DOCUMENTATION TECHNIQUE

RFID PIGGY BANK

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

CFPT-I | 04.2018

TABLE DES MATIERES

1 TABLEAU DES REVISIONS

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 globale 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
1.5	Ajout tests use cases + planning + conclusion	23.05.2018
1.6	Mise en forme générale + ajout test unitaire	24.05.2018

2 INTRODUCTION

Cette documentation a pour but de détailler le fonctionnement de l'application microcontrôleur 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 acquises durant ma formation.

3 POURQUOI J'AI CHOISI CE PROJET

3.1 POURQUOI UN MICROCONTROLEUR 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.

3.2 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.

4 RAPPEL DU CAHIER DES CHARGES

4.1 ORGANISATION

Elève :
Jean-Daniel Küenzi
jean-daniel.knz@eduge.ch

Maitre d'apprentissage :
Christophe Maréchal
christophe.marechal@edu.ge.ch

Experts :
Nicolas Terrond
nicolas.terrond@sig-ge.ch

Jean-Pierre Garnier
jpg@gsinfo.ch

4.2 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).

4.3 SPECIFICATIONS

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
- Voir la liste des badges enregistrés
- Déverrouiller la boite via un code secret[1] donné (code secret non modifiable)
- Verrouiller la boite automatiquement en cas d'oubli (sécurité)

4.4 RESTRICTIONS

Le microcontrôleur n'est pas capable de :

- Renommer les badges
- Modifier le code secret pour déverrouiller
- Utiliser la fonction tactile de l'écran

4.5 ENVIRONNEMENT

Matériel nécessaire au développement de l'application :

- 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[2], Drive[3], Local

[1] Séquence code secret : ↑↑↓↓←→←→
[2] Lien du Git : https://github.com/jeandanielkuenzi/RFIDPiggyBank
[3] Lien du Drive : https://drive.google.com/drive/folders/1W5imih7jNj85cT0PoX8YY9FAt0lW8b60

4.6 LIVRABLES

- Documentation technique
- Manuel utilisateur
- Planning prévu
- Planning effectif
- Diagramme de classes
- Code source
- Microcontrôleur (Application)

5 ANALYSE FONCTIONNELLE

5.1 FONCTIONNALITES

5.1.1 DEVERROUILLER / 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).

5.1.2 AJOUTER UN BADGE

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

5.1.2.1 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 !**

5.1.3 SUPPRIMER UN BADGE

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

5.1.4 DEVERROUILLER 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.

5.1.5 VERROUILLAGE AUTOMATIQUE

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

5.2 CAS D'UTILISATIONS

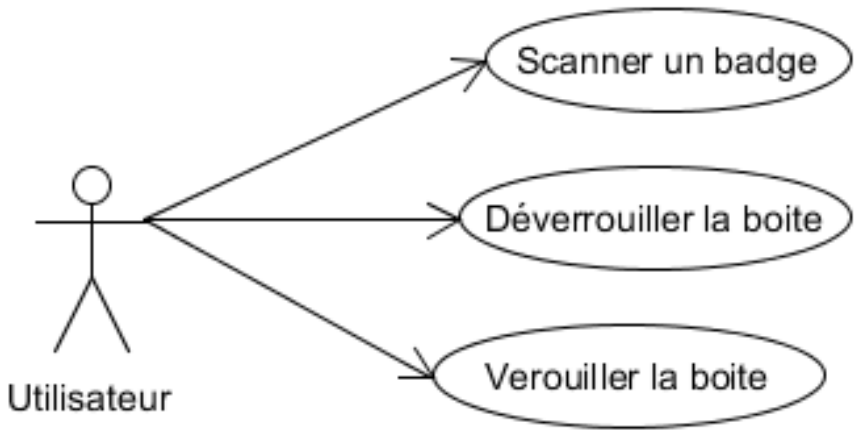


Figure 1 – Diagramme des cas d'utilisations

Nom : Déverrouiller la boite
Acteur principal : Utilisateur
Pré-requis : La boite est verrouillée
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 déverrouille la boite

Flot alternatif :

2a) Le badge n'est pas valide

2a1) Le système verrouille la boite

Nom : Verrouiller la boite
Acteur principal : Utilisateur
Pré-requis : La boite est déverrouillée
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 vérifie le badge
4) Le badge est valide (pas déjà enregistré)
5) Le système affiche un message de confirmation puis, le nom du badge
6) L'utilisateur valide le nom
7) Le système ajoute le badge et affiche le menu principal

Flot alternatif :

4a) Le badge n'est pas valide (déjà enregistré)

4a1) Le système affiche un message d'erreur pour informer l'utilisateur
4a2) Le système redirige l'utilisateur sur le menu principal

6a) L'utilisateur décide de nommer différemment son badge

4a1) L'utilisateur entre le nom qu'il souhaite 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 enregistré
2) L'utilisateur sélectionne un badge et clique sur « supprimer »
3) Le système supprime le badge et affiche le menu principal

Flot alternatif :

1a) La liste de badge enregistré est vide (aucun badge enregistré)

1a1) Le système affiche un message d'erreur pour informer l'utilisateur
1a2) Le système redirige l'utilisateur sur 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 » (bouton du joystick)

