Küenzi Jean-Daniel|jean-daniel.knz@eduge.ch

CFPT-I | 04.2018

Documentation Technique

RFID PIGGY BANK

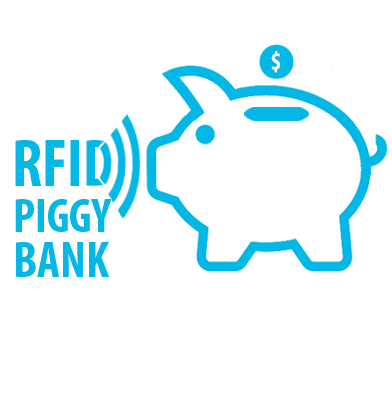


Table des matières

[1 Tableau des révisions 3](#_Toc514783832)

[2 Introduction 4](#_Toc514783833)

[3 Pourquoi j’ai choisi ce projet 4](#_Toc514783834)

[3.1 Pourquoi un microcontrôleur en C# 4](#_Toc514783835)

[3.2 Pourquoi la technologie RFID 4](#_Toc514783836)

[4 Rappel du cahier des charges 5](#_Toc514783837)

[4.1 But 5](#_Toc514783838)

[4.2 Spécifications 5](#_Toc514783839)

[4.3 Restrictions 5](#_Toc514783840)

[4.4 Environnement 5](#_Toc514783841)

[4.5 Livrables 5](#_Toc514783842)

[5 Analyse fonctionnelle 6](#_Toc514783843)

[5.1 Fonctionnalités 6](#_Toc514783844)

[5.1.1 Déverrouiller / Verrouiller la boite 6](#_Toc514783845)

[5.1.2 Ajouter un badge 6](#_Toc514783846)

[5.1.3 Supprimer un badge 6](#_Toc514783847)

[5.1.4 Déverrouiller via le code secret 6](#_Toc514783848)

[5.1.5 Verrouillage automatique 6](#_Toc514783849)

[5.2 Cas d’utilisations (Use Cases) 6](#_Toc514783850)

[6 Analyse organique 9](#_Toc514783851)

[6.1 Connectique 9](#_Toc514783852)

[6.2 Diagramme de classes 10](#_Toc514783853)

[6.3 Vue 11](#_Toc514783854)

[6.3.1 Program 11](#_Toc514783855)

[6.4 Modèle 11](#_Toc514783856)

[6.4.1 LCD 11](#_Toc514783857)

[6.4.2 RFIDReader 11](#_Toc514783858)

[6.4.3 ServoMotor 11](#_Toc514783859)

[6.4.4 ListOfCards 11](#_Toc514783860)

[6.4.5 Card 11](#_Toc514783861)

[6.4.6 SDCard 11](#_Toc514783862)

[6.5 Machines d’etats 12](#_Toc514783863)

[6.5.1 Scan d’un badge 12](#_Toc514783864)

[6.5.2 Ajout d’un badge 12](#_Toc514783865)

[6.5.3 Suppression d’un badge 13](#_Toc514783866)

[6.5.4 Dévérrouiller avec le code secret 13](#_Toc514783867)

[7 Tests 14](#_Toc514783868)

[7.1 Test Unitaires 14](#_Toc514783869)

[7.2 Test Use Case 14](#_Toc514783870)

[8 Planning 15](#_Toc514783871)

[9 Conclusion 16](#_Toc514783872)

# Tableau des révisions

|  |  |  |
| --- | --- | --- |
| Version | Description | Date |
| 1.0 | Création de la doc | 08.05.2018 |
| 1.1 | Ajout de la partie fonctionnalités | 09.05.2018 |
| 1.2 | Ajout de la partie use cases | 11.05.2018 |
| 1.3 | Modification global de la doc + ajout de la partie analyse organique | 17.05.2018 |
| 1.4 | Ajout machine d’état + description diagramme de classe | 22.05.2018 |

# Introduction

Cette documentation a pour but de détailler le fonctionnement de l’application web développée dans le cadre du TPI (Travail Pratique Individuel). Elle est destinée aux experts évaluant le travail ainsi qu’aux personnes susceptibles de la continuer.

