RAPPORT FINAL

PAR
CARLOS SANCHEZ

BAC EN INFORMATIQUE
FACULTÉ DES ARTS ET DES SCIENCES

TRAVAIL RÉALISÉ SOUS LA SUPERVISION DE
M. SÉBASTIEN ROY ET MME.SYLVIE HAMEL
DANS LE CADRE DU COURS IFT3150
PROJET D'INFORMATIQUE

AOÛT 2017

1. Présentation

1.1 Introduction

Ce document vise à présenter toutes les étapes de la conception d'un système de paiement automatisé de café réalisé dans le cadre du cour cours IFT3150 - Projet d'informatique. Ce rapport de projet fera part des objectifs visés par ce système, son fonctionnement, son implémentation puis je terminerai avec un compte rendu du processus de développement.

1.2 Objectifs

L'idée du projet m'a été proposé par mon superviseur, Sébastien Roy, lors de notre première rencontre pour discuter d'un projet à réaliser pour la session d'été 2017. Il s'agit d'un système de café libre service à paiement électronique automatique. Le paiement se fait à l'aide d'une puce électronique contenant un solde, collée à une tasse que l'on dépose sur une base contenant le lecteur et une balance. La balance permet de mesurer le volume du café acheté et de payer selon le volume que l'on utilise. L'objectif de ce projet est de remplacer le système de paiement d'honneur du café étudiant du département d'informatique dans lequel on paie 50 cents dans une boite pour remplir une tasse de volume arbitraire. Il permettrait de réduire le gaspillage en dissuadant les utilisateurs de remplir de tasses à grand volume et de d'offrir un meilleur rapport qualité/prix pour les utilisateurs qui boivent de plus petites quantités de café.

1.3 Matériel

Le logiciel est implémenté à l'aide de la plateforme Arduino. C'est une plateforme de prototypage open-source permettant de faire l'interface entre des senseurs, lecteurs et autre matériel électronique en C++. Les cartes NFC sont lues par le module MFRC522, le poids est mesuré par un load-cell de 10kg dont le signal électrique est géré par l'amplificateur HX711. Pour le retour visuel, on utilise une barre de 8 DEL de type NeoPixel ainsi qu'un écran OLED de 128x64 pixels. La structure sur laquelle est attaché toutes le composantes a été conçue et imprimée en 3D par Sébastien.

1.4 Cas d'utilisation typique du système libre service

Une utilisation du système se déroule ainsi:

1. La barre de DEL est allumée et de couleur bleu et l'écran numérique indique que la machine est prête à être utilisée, en attente d'une tasse.
2. L'utilisateur dépose sa tasse sur le module, l'affichage numérique indique que le module se prépare et la barre s'allume et s'éteint progressivement en mauve, signalant que la balance se stabilise.
3. Le lecteur tente de lire les informations sur la carte:
    - Si on ne lit rien après 3 secondes, le système se remet en mode attente et on retourne à l'étape 1.
    - Si on lit une carte invalide, le système se remet en mode attente et on retourne à l'étape 1.
    - Si une carte valide a été lue, la barre de DEL clignote en vert puis on indique le solde, le prix du volume courant dans la tasse sur l'écran et l'utilisateur peut se servir.
4. Pendant que le café coule, une DEL jaune se déplace de gauche à droite, signalant que le niveau de café monte.
5. L'utilisateur arrête de se servir et la tasse se stabilise.

6. Une fois stable, on débite le montant correspondant au café servi de la carte et la barre de DEL clignote en vert signifiant que le client peut reprendre sa tasse.
7. Le système se remet en mode attente.

1.5 Cas d'utilisation du système de recharge

Pour ce logiciel, nous disposons du même matériel, mais sans le senseur de poids et l'amplificateur HX711. Contrairement au système libre service normal, celui-ci serait opéré par un employé du café, car il a accès à des cartes administrateur. Une utilisation se déroulerait ainsi:

1. La système est en mode attente avec les DEL allumées en bleu et on demande de lire une carte administrateur.
2. Un employé lit une carte d'un montant donné:
    - Si la carte lue n'est pas une carte administrateur valide, la barre de DEL s'allume en rouge, on fait part de l'invalidité de la carte sur l'écran et on retourne en mode attente.
    - Si la carte lue est valide, on attend la lecture d'une carte usager.
3. L'employé lit la carte usager:
    - Si la carte lue n'est pas une carte administrateur valide, la barre de DEL s'allume en rouge, on fait part de l'invalidité de la carte sur l'écran et on retourne en mode attente.
    - Si la carte lue est valide, on ajoute au solde courant le montant de la carte administrateur lue à l'étape 2.
    - Si on ne lit pas de carte après 6 secondes, la barre de DEL s'allume en rouge, on fait part de l'absence de carte sur l'écran et on retourne en mode attente.
4. Le système se remet en mode attente.