2b1) Le système annule et affiche le menu principal

6 ANALYSE ORGANIQUE

6.1 CONNECTIQUE

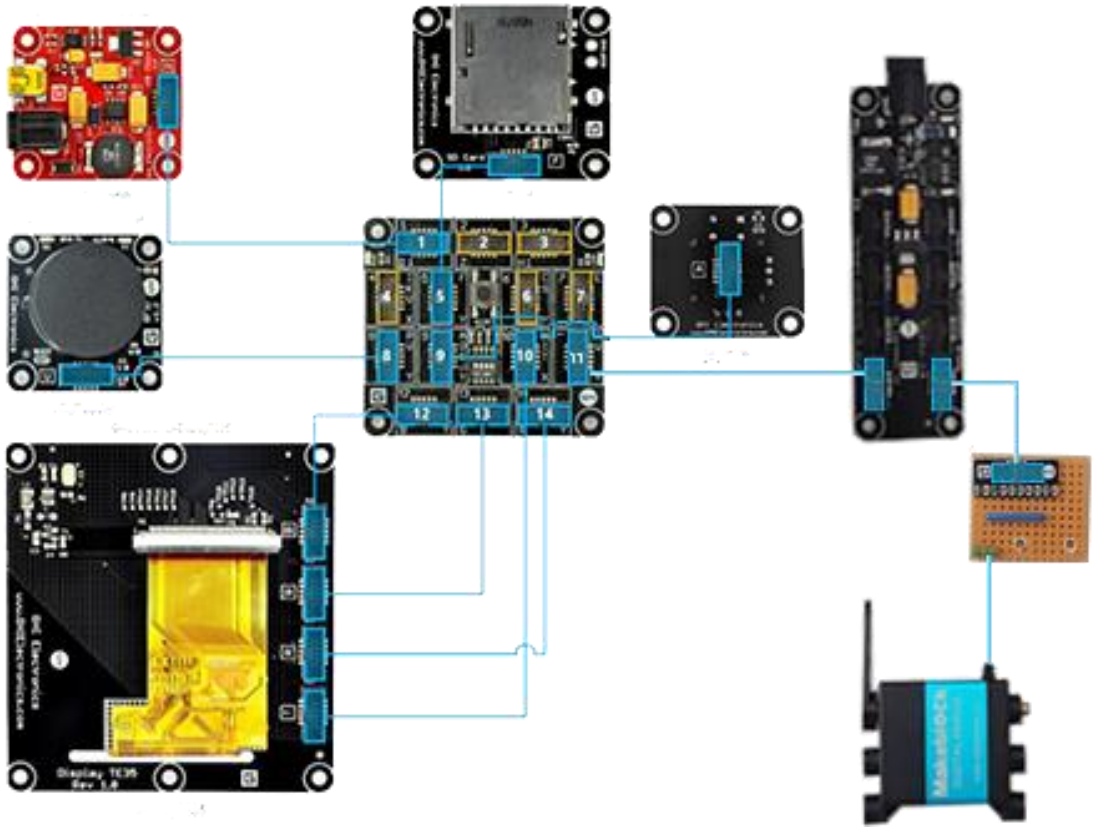


Figure 2 - Schéma connectique

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

6.2 MACHINES D'ETATS

6.2.1 SCANNER UN BADGE

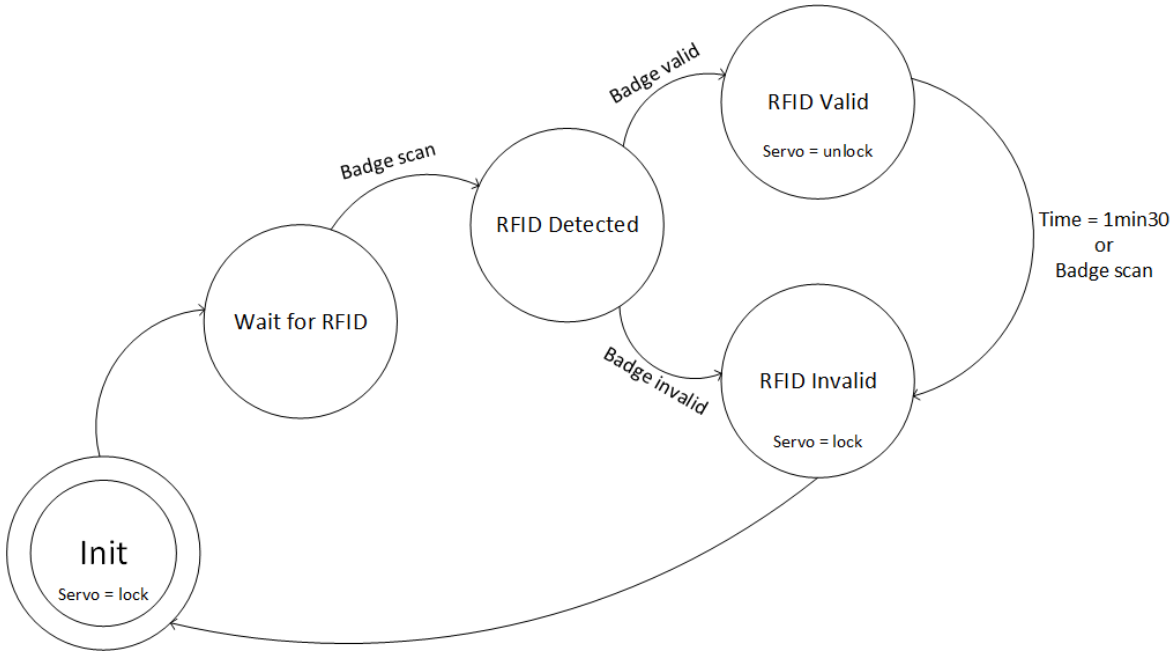


Figure 3 - Machine d'états : Scanner un badge

6.2.2 AJOUTER UN BADGE

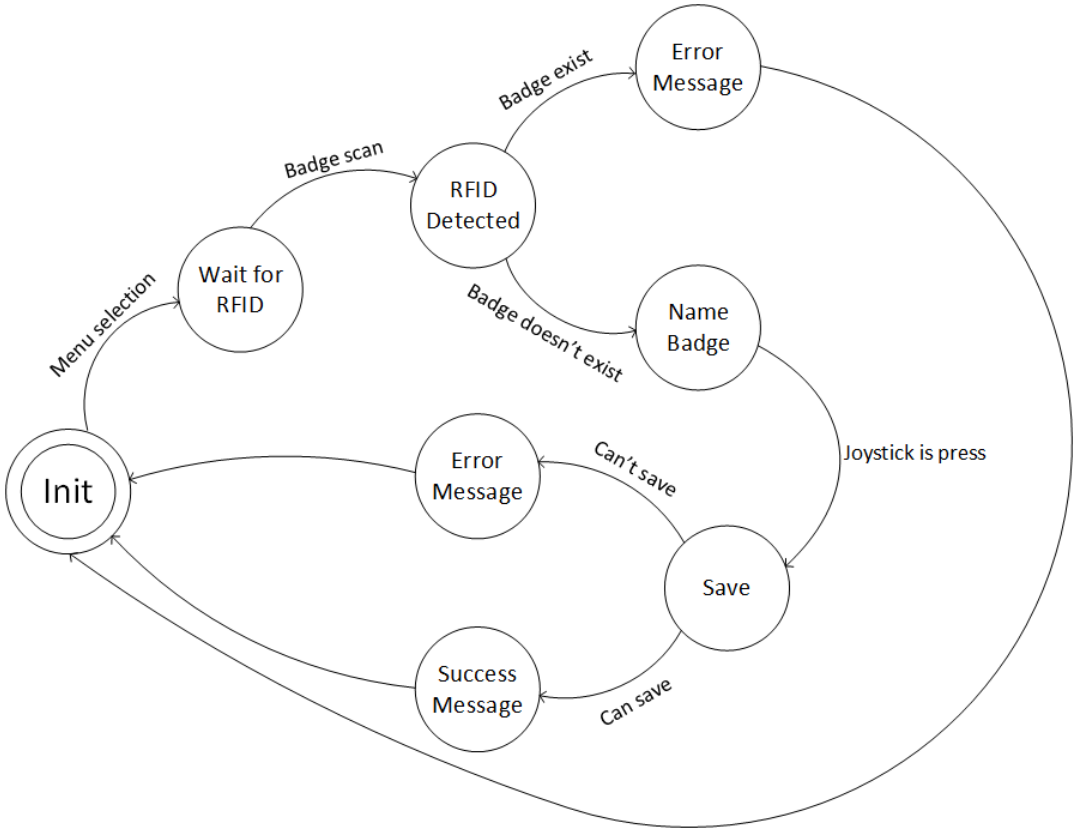


Figure 4 - Machine d'états : Ajouter un badge

6.2.3 SUPPRIMER UN BADGE

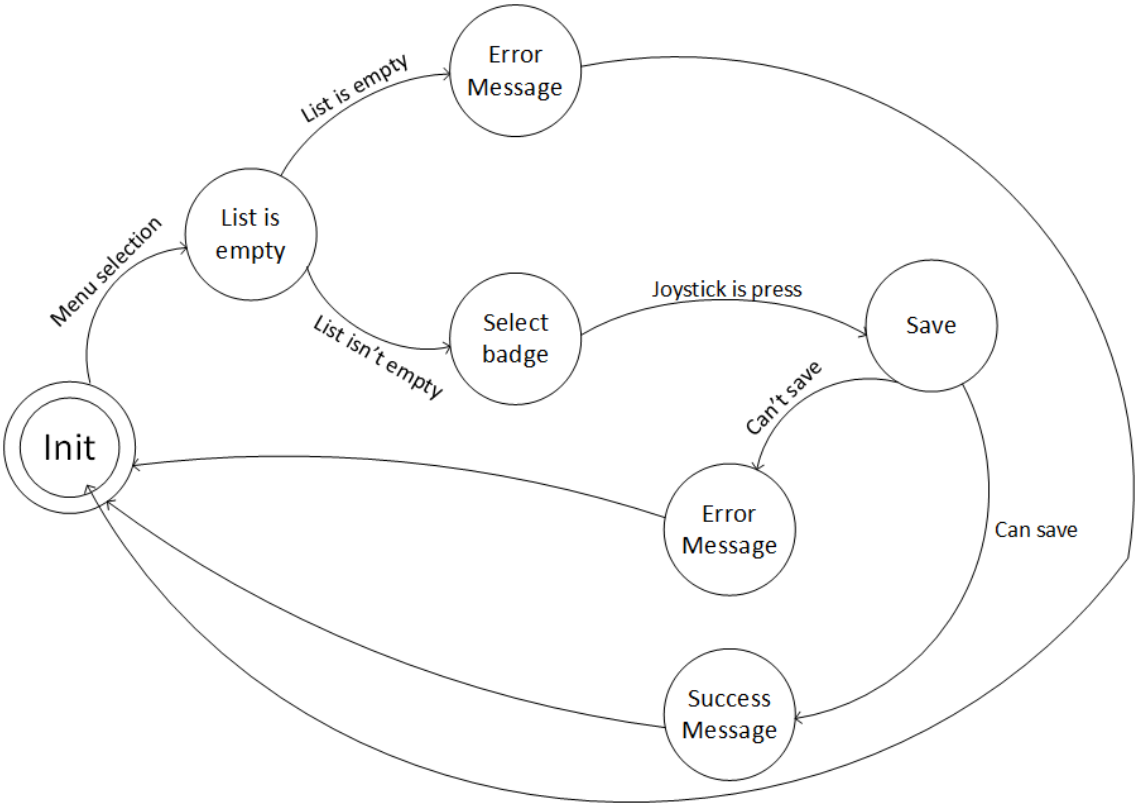


Figure 5 - Machine d'états : Supprimer un badge

6.2.4 DEVERROUILLER AVEC LE CODE SECRET

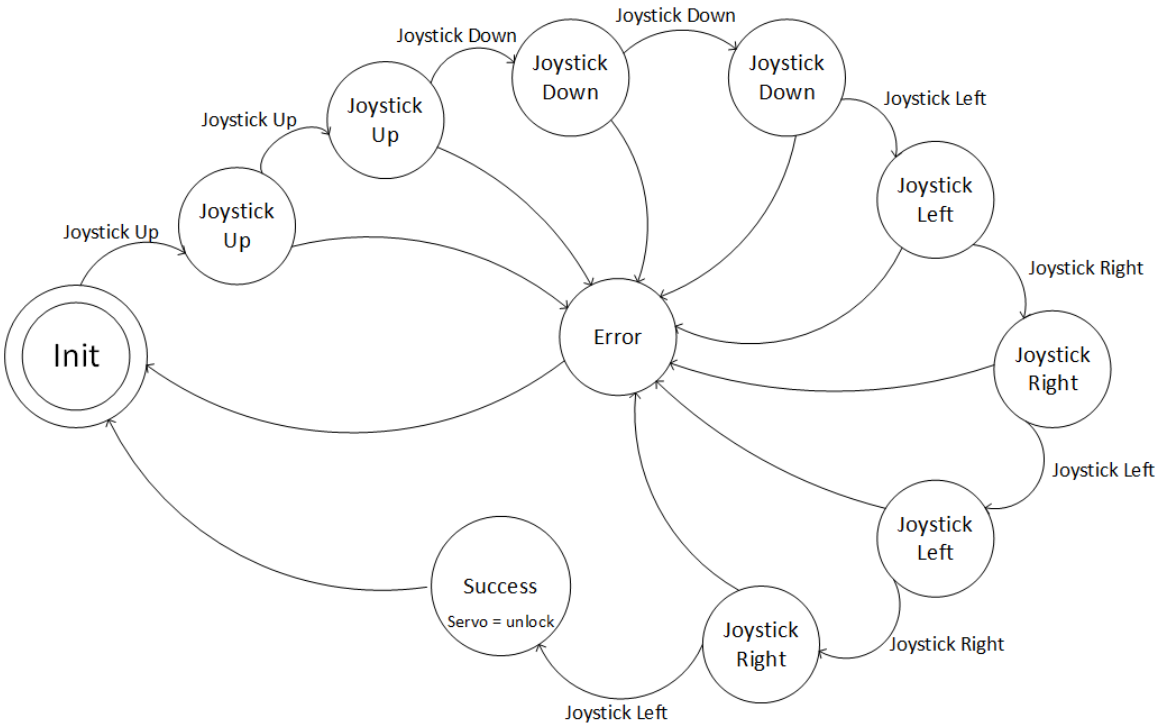


Figure 6 - Machine d'états : Déverrouiller avec le code secret

6.3 DIAGRAMME DE CLASSES

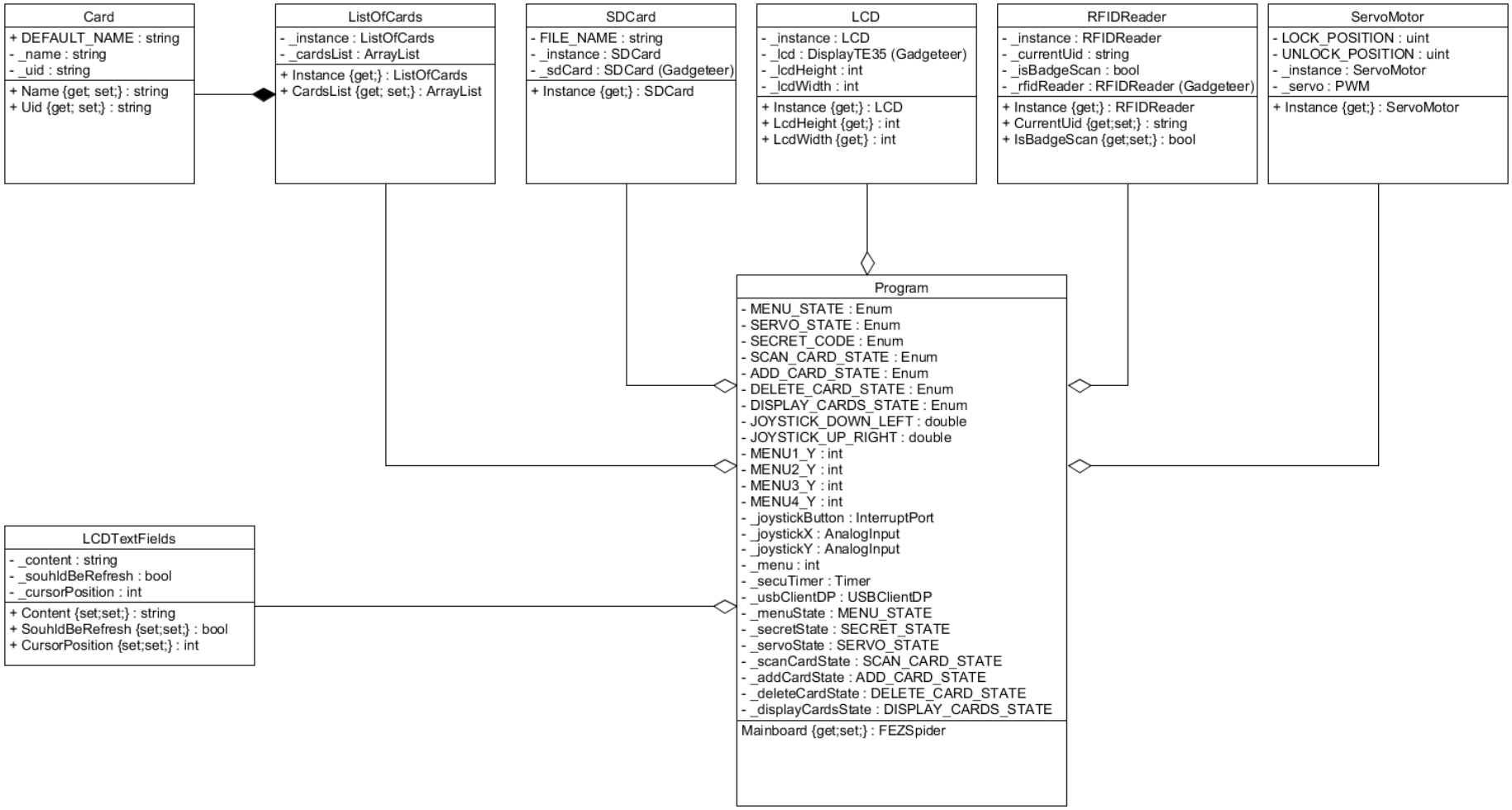


Figure 7 - Diagramme de classes

6.4 VUE

6.4.1 PROGRAM