Pour le cadre de ce travail j’ai décidé de développer une application microcontrôleur utilisant le C# comme langage. Le but de ce travail est d’avoir un microcontrôleur fonctionnel qui permet à un utilisateur de remplacer une tirelire ou un coffre-fort qui s’ouvre avec des méthodes standard (Code, Clefs, etc.) par une boite qui s’ouvre avec un système de badge RFID.

Cette réalisation me permet d’approfondir mes connaissances en C# et en microcontrôleur et de les appliquées dans le cadre d’un travail concret. Ce travail relate toutes mes connaissances et expériences acquise durant ma formation.

# Pourquoi j’ai choisi ce projet

## Pourquoi un microcontrôleur en C#

Tout d’abord, si j’ai choisi de faire une application microcontrôleur en C# plutôt qu’une application web c’est parce que je n’étais pas motivé à faire du web. Lors de ma 3ème année de formation au CFPT, notre classe à expérimenté une nouvelle branche de l’école qui s’appelle « Ecole Entreprise ». Pendant l’année entière nous devions développer un projet pour l’école. Mon équipe et moi avons été mandaté par le directeur du CFPT, M. Martinez, afin de développer un projet qui permettrai de faciliter la recherche de stage pour les techniciens et les futurs élèves de l’Ecole Entreprise. Nous avions le choix de la plateforme et nous avions donc décider de se lancer sur un projet web. Du coup pendant l’année entière nous avons développé un site utilisant les technologies ajax, java script, php, jquery, bootstrap, etc.

Le fait d’avoir développé un site web pendant toutes une année ne m’a pas « dégouté » du web, au contraire en voyant toutes les possibilités qui s’ouvre au monde du web ça m’a donné envie de continuer à approfondir mes connaissances. Cependant je voulais changer d’optique, voire de nouvelles choses, le monde informatique est très vaste. Voilà comment j’en suis venu à réaliser une application microcontrôleur. Lors de la formation nous n’avons pas fait beaucoup de microcontrôleur et c’était pour moi une chance de réaliser un travail concret pendant le TPI sur un microcontrôleur. Pour le langage de programmation (C#) et l’environnement de programmation (FezSpider, Gadgeteer, GHI) c’est surtout pour un confort, je me retrouve dans un environnement de développement que je connais bien.

## Pourquoi la technologie RFID

Lors de l’atelier « Nouvelles Technologies », cette année. J’ai pu étudier la technologie du RFID. Ce sujet m’a beaucoup plus et m’a surtout très intéressé. Aussi, cette technologie commence à beaucoup se répandre pour diverses utilisations (ouverture d’une porte, d’une barrière, etc.) et je voulais voir si j’étais capable d’adapter cette technologie à mes idées.

# Rappel du cahier des charges

## But

Le but est de créer un microcontrôleur capable de gérer l’ouverture et la fermeture d’une boite (sorte de petit coffre-fort) à l’approche d’un badge RFID.

L’application sera développée en utilisant le design pattern Modèle-Vue (MV)

## Spécifications

Le microcontrôleur est capable de :

* Déverrouiller / Verrouiller la boite à l’approche d’un badge RFID valide
* Supprimer / Ajouter des badges RFID
* Nommer un badge à l’ajout de celui-ci
* Déverrouiller la boite via un code secretdonné (code secretnon modifiable)
* Verrouiller la boite automatiquement en cas d’oubli (sécurité)

## Restrictions

Le microcontrôleur n’est pas capable de :

* Renommer les badges
* Modifier le code secret pour déverrouiller
* Être tactile

## Environnement

Matériel nécessaire au développement de l’application :

