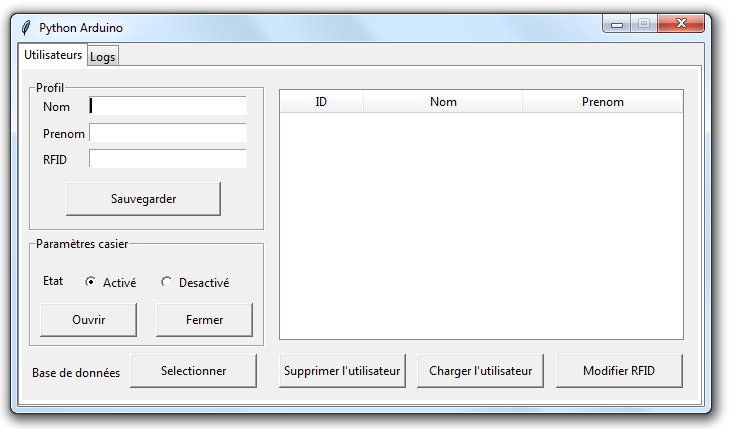
Thibault BOUTONNET

Projet:Un casier à cadenas électronique



Cahier des charges : Réalisation d’un programme en Python et Arduino permettant à un utilisateur de dévérouiller un casier (maquette) suite à la lecture d’un badge RFID personnel.

Problématique : Elargir l’utilisation de nos cartes de self\* pour permettre l’accès à un casier.

Avec Peyo FRAYSSINET

\*Actuellement nos cartes de self ne sont pas équipées de puce RFID. Seuls les professeurs ont des badges RFID. Les distributeurs de plateau au self fonctionnent avec le RFID. En passant sur des cartes avec puce RFID nous pourrions utiliser la même carte pour le self et pour déverrouiller notre casier.

Répartition des tâches :

Thibault :

– réalisation de l’IHM (Partie Python)

– participation à la partie Arduino

Peyo :

– réalisation de la fenêtre « Connexion » sur Python

– réalisation de la partie Arduino

– construction de la maquette.

Pour mener à bien ce projet nous nous sommes rencontrés à plusieurs reprises chez l’un et chez l’autre et nous avons régulièrement échangé par message et email.

I-Partie Python :

Le programme python est connecté à la base de données. C’est donc lui qui traite toutes les informations. L’arduino effectue juste les taches de capteurs et d’actionneur.

Lors de la réalisation de ce projet j’ai pu faire appel à diverses bibliothèques ;

* Tkinter
* Pyserial
* Sqlite3
* Threading
* Time

Tkinter m’a permis de faire la fenêtre de l’IHM.J’ai utilisé de nombreux éléments tels que :

* Les Notebook : c’est un onglet dans une fenêtre, on peut voir ça comme les onglets dans google chrome par exemple , dans notre cas c’est Utilisateurs et Log
* Les Entry : ce sont les textbox dans lesquelles on rentre le nom , le prenom et le RFID
* Les Button : bouton « sauvegarder » par exemple
* Les LabelFrame : ce sont des groupes d’éléments , dans notre cas nous avons 2 groupes : Profil et Parametre casier
* Les Radio : ce sont des cases à cocher , on ne peut pas cocher les 2 , c’est l’une ou l’autre
* Les Label : ce sont des éléments de texte
* Les Treeview : Ce sont les tableaux à colonnes

J’ai utilisé pyserial pour communiquer entre Pyhton et l’arduino. On utilise pour cela le port série .Le port série correspond aux ports COM sur les PC. Notre programme en Pyhton et l’arduino peuvent lire et envoyer des informations sur le port série en utilisant le protocole de communication série USB. L’arduino va donc evoyer le code RFID et le programme lui va envoyer le nom et le prénom ou alors un 0 si l’utilisateur n’est pas reconnu (flowchart un peu plus bas). Nous partons du principe qu’un utilisateur present dans la base de données est un utilisateur qui a accés au casier.On retrouve d’autres protocoles de communication série comme par exemple l’I2C. J’y reviendrai dans la partie Arduino.

