INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE
Département de Génie Électrique & Informatique

BE C++

Système de Commande de la Porte Intégré avec Alarme Anti-Intrusion

27 mai 2020

Auteurs :

Terence Chun Heng	LIEW	liew@etud.insa-toulouse.fr
Zheng	ZHOU	zzhou@etud.insa-toulouse.com

Enseignant :

Thierry MONTEIL Raphaël DEAU

Table des matières

Introduction

Dans le cadre de 4ème année de AE-SE à INSA Toulouse, nous avons réalisé un projet sur la conception et codage d'un *système de commande de la porte intégré avec alarme anti-intrusion*. Il s'agit d'un projet sur C++ et se fait en simulant une carte Arduino intégrée à divers *devices* simulés (capteurs et actionneurs).

1 La Conception

1.1 Use Case

Voici le Use Case Diagram montré en Figure 1.

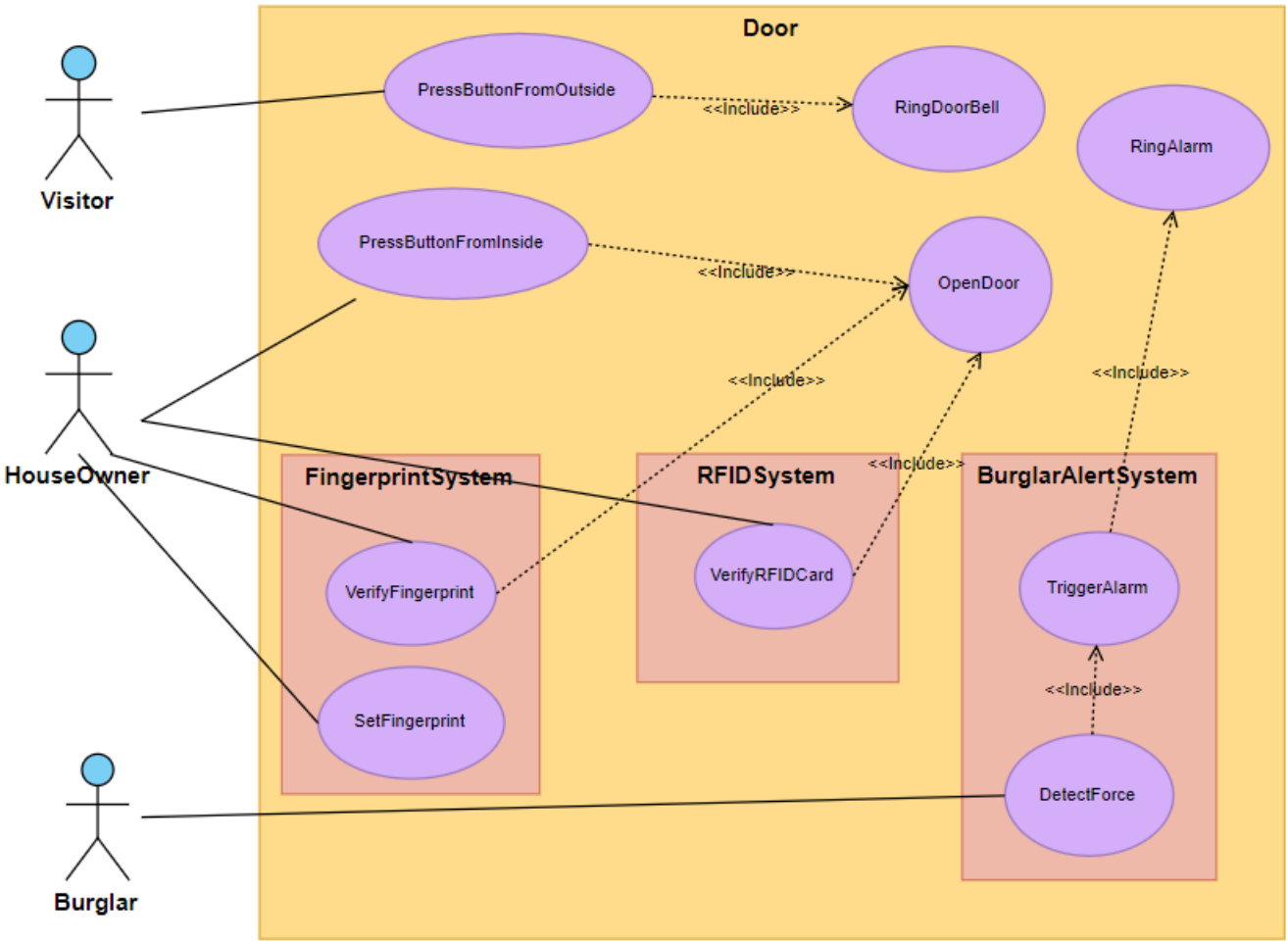

