes du projet

1. *Prise en main du micro contrôleur et compréhension de micropython*

Avant de démarrer le projet, il nous fallait comprendre le fonctionnement du microcontrôleur mis à notre disposition, d'y faire l'installation de micropython et de se l'approprier en y implémentant des petits programmes tels que le classique “Hello World”.

2. *Recherche et formation*

Après avoir maîtrisé les notions théoriques du Bluetooth Low Energy, il nous fallait trouver du code afin de l'étudier et de nous approprier cette technologie d'une manière plus pratique. Nous avons également trouvé sur le dépôt GitHub proposé dans le cadre du cours NET 4104 une fonction ***scan***, que nous avons utilisé telle quelle dans notre projet.

3. Code et expérimentation

Après cette longue phase de formation et de recherche, nous nous sommes lancés dans la phase de programmation et l'expérimentation.
Nous avions commencé par faire un programme qui, à partir d'une adresse MAC donnée, récupère la puissance émise par l'appareil associé. Notre but initial étant de calculer la distance à laquelle l'appareil se trouve, nous avons fait des expérimentations en nous déplaçant avec divers appareils (montre connectée, machine à café etc.), en comparant la puissance reçue par le microcontrôleur.
Nous nous sommes rapidement rendu compte que la puissance reçue variait beaucoup d'un appareil à un autre, mais surtout, que la puissance reçue variait beaucoup pour un même appareil à une distance donnée, ce qui était problématique.
Nous avons donc codé une fonction qui permet de calculer la puissance moyenne reçue sur un intervalle de temps à une distance donnée.
Pour optimiser ce programme, nous avons ajouté une fonction de calibration qui permet de calculer la puissance moyenne émise par un appareil dont on donne l'adresse MAC à un mètre du microcontrôleur. La puissance calculée permet par la suite de faire le calcul

de la distance de l'appareil quand le porteur se déplace.

Difficultés

La principale difficulté rencontrée lors de ce projet était dans la phase de recherche et formation. En effet, il y a peu de ressources disponibles concernant le Bluetooth Low Energy, et dans le dépôt GitHub mis à notre disposition, seul le readme.md nous donnait quelques explications. Nous devions donc nous plonger dans le code pour le décortiquer et se l'approprier, ce qui était fastidieux.

Limites

Comme énoncé dans les étapes du projet, une des limites de ce projet est lié à l'imprécision des mesures de la puissance reçue à une distance donnée.
De plus, nous nous plaçons dans un contexte où il n'y a pas d'obstacles entre l'utilisateur et le récepteur/microcontrôleur. En effet, dans un contexte réel, les obstacles tels que les meubles, les murs etc, affaiblissent la puissance reçue et faussent la distance résultante.
Enfin, l'ergonomie de notre programme est à revoir. Une amélioration serait de pouvoir naviguer plus facilement entre les fonctionnalités une fois dans le shell, et ne pas avoir à retaper *uasyncio.run(main.function())*. ou à *import* à chaque lancement du shell le code et la librairie uasyncio. Nous n'avons pas réussi à gérer les exceptions *Interrupt Keyboard* qui ne marchaient visiblement pas dans le shell picocom. On peut toujours interrompre un un programme avec ctrl-c mais l'exception *Interrupt Keyboard* de notre programme n'est pas appelée.