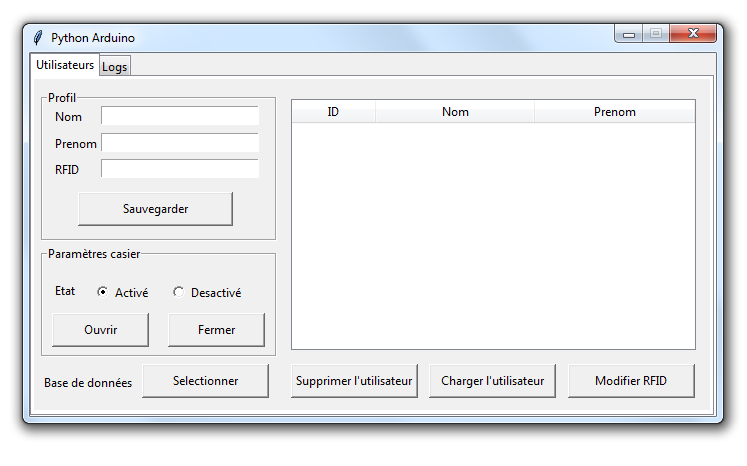
Sqlite3 me permet de gérer une base de données. Dans cette base de données on retrouve 3 tables :

* Logs : table dans laquelle on stocke l’ID , le nom , le prénom , l’heure et le RFID des passages
* Utilisateurs : table dans laquelle on retrouve l’ID ,le nom , le prénom et le RFID
* Sqlite\_sequence : cette table est gérée par la base de données elle-même , elle contient le numéro du dernier ID de chaques Tables. C’est elle qui permet une auto incrementation des ID.

Threading permet de lancer plusieurs éléments en simultané. Le langage python est linéaire ,il attend que la ligne d’avant soit fini pour executer celle d’aprés. Dans notre cas ce fonctionnement ne va pas. Une boucle infinie s’execute en arrière plan , elle ne s’arrête donc jamais , le programme serait donc en pause à l’infini. J’ai donc découpé le programme en 2 parties : la partie boucle infinie détaillée plus tard et la partie gestion des utilisateurs . Chaque partie correpond à un thread. Les 2 thread s’executeront en même temps (ils ne sont pas lancés en même temps dans notre cas). Je n’ai pas trouvé d’autre solution.La boucle infinie rend le programme très gourmand en CPU.

Gestion des utilisateurs :

Voici un visuel de la fenetre Tkinter

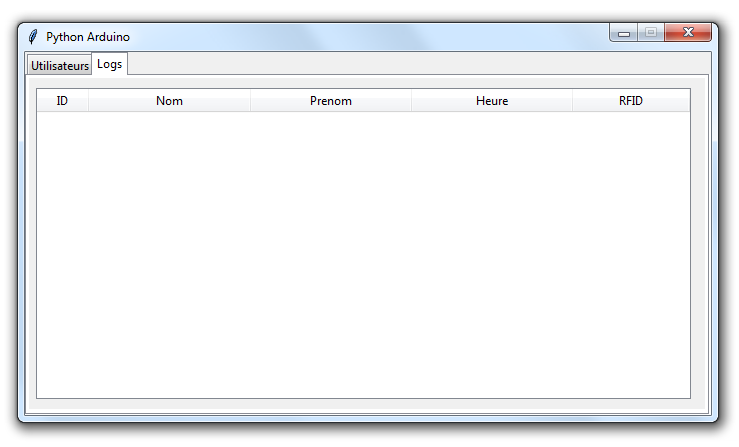


J’ai créé une IHM permettant de gérer des utilisateurs ainsi que l’ouverture du casier.

Ce logiciel nous permet d’ajouter, supprimer et modifier des utilisateurs. Chaque utilisateur est répertorié dans une base de données avec un ID, son nom, son prénom et son RFID en hexadécimal. Le contenu de cette base de données est affiché dans les tableaux à colonnes (Treeview).

L’administrateur peut choisir de mettre ou non un mot de passe pour se connecter au démarrage (Fenêtre connexion : Peyo) du logiciel. Ce mot de passe est là pour éviter que n’importe qui ayant accès au logiciel puisse ouvrir le casier. L’administrateur peut aussi choisir la base de données. L’administrateur peut choisir d’ouvrir le casier à n’importe quel moment. Il peut aussi désactiver le casier. C’est-à-dire que plus personne ne pourra ouvrir le casier (utile pendant les vacances).

Le changement de badge RFID (en cas de perte ou de vol) se fait directement depuis le logiciel. Il suffit d’appuyer sur le bouton « Modifier RFID» puis de passer son Badge pour que le code soit automatiquement saisi dans la case RFID.