FIGURE 1 – Use Case Diagram

1.2 Diagramme de Classe

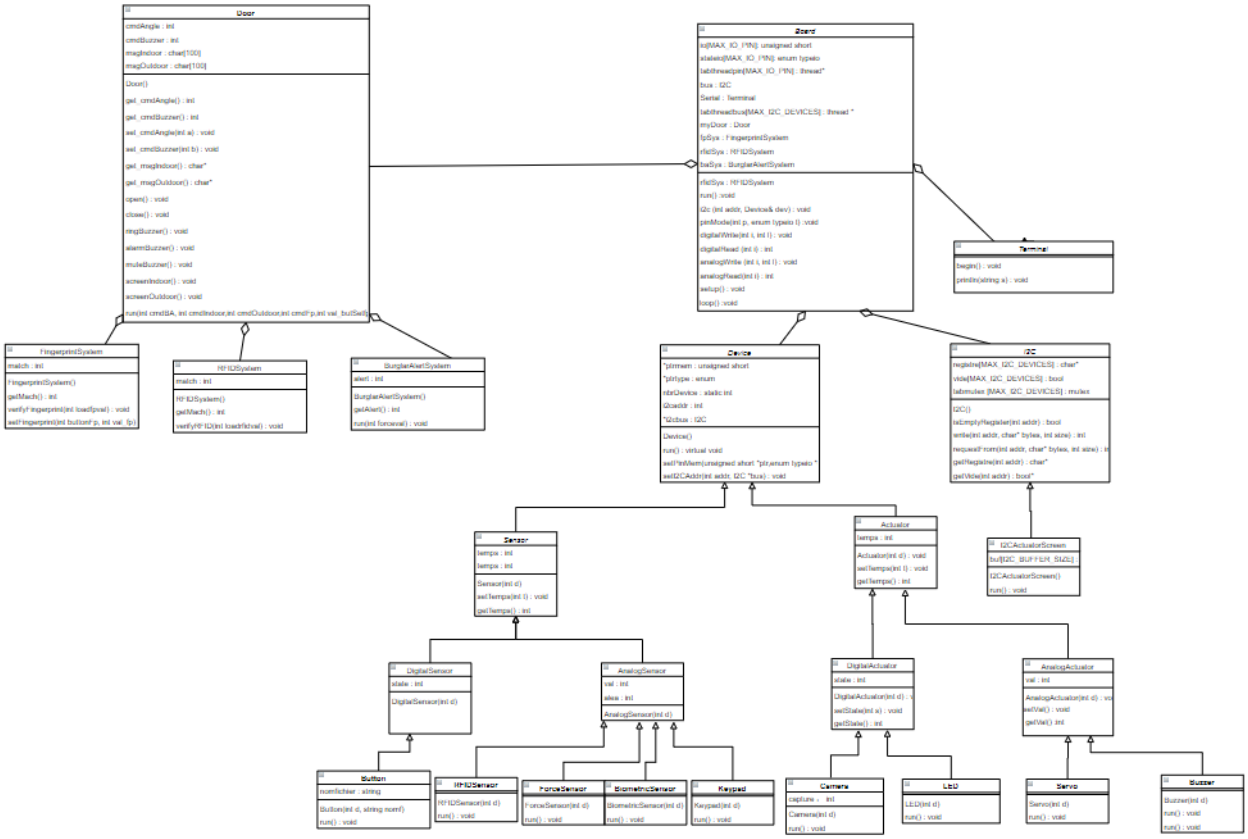

FIGURE 2 – Vue ensemble de La diagramme de classe

[1]Figure 2 montre le vue d'ensemble de la diagramme de classe. À partir des codes fournies par notre professeur, nous avons développer deux parties pour ce projet - «Hardware» et «Software». La section hardware contient toutes les classes de dispositif nécessaires à notre projet qui sont dans les fichiers «mydevices.cpp/.h». Et la section software contient toutes les classes d'applications qui sont dans les fichiers «applications.cpp/.h».

1.2.1 Partie «Hardware»

Dans cette partie, nous avons défini plusieurs devices. La classe «device» est une classe qui représente un capteur ou actionneur générique.Ensuite nous avons créé la classe capteur (analogique et