* Ordinateur de type PC
* Carte Microcontrôleur Fez Spider 1, Composants GHI + Makeblock + Carte SD (Stockage)
* Système d’exploitation : Windows 10 Entreprise
* Outil de développement : Visual Studio 2013
* Extension de développement : GHI Gadgeteer core, Micro .NET Framework SDK
* Outil de sauvegarde : Git[[1]](#footnote-1), Drive[[2]](#footnote-2), Local

## Livrables

* Documentation technique + Manuel utilisateur
* Planning prévu + Planning effectif
* Microcontrôleur (Application)

# Analyse fonctionnelle

## Fonctionnalités

### Déverrouiller / Verrouiller la boite

L’utilisateur aura la possibilité de déverrouiller / verrouiller sa boite à l’approche d’un badge RFID valide (reconnu par le microcontrôleur).

### Ajouter un badge

L’utilisateur pourra ajouter des badges RFID afin que le microcontrôleur les reconnaisse.

#### Nommer un badge

Au moment de l’ajout du badge, l’utilisateur pourra nommer son badge. **Attention on ne peut pas renommer les badges, cette action est irréversible !**

### Supprimer un badge

L’utilisateur aura la possibilité de voir la liste des badges acceptés et de supprimer des badges.

### Déverrouiller via le code secret

Si l’utilisateur perds ses badges, cette fonctionnalité permet de déverrouiller la boite via un code secret. L’utilisateur devra utiliser le joystick pour exécuter le code secret.

### Verrouillage automatique

Cette fonctionnalité permet de verrouiller la boite après certain temps si jamais l’utilisateur oubli de la verrouiller lui-même.

## Cas d’utilisations (Use Cases)

**Nom :** Déverrouiller la boite **Acteur principal :** Utilisateur  
**Pré-requis :** La boite est verrouiller **Déclencheur :** L’utilisateur approche un badge de la boite

**Flot principal :**

1. Le badge est scanné
2. Le badge est valide (présent dans la white liste)
3. Le système informe l’utilisateur que le badge est valide et déverrouille la boite

**Flot alternatif :**

2a) Le badge n’est pas valide

2a1) Le système informe l’utilisateur que le badge n’est pas valide

**Nom :** Verrouiller la boite **Acteur principal :** Utilisateur  
**Pré-requis :** La boite est déverrouiller **Déclencheur :** L’utilisateur approche un badge de la boite

**Flot principal :**

1. Le badge est scanné
2. Le système verrouille la boite

**Nom :** Ajouter un badge **Acteur principal :** Utilisateur **Déclencheur :** L’utilisateur sélectionne le menu « Ajouter un badge »

**Flot principal :**

1. Le système demande à l’utilisateur de scanner un badge
2. L’utilisateur scanne un badge
3. Le système affiche l’uid du badge et un nom par défaut
4. L’utilisateur valide le nom
5. Le système ajoute le badge et affiche le menu principal

**Flot alternatif :**

4a) L’utilisateur décide de nommer différemment son badge

4a1) L’utilisateur entre le nom qu’il souhaite et valide  
 4a2) Le système vérifie le nom et valide  
 4a3) Le système ajoute le badge et affiche le menu principal

**Nom :** Supprimer un badge **Acteur principal :** Utilisateur **Déclencheur :** L’utilisateur sélectionne le menu « Supprimer un badge »

**Flot principal :**

1. Le système affiche la liste des badges valides
2. L’utilisateur sélectionne un badge et clique sur « supprimer »
3. Le système lui demande de confirmer
4. L’utilisateur confirme
5. Le système supprime le badge et affiche le menu principal

**Flot alternatif :**

2a) L’utilisateur clique sur « annuler »

2a1) Le système annule et affiche le menu principal

4a) L’utilisateur ne confirme pas

4a1) Le système annule et affiche le menu principal

**Nom :** Déverrouiller avec le code secret **Acteur principal :** Utilisateur **Déclencheur :** L’utilisateur sélectionne le menu « code secret »

**Flot principal :**

1. Le système demande à l’utilisateur d’entrer le code secret
2. L’utilisateur entre le code correctement
3. Le système déverrouille la boite