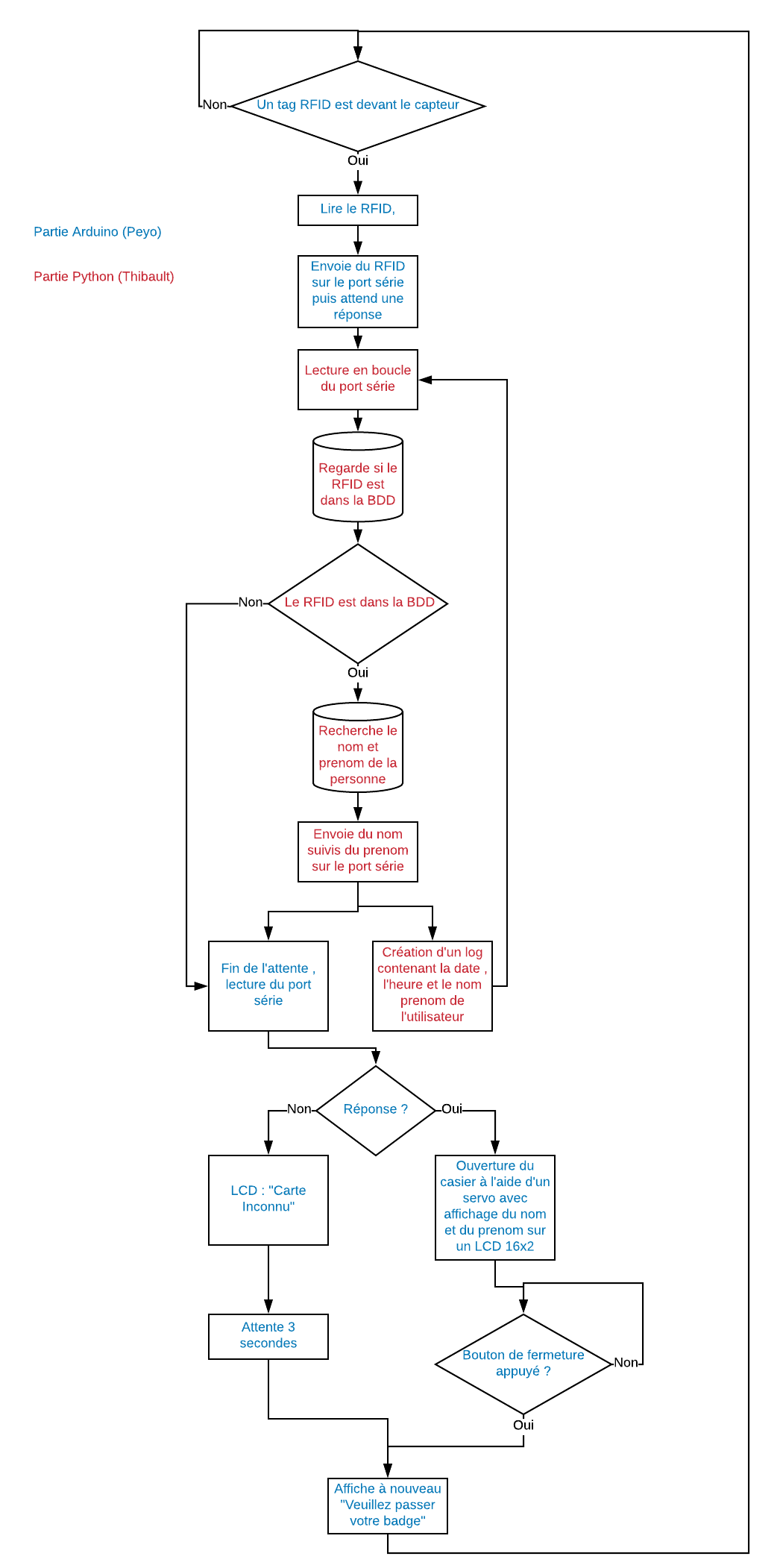
L’onglet Log lui est là pour garder un historique des actions utilisateurs. Dans notre cas nous avons un casier pour plusieurs personnes. On peut donc utiliser cet historique pour voir en cas de perte d’un objet qui a ouvert le casier. L’historique permet aussi d’avoir des statistiques sur l’utilisation des casiers. En effet si personne ne va au casier à certaines heures on pourrait par exemple mettre le casier en veille.