2. Implémentation

2.1 Fonctionnement du logiciel

La plateforme Arduino requiert que l'on fournisse à chaque script les méthodes `setup()` et `loop()`. La première méthode sert à l'initialisation des entrées/sorties et des variables du microcontrôleur. La seconde est une boucle infinie qui contient le reste du programme. Puisque cette structure est imposée par la plateforme, j'ai dû concevoir tout le projet autour de ces contraintes. Après l'initialisation, la méthode `loop()` fait l'appel successif de 3 blocs de méthodes. En premier lieu, on met à jour tout ce qui a trait à la balance, soit le poids à afficher et met à jour les variables pour la stabilisation de poids. Après la mise à jour des variables, on appelle la méthode `userLoop()` s'occupant de la gestion de la boucle d'utilisation. Cette méthode est une fonctions switch, évaluant à chaque appel une variable globale représentant l'état du système qui peut prendre une des valeurs suivantes: STANDBY, AUTHENTICATION, SERVING, PAYMENT ou DONE. Selon cette variable d'état et les valeurs rapportées par l'objet `Scale`, à chaque appel de `userLoop()` le système saura à quelle étape de la séquence d'utilisation nous sommes rendus. Après cet appel, on fait appel aux fonctions de gestion de l'écran OLED qui mettent à jour les valeurs à afficher pour l'utilisateur.

Le script de recharge de carte possède une structure semblable. La différence est que nous effectuons deux lectures de cartes distinctes pour avancer dans les étapes, la carte administrateur en premier lieu pour déclencher l'étape d'authentification, puis la carte de l'utilisateur par la suite pour ajouter le solde.

2.2 Système de stabilisation

Une partie critique du logiciel est le système de stabilisation. J'ai créé une librairie Scale pour le senseur de poids et son amplificateur qui calcule la stabilité à l'aide de la différence de poids en fonction du temps. À chaque mise à jour des variables dans la boucle `loop()`, on fait une comparaison avec le poids mesuré précédemment. Si nous constatons qu'il y a une différence de plus d'un gramme, on met un flag de stabilisation à `false`. Si après 3 secondes le poids n'a pas varié de plus d'un gramme, on met le flag de stabilité à `true` puis on garde en mémoire le dernier poids stable.

2.3 Carte MIFARE Ultralight et lecteur MFRC522

Les cartes NFC utilisées dans le cadre de ce projet sont de type Mifare Ultralight. Malgré la compatibilité partielle de ce format de carte Mifare avec le lecteur MFRC522, j'ai choisi cette famille de carte car elles ont été plus accessibles en termes de disponibilité (sous forme de carte OPUS jetable) lors de la réalisation du projet, de leur structure mémoire simple et pour la facilité avec laquelle on peut écrire dans la mémoire. Ces cartes NFC sont dotées de 64 bytes de mémoire et divisées en pages de 4 bytes, pour un total de 16 pages:

Page	**Bytes**			
	0	**1**	**2**	**3**
0	Numéro de série (UID)			
1	Numéro de série (UID)			
2	UID	Internal	Lock bytes	Lock Bytes
3	OTP	OTP	OTP	OTP
4 à 15	Mémoire d'utilisateur			

Les lock bytes servent à déterminer les permissions d'écriture des pages 3 à 15. Les bytes OTP sont programmables qu'une seule fois et peuvent servir par exemple de compteur permanent pour limiter le nombre d'utilisations d'une carte.

La lecture et l'écriture de cartes MIFARE Ultralight n'étaient que partiellement supporté par le lecteur de carte MFRC522. J'ai dû faire quelques modifications à la librairie, notamment pour la lecture d'une plage de pages donnée. Ces modifications ont été rapides et sans problèmes car le code de la librairie était bien commenté et il y avait déjà quelques fonctions de lecture et d'écriture pour les cartes MIFARE Ultralight sur lesquelles je pouvais prendre exemple. Le processus d'écriture dans le format Ultralight se fait sur une page (4 bytes) à la fois et la lecture se fait en bloc de 4 pages.