**Flot alternatif :**

2a) L’utilisateur n’entre pas le code correctement

2a1) Le système lui demande de recommencer

2b) L’utilisateur clique sur « Annuler »

2b1) Le système annule et affiche le menu principal

# Analyse organique

## Connectique

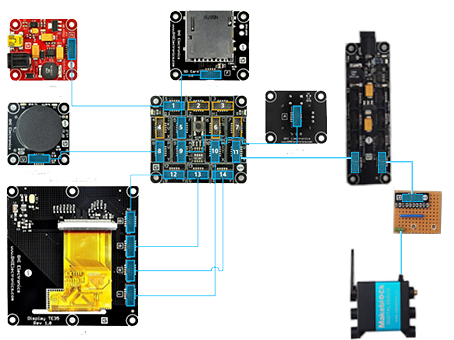


Figure 1 – Schéma connectique

Ce schéma représente la manière dont mes modules sont connectés à ma carte FezSpider (élément central) :

|  |  |  |  |
| --- | --- | --- | --- |
| Module | Constructeur | Utilisation | Socket(s) |
| USB Client DP | GHI | Alimente la carte FezSpider et la connecte au PC (Compilation, etc…) | 1 |
| SDCard | GHI | Sauvegarde les badges RFID (Remplace la base de données) | 5 |
| RFIDReader | GHI | Lit les badges RFID qui s’en approche | 8 |
| Joystick | GHI | Permet diverses actions (déplacement dans le menu, sélections, etc…) | 9 |
| Power Extender | GHI | Alimente le servomoteur et permet l’envoi de données | 11 |
| Servo Connector | Küenzi Jean-Daniel | Fait office de pont entre le servomoteur et le power extender | 11 |
| Digital Servo | Makeblock | Verrouille / Déverrouille la boite (fait office de loquet) | 11 |
| TE35 Display | GHI | Sert à avoir un affichage graphique | 10, 12, 13, 14 |