numérique) et actionneur (analogique et numérique) en héritant la classe «device». A la fin, nous avons créé tous les devices. «I2C» est une classe qui gère le bus I2C, elle définit des zones d'échangés entre un dispositif et la carte arduino, en héritant de cette classe, «Screen» est capable d'afficher les informations nécessaires pour les utilisateurs. Les diagrammes de classes sont montré en Figure 3 et Figure 4. Vous pouvez noter que certaines *devices* ne sont pas encore utilisé dans notre projet, ces *devices* pourront être intégré dans la version future de notre projet.

1. Voir le lien suivant pour avoir une vue plus claire de la diagramme de classe :
`https://drive.google.com/file/d/1bilHjex4nsEI0kIL8P1TbbAgAxyc3B7n/view?usp=sharing`

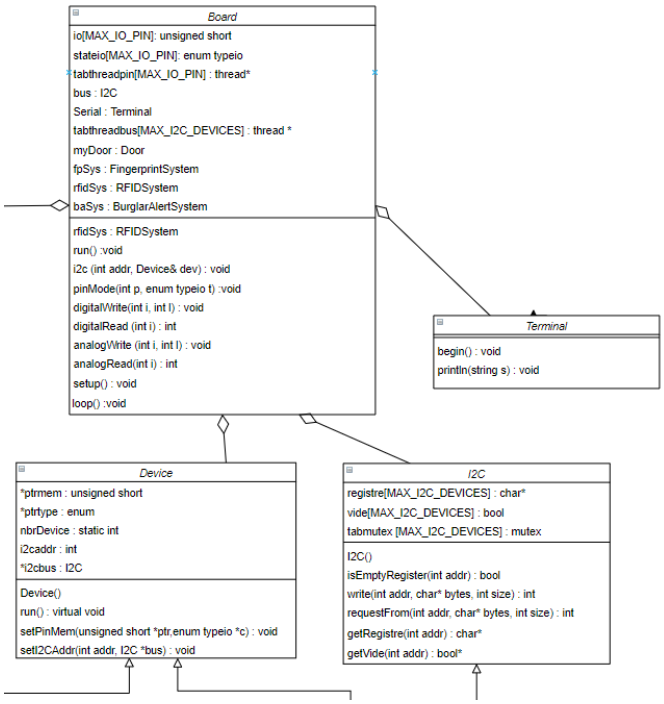

FIGURE 3 – Hardware (1)