2.4 Structure de donnée pour le projet

Pour les besoins du projet, j'ai réservé une plage de 8 pages (8 à 15) pour stocker les données d'une carte. J'ai choisi ces pages car les cartes avec lesquelles je travaillais durant la conception n'étaient pas restreinte au niveau des permissions d'écriture et il y a assez de mémoire dans 16 bytes pour stocker l'information nécessaire. Pour une carte usager, la mémoire est distribuée ainsi:

Page	**Bytes**			
	0	**1**	**2**	**3**
8	Dollars	Cents	Signe	Inutilisé
9	Numéro de série (UID)			
10	UID	UID	UID	Hasard
11-13	Hasard			
14	0xCA	0xFE	0xCA	0xFE
15	0xCA	0xFE	0xCA	0xFE

Pour valider une carte usager, nous avons des méthodes qui comparent le UID écrit dans la mémoire avec le UID permanent dans la carte. Cela nous assure uniquement que les données n'ont pas été copiées à partir d'une seconde carte. Une autre méthode vérifiant les deux dernières pages permettent de valider qu'il s'agit du format de carte usager plutôt qu'administrateur.

Les bytes de hasard pour le moment servent à rendre les données encryptées plus aléatoires. Ils pourraient éventuellement servir à stocker plus d'information comme un timestamp pour ajouter une couche de sécurité supplémentaire. Cet aspect sera discuté ultérieurement dans ce rapport.

Du côté administrateur, la structure de donnée utilisé est écrite aussi sur les pages 8 à 15. Elle est distribué ainsi:

Page	**Bytes**			
	0	**1**	**2**	**3**
8	0xCA	0xFE	0xCA	0xFE
9	Numéro de série (UID)			
10	UID	UID	UID	Hasard
11	Dollars	Hasard		
13-15	Hasard			

Comme pour les cartes utilisateurs, nous disposons du même type de méthodes de validation, une pour le UID en mémoire, et une autre pour valider le type de carte.

2.5 Cryptographie

Puisque ce projet possède un aspect transactionnel, il est impératif d'avoir une couche de sécurité afin de protéger les données sensibles sur la carte et d'en empêcher la copie. Pour cela, j'utilise la librairie ChaCha pour Arduino. C'est une variante de l'encryption Salsa20 qui est déjà bien connue en cryptographie pour sa simplicité et sa robustesse.

L'algorithme adapté pour Arduino en soi est une fonction de hashage prenant un bloc de données à encrypter de taille arbitraire en entrée, mélange les données à l'aide d'arithmétique binaire puis retourne un bloc de données encryptées de même taille. Pour le mélange des données, l'algorithme prend au départ une clé d'encryption de 16 bytes, un vecteur d'initialisation aléatoire de 8 bytes à usage unique, et un tableau de 8 bytes de valeurs servant de compteur. Ce qui rend cet algorithme très pratique est qu'on utilise la même séquence de calculs pour la décryption d'un bloc de données mélangées.

La différence principale entre les algorithmes Salsa20 et ChaCha se trouve dans les opérations de mélange de données qui sont altérés afin de mieux diffuser les données dans le bloc de données encryptées et profite d'une meilleure efficacité de calcul.

3. Rapport de développement

3.1 Défis et apprentissages

De manière générale, l'implémentation logiciele du matériel électronique s'est déroulée sans trop de problèmes grâce au support de la communauté du logiciel libre sur internet et de la panoplie de librairies disponibles faites sur mesure pour le matériel utilisé pour le projet. Le plus grand défi pour ma part fut la structuration du logiciel. C'est la première fois que j'avais un programme d'une telle envergure à écrire et avec plusieurs senseurs et affichage à gérer, l'approche à utiliser a dû être choisie de manière judicieuse. Par exemple, je devais me plier à la structure du script Arduino. Cela impliquait donc que puisque le corps du logiciel fonctionne sur une boucle, tout calcul que j'ajoute à l'intérieur de celui-ci aura un impact sur le temps d'exécution d'une boucle. Je devais donc éviter tout calcul qui soit se base sur le temps de boucle ou qui pourrait être grandement touché par la variation du temps de la boucle. Au départ, je voulais baser la détection de fluctuation de poids sur le taux de variation du poids (Δpoids/Δtemps) mais Sébastien, mon superviseur de projet, m'a mis en garde quant à la fragilité d'une telle approche. Il m'a donc conseillé d'utiliser la méthode actuelle où je compare constamment le poid mesuré avec la mesure précédente.