La classe Program est en quelque sorte ma vue, c'est là que toute la logique et les appels aux modèles vont être effectués. Elle n'instancie pas directement les autres classes car ce sont des singletons à l'exception de la classe LCDTextFields qui elle est static.

Tous les composants qui y sont branchés et initialisés sont définis 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 donnée sur la carte. Ce ne sera donc pas les mêmes valeurs pour la position 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.

6.5 MODELE

6.5.1 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.

6.5.2 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.

6.5.3 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) le servomoteur.

Le servomoteur est une variable de type **PWM** (modulation de largeur d'impulsion), on peut donc en modifier sa période ainsi que sa durée. Le servomoteur 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.

6.5.4 CARD

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

- Name (string) → le nom du badge
- Uid (string) → l'uid du badge

Ces objets permettent de différencier les badges dans le programme.

6.5.5 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 fichiers afin de les sauvegarder.

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

6.5.6 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.

De base la classe est sensé sérialiser les données que nous lui passons en XML. Mais la sérialisation est très compliquée en microcontrôleur avec le Micro Net Framework 4.3. Il n'existe pas déjà de méthode ou de librairies pour pouvoir sérialiser un objet en xml. Pour pouvoir sérialiser en xml je devrai créer ma propre méthode de sérialisation ce qui est très compliqué et demande beaucoup de temps. J'ai alors essayé de sérialiser mes données en json, le format change mais le principe reste le même, on sauvegarde les badges dans un fichier. Mais pour le json c'est pareil que pour l'xml, aucune méthode de sérialisation. J'ai trouvé une librairie sur le site de ghi (old.ghielectronics.com) qui permet de sérialiser un objet en un objet primitifs json. J'ai pensé qu'en réécrivant la classe je pourrai peut-être écrire cet objet primitif dans un fichier et le charger par la suite mais ça s'est avéré plus compliqué que prévu.

C'est pour ça que j'ai décidé de sérialiser mes données en tableau de bytes, une méthode ainsi qu'une librairie sont déjà incluse dans le Micro Net Framework 4.3 (Assembly : System.Refelction). Mes données sont donc stockées dans un fichier xml mais sous forme de bytes, ce qui me permet de les sérialiser et désérialiser très facilement.

6.5.7 LCDTEXTFIELDS

La classe LCDTextFields est une classe static. Elle possède 3 propriétés :

- Content (string) → Valeur du champ sur le LCD
- ShouldBeRefresh (bool) → Savoir si le LCD doit être rafraichit
- CursorPosition (int) → La position du curseur sur le LCD (Champs, etc…)