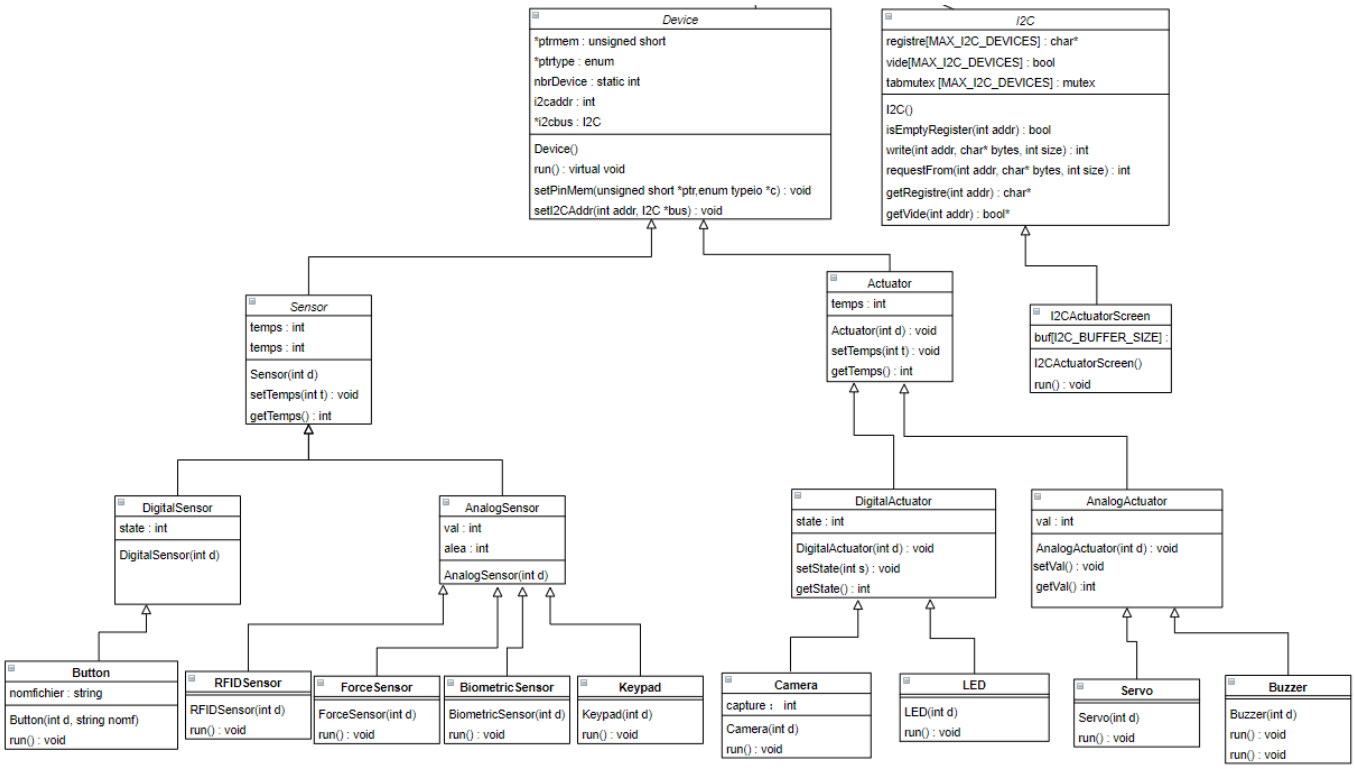

FIGURE 4 – Hardware (2)