## Diagramme de classes

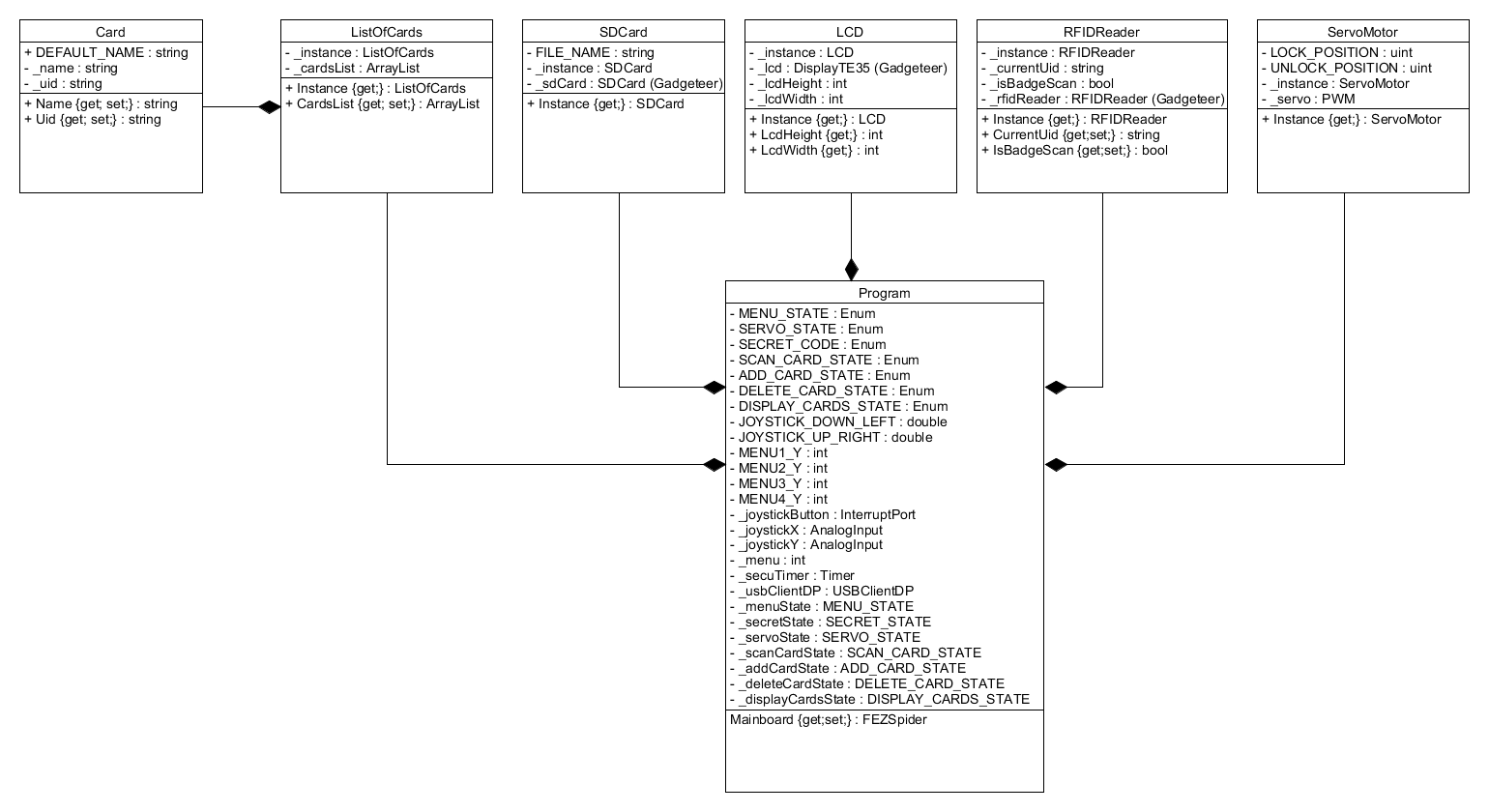


Figure 2 - Diagramme de classes

## Vue

### Program

La classe Program est en quelques sortes ma vue, c’est là que toutes la logique et les appels aux modèles vont être effectuée. Elle n’instancie pas directement les autres classes car ce sont des singletons.

Tous les composants qui y sont branché et initialisé sont défini par rapport à mes placements sur ma carte. Aussi la valeur du joystick ainsi que ses actions sont définies par rapport à l’orientation que je lui ai donné sur la carte. Ce ne sera donc pas les mêmes valeurs pour la positions X et Y du joystick en fonction de son orientation.

La propriété **Mainboard** est le type de carte que l’on utilise, si l’on utilise une FezCobra (par exemple) il suffit juste de changer la valeur de la propriété dans le programme pour qu’il reconnaisse la carte.

## Modèle

### LCD

La classe LCD permet d’utiliser le TE35 Display de GHI. Elle utilise le design pattern singletons. Elle contient 2 méthodes, une qui permet d’écrire sur le LCD et une qui permet d’effacer.

Le module TE35 Display est instancier directement dans la classe avec les sockets qui lui sont attribué, il est parfaitement possible de changer le modèle du LCD que l’on veut utiliser avec la classe cependant il faut changer directement dans la classe le type.

Les variables **\_lcdWidth** et **\_lcdHeight** s’instancie avec les propriétés du modules LCD que l’on utilise.

### RFIDReader

La classe RFIDReader permet de gérer le module RFIDReader de GHI. Elle utilise le design pattern singletons. Elle contient 2 méthodes, une pour les badges qui ont été correctement scanné et une si jamais un badge a mal été scanné.

De base les méthodes du RFIDReader ne sont pas prévues pour retourner quelque chose, il faut donc créer soit même sont moyen de retourner une valeur quand un badge est scanné. C’est pour ça que dans la classe RFIDReader j’ai ajouté 2 variables :

* **\_isBadgeScan 🡪** Elle vaut True quand un badge est scanné. Ensuite il faut la refaire passer à False manuellement pour pouvoir la réutilisez dans le programme.
* **\_currentUid 🡪** Elle sert à stocker l’Uid du badge scanné afin de pouvoir la retourner ensuite. On peut changer sa valeur manuellement aussi.