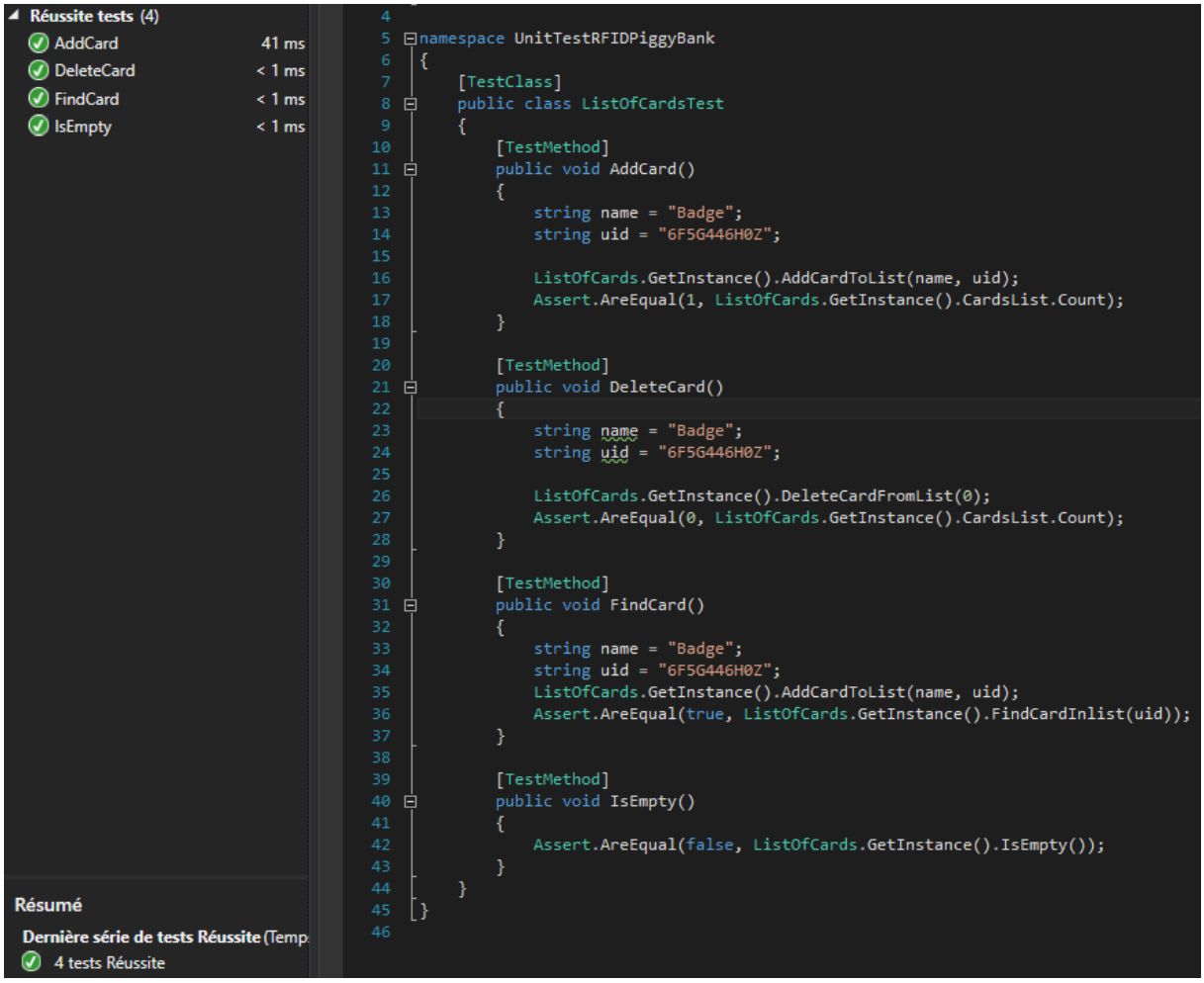
J'utilise cette classe au moment où je nomme mon badge, elle me permet d'afficher le nom du badge comme étant un chant et de pouvoir le modifier. Mon programme étant une machine d'état, si j'initialisais la valeur du champ au moment du nommage elle serait écrasée à chaque fois, c'est pour ça que je l'ai sortie et j'ai créé une classe static.

7 TESTS

7.1 TEST UNITAIRES

7.1.1 LISTOFCARDS

Tests réalisé le : **23.05.2018**




```
namespace UnitTestRFIDPiggyBank
{
    [TestClass]
    public class ListOfCardsTest
    {
        [TestMethod]
        public void AddCard()
        {
            string name = "Badge";
            string uid = "6F5G446H0Z";

            ListOfCards.GetInstance().AddCardToList(name, uid);
            Assert.AreEqual(1, ListOfCards.GetInstance().CardsList.Count);
        }

        [TestMethod]
        public void DeleteCard()
        {
            string name = "Badge";
            string uid = "6F5G446H0Z";

            ListOfCards.GetInstance().DeleteCardFromList(0);
            Assert.AreEqual(0, ListOfCards.GetInstance().CardsList.Count);
        }

        [TestMethod]
        public void FindCard()
        {
            string name = "Badge";
            string uid = "6F5G446H0Z";
            ListOfCards.GetInstance().AddCardToList(name, uid);
            Assert.AreEqual(true, ListOfCards.GetInstance().FindCardInlist(uid));
        }

        [TestMethod]
        public void IsEmpty()
        {
            Assert.AreEqual(false, ListOfCards.GetInstance().IsEmpty());
        }
    }
}
```

7.2 TESTS CAS D'UTILISATIONS

7.2.1 TABLEAU DESCRIPTIF DES TESTS

N°	Description	Date prévue
1	Navigation dans le menu principal	24.05.2018
2	Scan d'un badge valide alors que la boite est verrouillée	24.05.2018
3	Scan d'un badge invalide alors que la boite est verrouillée	24.05.2018
4	Scan d'un badge valide et invalide alors que la boite est déverrouillée	24.05.2018
5	Ajout d'un badge avec le nom par défaut	24.05.2018
6	Ajout d'un badge avec renommage du badge	24.05.2018
7	Ajout d'un badge déjà existant	24.05.2018

8	Suppression d'un badge	24.05.2018
9	Suppression d'un badge alors que la liste des badges est vide	24.05.2018
10	Afficher la liste des badges enregistrés	24.05.2018
11	Afficher la liste des badges enregistrés alors qu'elle est vide	24.05.2018
12	Déverrouiller avec le code secret	24.05.2018
13	Verrouillage automatique de sécurité	24.05.2018

7.2.2 TABLEAU DE RESULTATS DES TESTS

Test réalisé par : **Ronaldo Loureiro Pinto**

N°	Résultat attendu	Etat	Commentaire
1	L'utilisateur peux naviguer dans le menu et un curseur indique sa position	OK	-
2	La boite se déverrouille	OK	-
3	La boite reste verrouillée	OK	-
4	La boite se verrouille	OK	-
5	Le badge s'ajoute et rajoute un numéro à la fin du nom. Un message de confirmation apparait	OK	-
6	Le badge s'ajoute avec le nom donné. Un message de confirmation apparait	OK	-
7	Un message d'erreur apparait au moment du scan pour prévenir l'utilisateur	OK	-
8	Le badge est supprimé. Un message de confirmation apparait	OK	-
9	Un message d'erreur préviens l'utilisateur qu'il n'y a pas de badges enregistrés	OK	-
10	Les badges enregistrés s'affichent en colonnes	OK	-
11	Un message d'erreur préviens l'utilisateur qu'il n'y a pas de badges enregistrés	OK	-
12	La boîte se déverrouille si le code secret est juste	OK	Parfois les commandes du joystick ne sont pas bien lues
13	La boîte se verrouille toutes seul après 1min30 d'ouverture	OK	-