Boucle Infinie



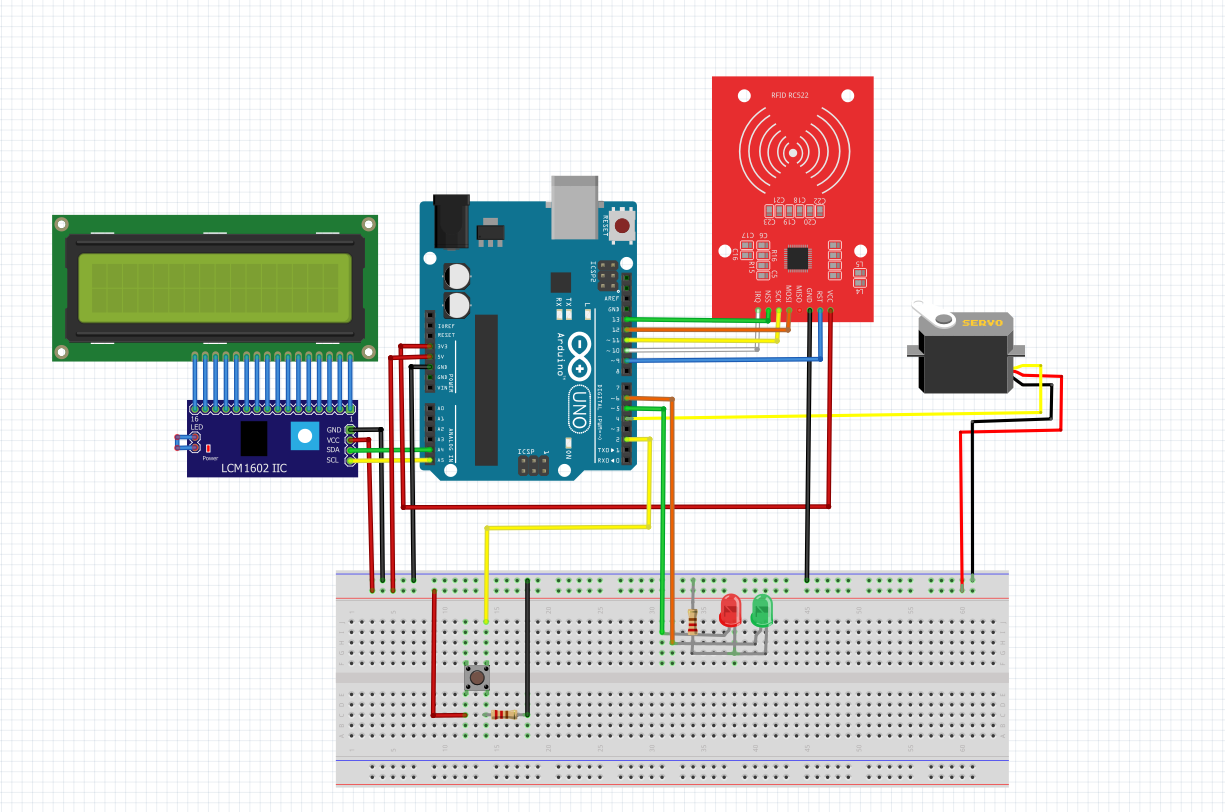
On retrouve aussi une boucle infinie, cette boucle est là pour traiter les informations envoyées par l’arduino. L’arduino va envoyer le Tag RFID (code de la puce RFID) et c’est la boucle infinie qui va se charger de voir si oui ou non un utilisateur existe avec ce Tag et s’il existe, de savoir qui c’est. Ensuite c’est cette boucle qui renvoie l’information nécessaire pour ouvrir le casier.

J’ai fait un flowchart avec Lucidchart pour vous représenter la reconnaissance des badges RFID.



II-Montage Arduino :

Notre câblage étant très peu lisible j’ai préféré utiliser le logiciel Fritzing pour générer un schéma propre du montage.



Composants

* 1x Arduino UNO
* 2x LEDs
* 1x Servo
* 1x RC522
* 1x I2C
* 1x LCD 1602
* 1x Résistance 220 Ohm
* Beaucoup de fils
* Des badges RFID

Nous avons opté pour un Arduino UNO, nous n’avions pas besoin des fonctionnalités des autres cartes (LEONARDO, MEGA..) et nous n’avions pas de contrainte de place (Mini).