### ServoMotor

La classe ServoMotor permet de gérer le Servomoteur que j’utilise (Digital servo de Makeblock). Elle utilise le design pattern singletons. Elle contient 2 méthodes, une qui Lock (ferme) et une qui Unlock (ouvre) mon servomoteur.

Le servo est une variable de type **PWM** (modulation de largeur d’impulsion), on peut donc en modifier ça période ainsi que sa durée. Le servo est instancié avec des paramètres par défaut que l’on peut changer au besoin. Pour pouvoir faire bouger le servomoteur nous modifions sa durée, la durée est de type **uint**. Les plages de durée sont différentes pour chaque moteur, donc mes constantes ne sont utilisables que pour mon moteur. Pour les autres moteurs il faut les trouver soit même.

Cette classe ne nécessite pas que l’on utilise un type de moteur précis, que l’on utilise un moteur pas à pas ou a mouvement continu importe peu. Cependant il faudra réécrire les méthodes pour qu’elles correspondent à ce que l’on souhaite.

### Card

Cet objet est la représentation numérique des badges RFID. Il possède 2 attributs :

* Name (string) 🡪 le nom du badge
* Uid (string) 🡪 l’uid du badge

Ces objets permettent de différencier les badges dans le programme.

### ListOfCards

La classe ListOfCards contient les objets de type Card et permet de les gérer. Elle utilise le design pattern singletons. Elle contient 4 méthodes, une qui permet d’ajouter un nouvel objet de type Card, d’en supprimer un, de savoir si un badge RFID existe déjà et de savoir s’il y a des badges enregistrés ou pas.

La variable **\_cardsList**, de type **ArrayList**, contient tous les badges sauvegardés dans le programme. Cette classe possède la propriété **[Serializable]** qui permet de sérialiser la classe ou ses attributs dans des fichier afin de les sauvegarder.

Cette classe est la seule à pouvoir créer des nouveaux objets de type Card et d’en supprimer.

### SDCard

La classe SDCard permet de gérer le module SDCard de GHI. Elle utilise le design pattern singletons. Elle contient 2 méthodes, une sérialiser un objet de type ArrayList en byte[] afin de l’écrire dans un fichier stocker sur une carte sd et une qui permet de charger un fichier de byte[] et de le désérialiser en objet de type ArrayList.

Le module SDCard de GHI est instancier par cette classe, si l’on veut changer pour le module microSD de GHI il suffit juste de remplacer le type de module ainsi que le socket qu’il utilise sur la carte dans la classe.

La classe contient une constante de type **string 🡪 FILE\_NAME**, il s’agit ici du nom du fichier dans lequel nous allons écrire les données et que nous allons aussi charger pour récupérer les données.