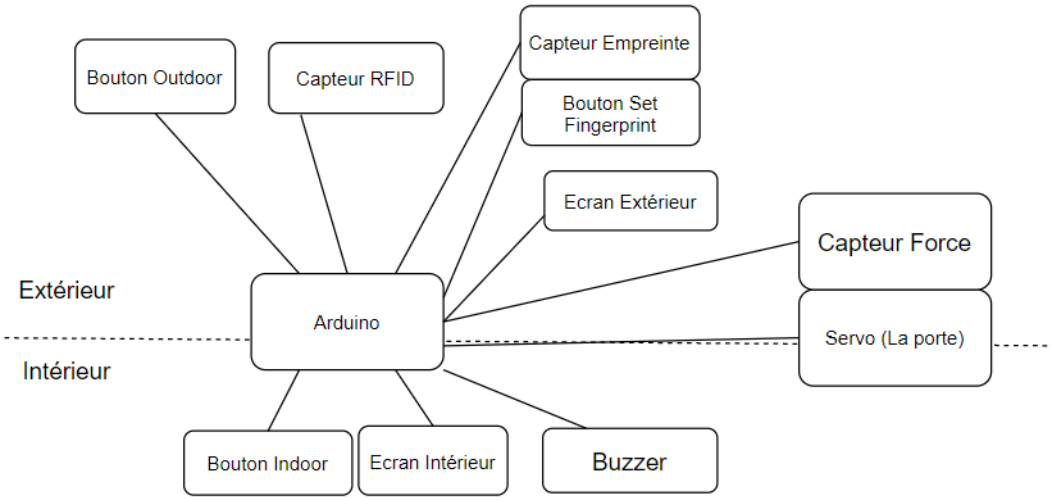

FIGURE 5 – Schéma de fonctionnement

Comme montre le schéma de fonctionnement en Figure 5, dans cette version de notre projet, nous avons utilisé 6 **capteurs** et 4 **actionneurs** :

Capteurs :

- 1 BiometricSensor (Fingerprint Sensor)
- 1 RFIDSensor
- 3 Button
- 1 ForceSensor

Actionneurs :

- 1 Servo (La porte)
- 1 Buzzer (L'alarme et La sonnette)
- 2 I2CActuatorScreen (1 écran à l'intérieur et 1 écran à l'extérieur)

1.2.2 Partie «Software»

Dans notre projet, il y a 4 systèmes principales «DoorSystem», «RFIDSystem», «FingerprintSystem», «BurglarAlertSystem».

1. **Door**
   Ce système est capable de contrôler la porte et l'écran intérieur et l'écran extérieurs et extérieurs. Quand on appuie sur le bouton intérieur, la porte va ouvrir. Si on appuie sur le bouton extérieur, le buzzer va sonner.

2. **RFIDSystem** Ce système est capable de détecter une carte RFID et de demande d'ouvrir la porte si la fréquence RFID correspond à la fréquence enregistrée.

3. **FingerprintSystem**
   Ce système peut détecter les empreintes digitales et de demande d'ouvrir la porte si une empreinte digitale est pareil que l'empreinte digitale enregistrée.

4. **BurglarAlertSystem**
   Ce système est capable de détecter la force exercée sur la porte grâce à un capteur de force. Un signal sera envoyé à un buzzer (identique à la sonnette mais avec une fréquence plus élevée) qui alarmera le propriétaire lorsque la force exercée sur la porte dépasse une limite définie (par défaut 88).