Ce projet m'a aussi permis de me familiariser avec le langage C++. En changeant de logiciel de programmation et en créant/modifiant des librairies pour les senseurs, j'ai eu l'opportunité de réviser les concepts de gestion de mémoire et d'arithmétique de pointeur. Ce sont des notions que je trouvais ambigües lorsque je les ai apprises dans le cadre de mes cours, et ce projet m'a permis de mieux saisir ces principes.

Un inconvénient rencontré durant le projet est que la compatibilité des modules avec les nouvelles plateformes Arduino. Au départ, nous voulions utiliser un plus petit modèle d'Arduino comme le UNO, mais les nombreuses librairies utilisées et l'envergure du code faisaient en sorte que la taille du programme dépassait grandement la taille disponible sur le microcontrolleur. Nous avons ensuite trouvé

une version compacte conçue par Adafruit possédant le 256 kb de mémoire en plus d'un lecteur de carte SD intégré. Cependant, la librairie du lecteur MFRC522 datant de plusieurs années déjà n'était pas compatible avec la plateforme Adafruit. Nous avons donc opté de garder le Arduino Mega malgré toutes les entrées et sorties de trop.

Un autre inconvéniant rencontré lors de la conception était l'aspect sécurité des données de la carte NFC. En ce moment, l'implémentation empêcherait de recopier les données d'une carte à un autre, mais rien n'empêche de cloner les données d'une carte pour la recopier sur elle même. Cela permettrait à quelqu'un de se recopier sa carte dans un état où elle détient un solde de 10$ par exemple. Sébastien et moi avions pensé à mettre un lecteur de carte SD qui permettrait de garder des informations de transaction en mémoire et d'écrire par exemple un timestamp ou la valeur d'un compteur sur la carte NFC à chaque transaction. Cela aurait permis de vérifier la carte NFC à la prochaine utilisation afin de voir si l'utilisateur ne tenterait pas d'utiliser la machine avec des données recopiées datant d'une ancienne transaction. Cela aurait pu être implémenté avec la plateforme d'Adafruit car possédait déjà un lecteur de carte SD, mais puisqu'elle était incompatible avec le lecteur MFRC522 et que la session était bientôt terminée, nous avons manqué de temps pour mettre ce système en place.

3.2 Expansions potentielles et considérations futures

Présentement ce projet est d'après moi une démonstration de faisabilité d'un système libre-service à paiement automatique. Il y a une panoplie d'améliorations potentielles à apporter, comme la mise en ligne du service afin qu'un utilisateur recharge sa carte vie une plateforme web ou même une application mobile utilisant le lecteur NFC d'un téléphone intelligent. Cela ajouterait plusieurs couches de complexités bien entendu comme l'ajout d'une plateforme pouvant accéder à l'internet comme le Raspberry Pi, mais cela est loin d'être impossible.

Il serait aussi envisageable de refaire la base de la cafetière en entier pour permettre de mettre une porte qui se barre lorsqu'on verse le café afin de prévenir le vol de café et les éclaboussures.

Un autre aspect important aussi à prendre en considération si l'on envisage de mettre ce produit en service auprès du public est le côté santé et salubrité. Présentement la base est contaminée par le plomb utilisé pour la soudure du matériel électronique et ne respecte pas les normes RoHS (Restriction of Hazardous Substances). Son utilisation pourrait donc être nocif pour la santé de ses utilisateurs. Il serait donc préférable de refaire des modèles avec une soudure RoHS afin d'éviter tout ennui.

4. Conclusion

Pour conclure, ce projet m'a permis de travailler plusieurs aspects en informatique que j'ai appris durant mon parcours universitaire. J'ai notamment révisé des notions de génie logiciel avec la structuration du logiciel ainsi que des principes de programmations vues dans le cours de concepts de langages de programmation. Je me suis aussi initié à de nouvelles technologies comme les cartes NFC et j'ai acquis quelques notions en cryptographie. Il serait maintenant intéressant de voir une version plus sophistiquées du système avec les fonctionnalitées mentionnées précédemment. Ils pourraient faire l'objet d'un autre projet IFT3150 par exemple et pourrait même être commercialisable suite à ces améliorations et peaufinements.