8 PLANNING

8.1 PLANNING PREVISIONNEL

gantter™

Project Name **TPI_Planning_Previsionnel_Kuenzi**

	Nom	Durée	Début	Fin	Prédécesseurs	Ressources
1	Immersion et analyse globale	0.5journée	05/08/2018	05/08/2018		Jean-Daniel
2	Inventaire des fonctionnalités et diagrammes de classes	1.5journées	05/08/2018	05/09/2018	1	Jean-Daniel
3	Développement des classes	0.5journée	05/11/2018	05/11/2018	2	Jean-Daniel
4	Gestion de la connexion avec le servomoteur	4journées	05/11/2018	05/18/2018	3	Jean-Daniel
5	Développement du menu LCD (Interface + fonctionnalités)	2journées	05/18/2018	05/23/2018	4	Jean-Daniel
6	Test et correction des bug	1.5journées	05/23/2018	05/24/2018	5	Jean-Daniel
7	Documentation technique, descriptif de la solution technique	10journées	05/08/2018	05/24/2018		Jean-Daniel
8	Documentation fonctionnelle, descriptif des fonctionnalités dévelo...	10journées	05/08/2018	05/24/2018		Jean-Daniel
9	Tenue du journal de bord	10journées	05/08/2018	05/24/2018		Jean-Daniel

PNG Generated On: 23/05/2018 à 22:13:39

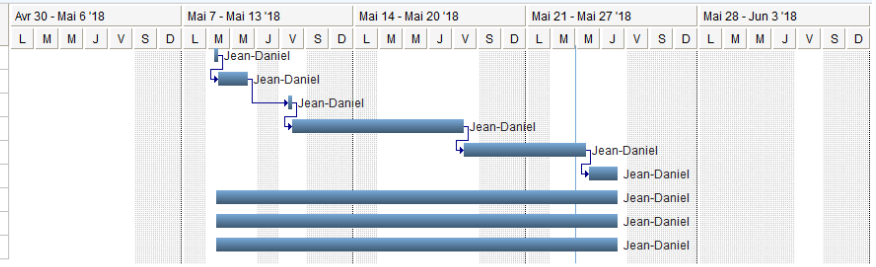


Figure 8 - Planning prévisionnel

8.2 PLANNING EFFECTIF

gantter™

Project Name **TPI_Planning_Effectif_Kuenzi**

	Nom	Durée	Début	Fin	Prédécesseurs	Ressources
1	Immersion et analyse globale	0.5journée	05/08/2018	05/08/2018		Jean-Daniel
2	Inventaire des fonctionnalités et diagrammes de classes	0.5journée	05/08/2018	05/08/2018	1	Jean-Daniel
3	Développement des classes	3journées	05/09/2018	05/15/2018	2	Jean-Daniel
4	Analyse pour la sérialisation (Classe SDCard)	3journées	05/09/2018	05/15/2018	2	Jean-Daniel
5	Gestion de la connexion avec le servomoteur	1journée	05/15/2018	05/16/2018	3,4	Jean-Daniel
6	Développement du menu LCD (Interface + fonctionnalités)	3journées	05/16/2018	05/22/2018	5	Jean-Daniel
7	Factorisation du code (Machine d'états, améliorations des classes, et...	1.5journées	05/22/2018	05/23/2018		
8	Test et correction des bug	1.5journées	05/23/2018	05/24/2018	6	Jean-Daniel
9	Documentation technique, descriptif de la solution technique	3journées	05/08/2018	05/11/2018		Jean-Daniel
10	Documentation technique, descriptif de la solution technique	5journées	05/17/2018	05/24/2018	9	Jean-Daniel
11	Documentation fonctionnelle, descriptif des fonctionnalités développé...	2journées	05/23/2018	05/24/2018		Jean-Daniel
12	Tenue du journal de bord	10journées	05/08/2018	05/24/2018		Jean-Daniel

PNG Generated On: 23/05/2018 à 22:15:11

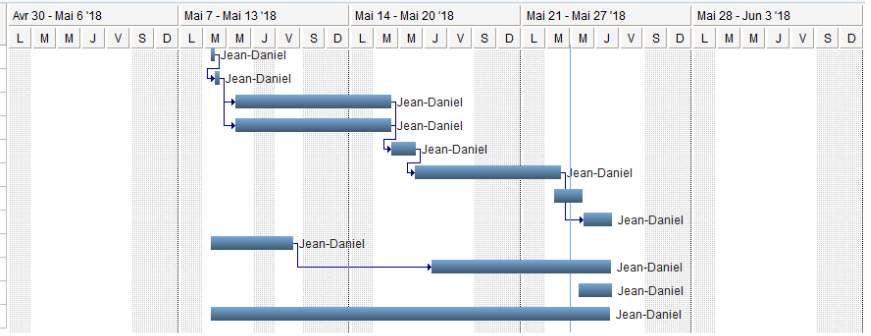


Figure 9 - Planning effectif

8.3 DIFFERENCE ENTRE LES PLANNINGS

On peut remarquer une différence majeure entre les deux plannings.

Premièrement, on peut voir que j'avais prévu ½ journée pour développer les classes mais au final ça m'en a pris 3 journées. Cela s'explique avec le problème que j'ai rencontré pour la classe SDCard, c'est essentiellement elle que j'ai développée pendant ces 3 journées. Comme expliqué dans la partie analyse organique, de base je pensais utiliser la sérialisation en xml pour sauvegarder mes badges sur la carte SD. Mais la sérialisation xml n'étant pas déjà développé pour le Micro Framework 4.3 j'ai dans un premier temps essayé de créer ma propre méthode de sérialisation qui s'est voué être un échec. J'ai ensuite cherché des solutions, d'où la branche d'analyse qui arrive en parallèle (planning effectif). Finalement j'ai trouvé une solution, sauvegarder en byte mes données à la place du xml.