## Machines d’états

### Scan d’un badge

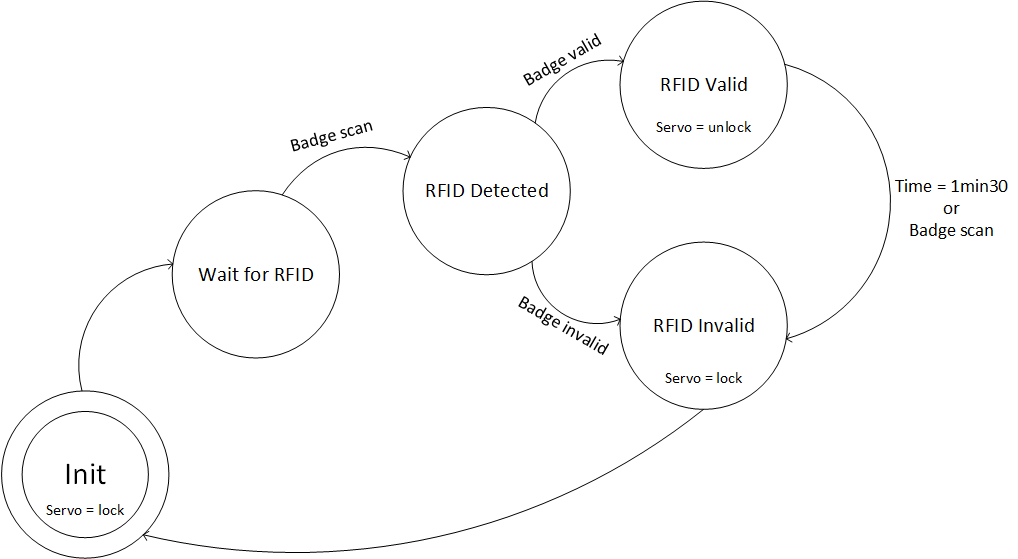


Figure 3 - Machine d'état : Scan d'un badge

### Ajout d’un badge

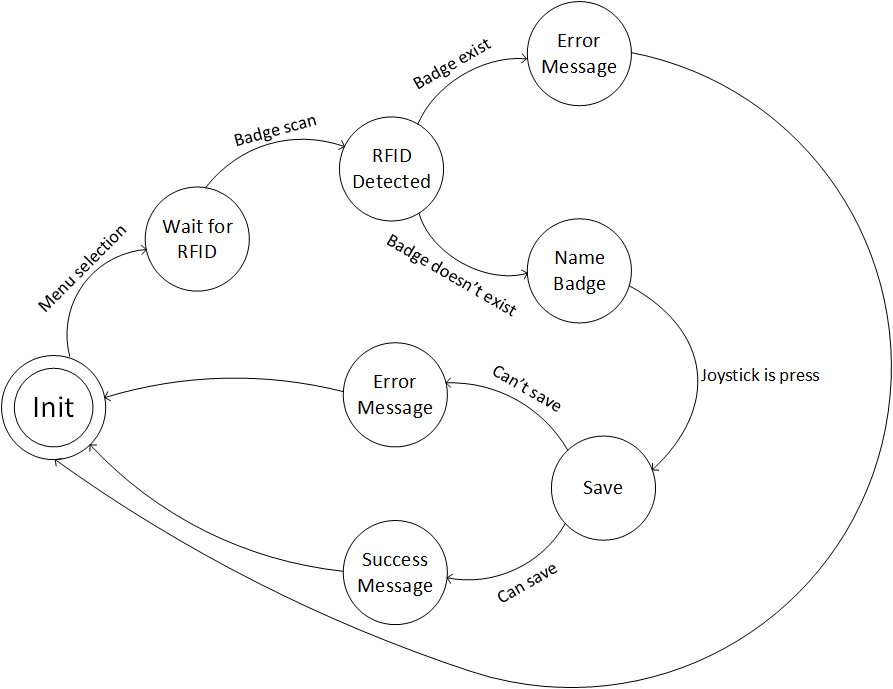


Figure 4 - Machine d'état : Ajout d'un badge

### Suppression d’un badge

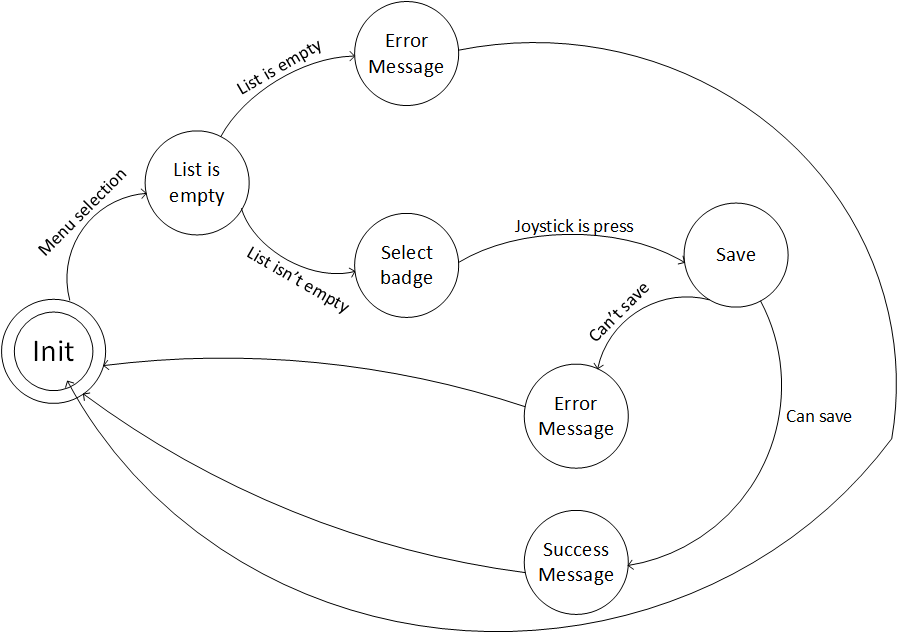


Figure 5 - Machine d'état : Suppression d'un badge