Le module I2C nous permet de connecter l’écran à l’arduino avec seulement 4 Pin. 2 d’alimentation et 2 de signal (clock et data). Pas besoin non plus de monter un potentiomètre pour régler le contraste, il est déjà sur le module I2C.

L’écran comporte 2 lignes de 16 caractères. On utilisera les caractères par défaut mais on peut tout à fait créer nous-même nos caractères (5x8).

Nous utilisons une seule résistance pour nos 2 leds.Elle est là pour éviter de griller les leds.

Le module RC522 est un lecteur de carte sans fil RFID. Les cartes RFID sont dotées d’une antenne et d’une puce) .La portée est vraiment très limitée mais suffisante dans notre cas.

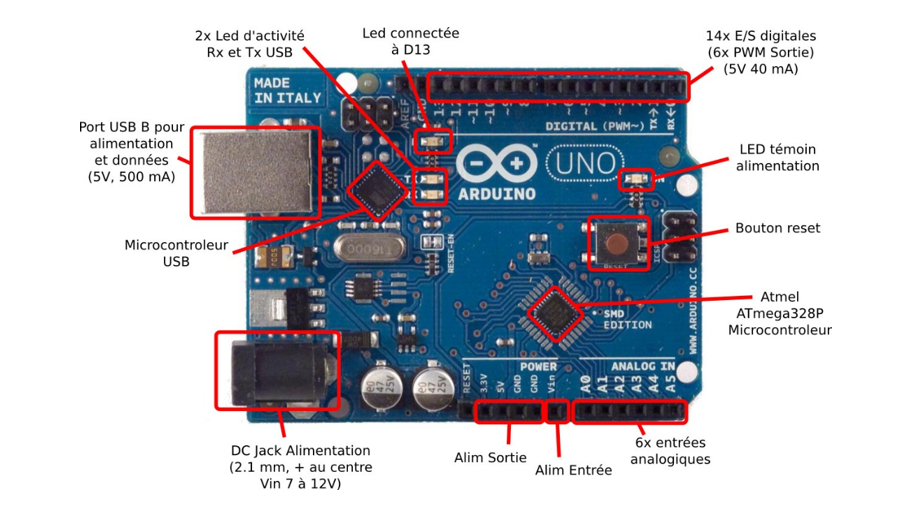
Le servo est un servo avec 1.2Kg/cm de couple. Il est là à des fins démonstratives. Si cette solution de casier devait être commercialisée en aucun cas je laisserais un servo. Un électroaimant est bien plus performant à mon avis.

Arduino :

« L’Arduino est une plateforme de prototypage électronique open-source, basée d’une part sur du matériel et d’autre part sur un ensemble de logiciels faciles à utiliser. »

Open source signifie que n’importe qui a accès au schéma de la carte et a donc le droit de le modifier ou de le redistribuer.

Voici la composition d’un Arduino (image provenant d’Internet) :



Je laisse Peyo vous expliquer en détail sa partie. Mis à part un problème avec l’i2c c’est lui qui s’est occupé de l’Arduino.

Bilan :

Hélas, notre projet n’est pas parfait, il y a sûrement de nombreux bugs que nous n’avons pas vus et de nombreux manques d’optimisation. Nous avons pensé à de nombreuses améliorations possibles. Nous avons choisi de faire exécuter les recherches dans la base de données par notre programme .Cette solution est efficace mais le logiciel doit être allumé pour que le casier fonctionne correctement. On pourrait porter la recherche des utilisateurs sur l’arduino. Il faudrait modifier le code de l’arduino pour qu’il puisse vérifier lui-même dans la base de données les accès des utilisateurs. Ce genre de modification est tout à fait faisable. Il faudrait monter sur l’arduino un autre système de communication que l’USB .On pourrait par exemple monter une Ethernet Shield pour accéder à une Base de données distante. Une fois alimenté et connecté à internet l’arduino serait autonome. On pourrait aussi ajouter une recherche automatique du bon port. Actuellement nous sommes obligés de saisir manuellement dans le code le bon COM. On pourrait ajouter d’autres données sur les utilisateurs : par exemple une adresse, une photo, les retards .Notre système pourrait être lié à un ENT. Il n’y a pas non plus de zone de recherche pour trouver un certain utilisateur. Il faut donc scroller jusqu’à ce que l’on trouve l’utilisateur voulu.

Partie Python disponible sur Pastebin : https://pastebin.com/w44U9Awx

Partie Arduino disponible sur Pastebin : https://pastebin.com/QhnTUEvn