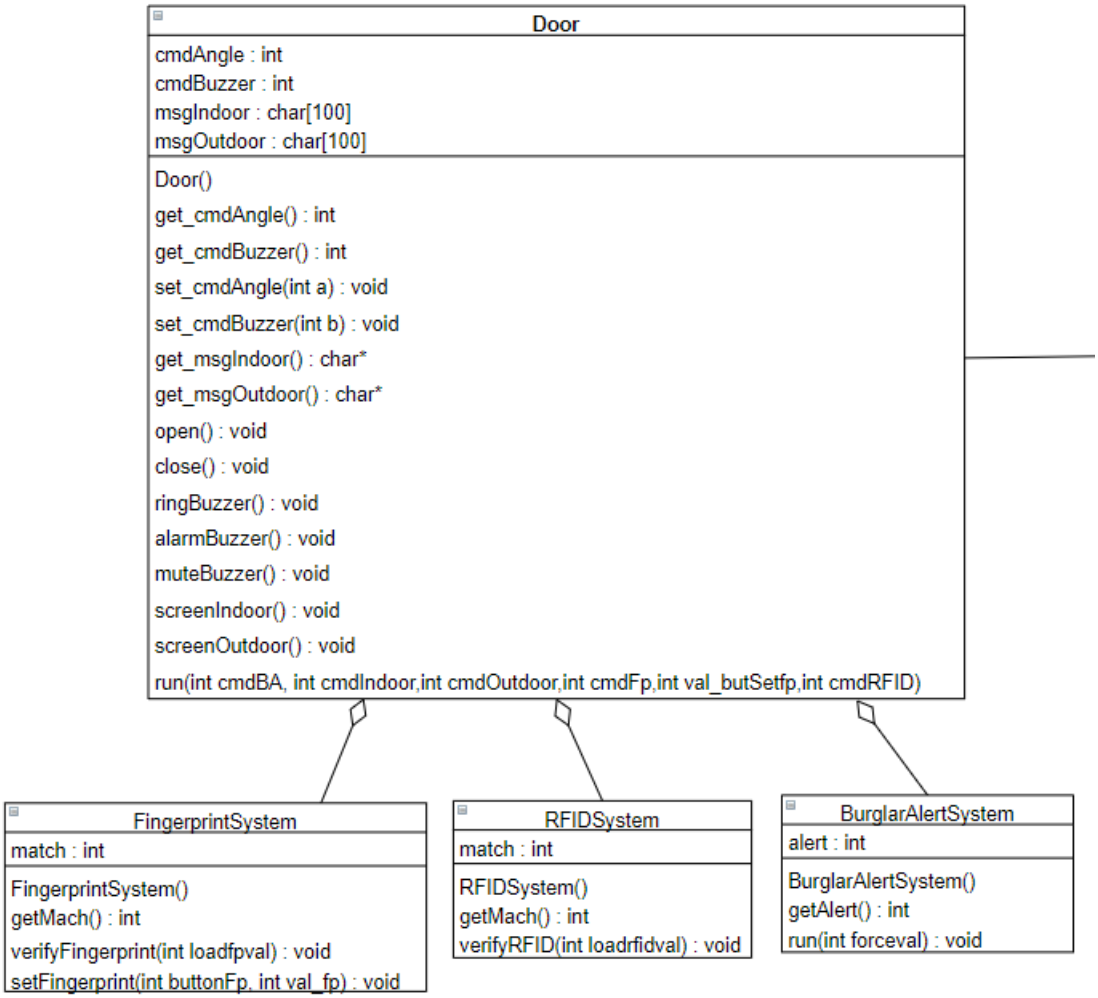

FIGURE 6 – Software

Pour la partie «software», nous avons choisi de représenter le diagramme de classe qui correspond à notre code (pour le cas de simulation). A noter que cette diagramme de classes sera différent si on construit ce projet en réel. Comme nous avons mis les classes «software» comme attributs de la classe Board, nous établissons donc une relation d'agrégation entre la classe *Board* et la classe *Door* qui gère toutes les fonctionnalités des systèmes de sécurité qui sont intégrés. En réel, il y aura d'autres liens entre les classes «software» et les classes des capteurs et actionneurs qu'on utilise.

2 Le Codage

2.1 Déroulement de codage

1. Codage des classes mères : Sensor, Actuator
2. Codage des classes Analog et Digital pour les classes mères : AnalogSensor, DigitalSensor, AnalogActuator, DigitalActuator
3. Codage des classes filles capteurs et actionneurs : Button, RFIDSensor, ForceSensor, BiometricSensor, Servo, Buzzer
4. Codage des applications : Door, FingerprintSystem, RFIDSystem, BurglarAlertSystem
5. Codage dans le fichier *Board* pour instancier notre *devices* et les connecter avec les pins du *Board*.
6. Codage de la méthode *setup* du *Board* pour configurer les pin en entrée et en sortie en fonction des capteurs et actionneurs
7. Codage de la méthode loop :
   (a) Récupération des valeurs de capteurs
   (b) Affichage sur Terminal les valeurs des capteurs
   (c) Appel de software
   (d) Faire les commandes des actionneurs
   (e) Affichage sur Terminal les valeurs des actionneurs
   (f) Gestion affichage des écrans Indoor et Outdoor