### Dévérrouiller avec le code secret

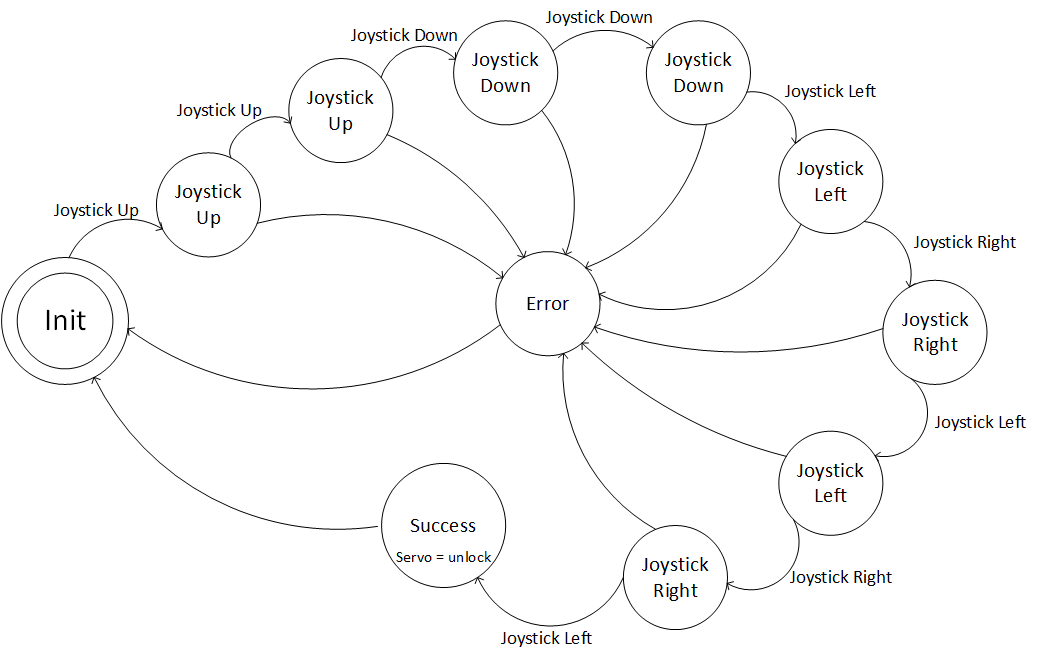


Figure 6 - Machine d'état : Déverrouiller avec le code secret

# Tests

## Test Unitaires

## Test Use Case

# Planning

# Conclusion

1. Lien du Git : <https://github.com/jeandanielkuenzi/RFIDPiggyBank> [↑](#footnote-ref-1)
2. Lien du Drive : <https://drive.google.com/drive/folders/1W5imih7jNj85cT0PoX8YY9FAt0lW8b60> [↑](#footnote-ref-2)