Deuxièmement, sur le planning prévisionnel j'avais prévu 4 journées pour développer la connexion au servomoteur, je pensais que ça allait être la partie la plus compliquée à gérer. Au final il s'est avéré que c'est la gestion du LCD et ce problème de sérialisation qui m'ont pris le plus de temps.

Troisièmement, on peut remarquer que ma documentation n'est pas constante. Au départ je pensais développer le manuel utilisateur et la documentation technique en même temps que j'avançais le projet mais on peut constater que ma documentation technique a été stoppée pendant 2 jours. J'étais tellement pris dans ma recherche pour trouver une solution à mon problème de sérialisation que je n'ai pas consacré de temps à la documentation. Ensuite le gros changement est pour le manuel utilisateur que j'ai finalement développé que les 2 derniers jours du TPI. Je me suis rendu compte pendant le TPI que c'était bizarre de faire un manuel utilisateur sur un produit qui n'était pas fini et qui allait surement évoluer entre le début et la fin de mon TPI. J'ai donc préféré attendre que mon application soit stable pour commencer le manuel utilisateur.

Ces trois points sont les différences majeures entre mon planning prévisionnel et mon planning effectif.

9 CONCLUSION

9.1 BILAN PERSONNEL

Je suis satisfait de mon travail, ces 3 semaines de TPI (80 heures) ont été très intenses et m'ont permis de plus me retrouver dans une situation proche du professionnel. Aussi pour moi ce TPI était une sorte de défi, je voulais me lancer dans quelque chose de nouveau et d'original. Mon programme fonctionne plutôt bien et je suis content d'en être arrivé à bout. Le TPI aura été aussi une occasion d'être complètement autonome au niveau du temps et de la gestion de notre projet. Cela change beaucoup du quotidien de la vie d'un étudiant. Ce projet professionnel fait l'inventaire des capacités que j'ai obtenues lors de ses 4 ans au CFPT-I.

9.2 AMELIORATIONS POSSIBLES

Pour ce projet il y a plusieurs améliorations auxquelles j'ai pensé mais je n'ai pas eu assez de temps pour les réaliser.

9.2.1 ASPECT SECURITE ET ERGONOMIE :

Premièrement, un système de badge admin, qu'on ne peut pas supprimer, pour enregistrer ou supprimer un badge. Effectivement pour l'instant on peut supprimer n'importe quelle carte enregistrée sans devoir confirmer le choix mais surtout on peut enregistrer n'importe quelle carte RFID. Donc il suffit juste de scanner un badge et de l'enregistrer pour pouvoir ouvrir la boîte. Il y a donc un problème au niveau de la sécurité.

Deuxièmement, un moyen de retourner en arrière, par exemple on a scanné le mauvais badge on retourne en arrière pour en scanner un nouveau, où on a mal cliqué sur un menu il peut annuler en revenant en arrière. Dans la version actuelle de l'application, si l'on clique accidentellement sur ajouter un badge ou supprimer un badge il n'y a pas de retour possible.

Finalement, lorsque l'on exécute le code secret, sur le LCD j'affiche la progression du mot de passe. Le problème étant que le joystick n'est pas toujours bien lu par le programme, si on n'affiche pas un indicateur pour dire que l'entrée a bien été lu nous ne savons pas dans quel « état » nous sommes pour le mot de passe. De ce fait il est facilement possible de deviner le mot de passe, il faudra améliorer la fonctionnalité du mot de passe pour la rendre plus fluides et affiché autre chose que la progression du mot de passe.

9.2.2 SUPPRESSION D'UN BADGE

Pour la suppression on pourrait ajouter une étape de confirmation du choix ainsi que le choix multiple possible. Pour l'instant on ne peut supprimer qu'un badge à la fois, imaginons qu'on a beaucoup de badge à supprimer ça va prendre du temps alors que si l'on peut tous les supprimer en une fois c'est plus rapide et plus ergonomique.

9.2.3 AJOUT D'UN BADGE

Pour l'instant, lorsque l'on ajoute un badge, j'autorise tous les caractères de la table. Je pourrai rajouter un filtre pour n'autoriser que certains caractères. Aussi pour l'instant il n'est possible que de faire des noms de 5 caractères. On pourrait rajouter le fait de pouvoir enlever des caractères et d'en ajouter pour permettre une plus grande flexibilité au niveau des noms.

Aussi lorsque l'on essaye d'ajouter un badge qui existe déjà le programme nous informe que ce badge est déjà enregistré mais il ne nous dit pas lequel c'est. Il pourrait être intéressant que le programme nous dise aussi le nom du badge pour que l'on puisse le retrouver où le supprimer.

9.2.4 CODE SECRET

Pour l'instant le code secret est imposé et non modifiable. On pourrait rajouter le fait que le code secret soit modifiable par l'utilisateur et même rajouter un système de backup pour le code secret si on perd nos badges et qu'on oublie notre code secret.

10 BIBLIOGRAPHIE

10.1 SITES UTILISES

- https://openclassrooms.com
- https://stackoverflow.com
- https://old.ghielectronics.com

10.2 AIDE REÇUE

Professeur :

- **M. Maréchal** : Suivi de la documentation et conseils pour la partie programmation
- **M. Wanner** : Aide pour l'installation de l'environnement de développement et pour le problème de sérialisation XML

Elèves :

- **Dario Genga** : Conseils sur la structure du code
- **Ronaldo Loureiro Pinto** : Testeur de mon application et conseils pour la documentation

11 TABLE DES FIGURES

12 ANNEXES

- Code source
- Planning prévisionnel
- Planning effectif
- Manuel utilisateur
- Journal de bord manuscrit