2.2 Problèmes et Résolution

2.2.1 Instanciation des classes «software»

Au début, nous posons des questions sur comment instancier les classes de «software» que nous avons défini. Pour répondre à cette question, il faut savoir où est-ce qu'on appelles ses méthodes. La réponse étant le "main" ou le boucle de notre programme qui dans ce cas là, c'est *Board : :loop*. Cette méthode ne connaît que deux types de variable, la variable globale et les attributs de la classe *Board*. Nous avons donc choisi de mettre les classes «software» comme attributs de la classe *Board*, ce qui permet d'instancier ces classes pendant l'instanciation de *Board*.

2.2.2 Lecture et Écriture des Capteurs et Actionneurs

Au début nous avons pensé de faire les lectures et écritures (*analogRead, analogWrite, digitalRead et digitalWrite*) des capteurs directement sur notre classes de «software». Mais cela n'est pas faisable parce que ces méthodes sont propre à la classe *Board*. Nous avons donc opté pour la solution de ne pas utiliser ces méthodes directement dans notre classe «software». Pour faire fonctionner notre application, nous avons choisi de récupérer tous les valeurs de capteurs et passer ces valeurs comme arguments et de stocker tous les commandes des actionneurs dans les attributs (par exemple : cmdAngle et cmdBuzzer) de la classe «software» *Door* qui sera accessible par *Board : :loop* en utilisant les *getters*.

3 Le programme

3.1 Comment utiliser le programme ?

Voici comment tester et interagir avec le programme.

3.1.1 Indoor Button (Bouton d'ouverture de la porte de l'intérieur)

— Pendant l'exécution du programme, créer une fichier *.txt* avec le nom *indoor.txt* dans le dossier *src* pour simuler l'ouverture de la porte.
— Supprimer le fichier pour simuler la fermeture de la porte.

3.1.2 Outdoor Button (Sonnette)

— Pendant l'exécution du programme, créer une fichier *.txt* avec le nom *outdoor.txt* dans le dossier *src* pour sonner la sonnette à 480 MHz.
— Supprimer le fichier pour couper le son de la sonnette.

3.1.3 Fingerprint System

— Pendant l'exécution du programme, par défaut, l'ID de l'empreinte n'est pas identique à celle enregistrée.
— Pour ouvrir la porte, changer l'ID dans le fichier *loadfp.txt* pour qu'il soit pareil que celle dans *savedfp.txt*.
— Pour fermer la porte, changer l'ID dans le fichier *loadfp.txt* pour qu'il soit différent de celle dans *savedfp.txt*.
— Pour enregistrer une nouvelle empreinte, écrire l'ID de l'empreinte que vous souhaitez enregistrer dans *loadfp.txt* et puis créer une fichier avec le nom *setFp.txt* dans le dossier *src*. La nouvelle ID de l'empreinte sera copiée automatiquement dans le fichier *savedfp.txt*.

3.1.4 RFID System

— Pendant l'exécution du programme, par défaut, la fréquence RFID détectée n'est pas identique à celle enregistrée.
— Pour ouvrir la porte, changer la fréquence RFID dans le fichier *loadrfid.txt* pour qu'il soit pareil que celle dans *savedrfid.txt*.
— Pour fermer la porte, changer la fréquence RFID dans le fichier *loadrfid.txt* pour qu'il soit différent de celle dans *savedrfid.txt*.

3.1.5 Burglar Alert System (Alarme anti-intrusion)

— La valeur de force est enregistrée dans le fichier *force.txt*. Par défaut, elle est de valeur 10.
— Pour déclencher l'alarme, écrire une valeur plus grande que la limite de force (88 par défaut). Le buzzer va sonner à 500 MHz.
— Pour fermer l'alarme, diminuer la valeur pour qu'il soit plus petite que la limite. Cela va donc couper le son de buzzer.

3.2 Analyse du programme fait & Perspective d'évolution

3.2.1 Set Fingerprint

La modification de l'empreinte entraîne que la variable *match = 1* dans la classe *FingerprintSystem*. Pour empêcher d'ouverture de la porte pendant cette phase, nous avons modifier le programme afin que l'ouverture de porte ne fait pendant qu'il existe pas le fichier *setFp.txt*. Mais la porte est ouvert dès qu'on supprime cette fichier ce qui ne permet pas d'utilisateur d'avoir le temps de changer ou enlever l'empreinte comme en réel. Une suggestion pour améliorer cette système est donc d'ajouter une mécanisme de compteur afin d'attendre quelque seconds avant de relancer le système après le supprime de fichier *setFp.txt*.

3.2.2 Exception

Nous avons imposer une limitation sur certains valeurs de capteurs suivant :

— L'ID de l'empreinte detectée (0-55555)
— La fréquence RFID (0-44444)
— La force exercée sur la porte (0-100)

Le but d'inclure ces exceptions étant d'empêcher l'utilisateur de taper n'importe quelle valeurs dans les fichiers pour ses capteurs. Cela risque d'avoir les valeurs de capteurs qui sont faussés parce que la valeur de chaque capteur étant stocké sur un pointeur de mémoire de taille (0-65535 *unsigned short*).

Pourtant, ces exceptions ne seront pas capté si la valeur inséré par l'utilisateur dans ces fichiers est plus grandes que 65535 parce que C++ peut "'*caster*" une *int* à *unsigned short* implicitement. Une suggestion pour faire évoluer ce programme est donc d'ajouter une gestion de l'*exception* au niveau de la classe Device afin de jeter ses *exception* tout de suite pendant lecture du fichier et pas pendant appel de *analogRead*.

3.2.3 Ajout des Fonctionnalités

Dans notre diagramme de classe nous avons ajouté plusieurs classes ne sont pas utilisé dans cette version de projet. Ces classes de devices pourront être construites pour améliorer la sécurité de la porte. Voici quelques exemple de fonctionnalités qu'on pourra ajouter :

— **Caméra de Surveillance**
Un *device* Caméra peut-être ajouté afin de faire une capture d'image pendant le declenchement de l'alarme anti-intrusion. Cela permet de capturer une image du malfaiteur.

— **Identification par mot de passe**
On peut ajouter un *keypad* comme *device* pour permettre une autre façon d'identification dans le cas où l'utilisateur a perdu sa carte RFID ou simplement d'autoriser d'autre personne à ouvrir la porte si l'utilisateur principal n'est pas disponible.

— **Lumière Intelligent**
On peut ajouter aussi les *LEDs Intelligent* et un capteur de luminosité comme défini pendant la prise en main de simulation au début de séance. Cela permet d'éclairer la porte quand l'environnement de la porte devient sombre.

Conclusion

Ce projet nous a permet de mettre en pratique notre connaissance sur la programmation objet orientée non seulement sur la partie conception, mais aussi sur le codage. Malgré le fait de ne pas pouvoir utiliser des capteurs et actionneurs en réel, nous avons beaucoup appris sur les démarches qu'il faut prendre pour concevoir un *smart system*. En outre, nous avons appris beaucoup sur la programmation en langage C++ notamment sur la notion d'objet et classes, le polymorphisme, l'utilisation d'exception, et aussi sur la gestion de fichier. Le fait de collaborer en utilisant le plateforme GitHub nous a permis de prendre en main cette outil qui est souvent utilisé en industrie.

Annexe

— **GitHub** :
`https://github.com/terenceliew/BE_Board_LIEW_ZHOU`

— **Diagramme de Classe** :
`https://drive.google.com/file/d/1bilHjex4nsEI0kIL8P1TbbAgAxyc3B7n/view?usp=sharing`