Rapport Tuteuré

Gestion de l'accès d'un parking à vélo avec un Raspberry Pi en langage Python

Développement du plan

Introduction

Conclusion

Introduction

Le projet en deuxième année de DUT Génie Electrique et Informatique Industriel est un projet assez conséquent puisqu'il se déroule sur la totalité du semestre 3 et du semestre 4. Le choix du sujet s'est porté sur la gestion de l'accès d'un parking à vélo avec un Raspberry Pi en langage Python.

C'est un projet qui nous fait travailler sur différents domaines que l'on a vu au cours de notre formation : développement d'une base de donnée, conception et réalisation d'une carte électronique mais aussi de la programmation. De plus, celui-ci nous a permis de pouvoir commencer à apprendre un nouveau langage de programmation qui est le Python.

L'équipe en charge de ce projet est composée de Hervé Croissant et de Dimitri Texier. Mme Jacob nous a proposé ce sujet et assure l'encadrement avec M Walter.

Nous allons donc voir à travers ce rapport dans une première partie, une présentation du projet ainsi que ces principaux objectifs. Puis dans une seconde partie, nous allons voir le déroulements de notre projet. Enfin, dans une dernière partie, nous verrons les résultats obtenus ainsi que les améliorations possibles du projet.

1. Etude du projet

1.1. Les contraintes

Afin de répondre au cahier des charges qui nous a été donné, nous avons été soumis à des contraintes aussi bien matérielles que techniques.

1.1.1. Les contraintes matérielles

La base de notre projet est l'utilisation d'un Raspberry Pi modèle B (cf 2.1.2. Prise en main du Raspberry Pi). En effet, la programmation que l'on doit effectuer pour gérer l'accès au parking à vélo sera effectué grâce à celui ci.

Dans le cadre de ce projet, nous allons associé le Raspberry Pi à un lecteur de carte à distance utilisant la technologie NFC[1] : l'Explore_NFC. Le fonctionnement de celle-ci sera détaillé un peu plus loin.

Nous devons aussi prévoir la fabrication d'une carte électronique qui va nous permettre d'interagir avec le Raspberry Pi et le lecteur de carte mais aussi de fournir l'alimentation du Raspberry Pi qui est en 5V.

Raspberry Pi avec son lecteur de carte

1.1.2. Les contraintes techniques

En ce qui concerne les contraintes techniques, il y en a qu'une seule. En effet, toute la partie programmation de notre projet devait être effectué en langage Python, un langage que nous n'avons pas étudié en cours.

[1] Near Field Communication ou communications en champ proche

1.2. Elaboration du cahier des charges

Le cahier des charges a été défini avec nos professeurs référents. Cependant, nos choix ont été assez libre. On s'est vite décidé sur le cahier des charges et les fonctionnalité qu'aurait notre projet. Le but principal de notre projet est de pouvoir contrôler l'ouverture d'une porte de garage à vélo à l'aide du Raspberry Pi et du lecteur de carte NFC.

Notre système disposera donc de deux modes : un mode administrateur et un mode normal.

En mode normal, lorsqu'une carte qui est enregistrée dans une base de donnée que nous avons créée, est détectée, cela entraine l'ouverture de la porte du garage à vélo. En mode administrateur, une personne disposant d'une carte maître peut si il le souhaite autoriser une carte ou au contraire enlever l'autorisation d'une carte à l'accès dans le garage à vélo. L'autorisation d'ouverture de la porte du garage se fera en passant la carte maître devant le lecteur et en appuyant sur un bouton poussoir qui est situé sur notre carte électronique. Ce sera la même chose pour enlever l'autorisation mais en appuyant sur un autre bouton poussoir.

Un système de LEDs sera mis en place afin d'avoir un interface pour l'utilisateur si l'on ne dispose pas d'écran d'ordinateur. En effet, si on est autorisé à entrer dans le garage, une LED verte s'allumera. SI la carte n'est pas autorisé à entrer, une LED rouge sera allumé. Ces LEDs vont aussi nous permettre de voir si l'on est dans le mode administrateur grâce à un clignotement de celles-ci.

2. Déroulement du projet

2.1. Maitriser de langage Pyhton

Python est un langage qui permet une approche modulaire et orientée objet de la programmation. Il est développé depuis 1989 et est directement installé avec Raspbian. C'est un des langages informatiques les plus populaires avec C, C++, C#, Objective-C, Java, PHP, JavaScript, Delphi, Visual Basic, Ruby et Perl.

Actuellement, le langage Python est gratuit, sous licence libre et en est à sa version 3.
Cependant, la version 2 est encore largement utilisée et Python2 n'est pas compatible avec Python3. Nous avons choisi de travailler avec Python 2.7.

Les avantages de Python sont nombreux, c'est un langage :

• Facile à apprendre, à lire, à comprendre et à écrire ;

• Portable, dynamique, extensible, gratuit ;

• Il doté d'une communauté active.

L'interpréteur Python commence par analyser la première ligne:

• Si celle-ci contient une instruction, alors il l'exécute ;

• Si l'instruction n'est pas une instruction de contrôle, alors, il passe à la ligne suivante, l'analyse et l'exécute ;

•Si le programme Python arrive à la fin du fichier à exécuter, alors, il sort du programme et en arrête l'exécution.

Il est finalement assez proche du C/C++, et la plupart des commandes ainsi que des types de variables sont les mêmes.

Exemple :

```
nom  = 'Python'
print "Programmer en %s, c'est facile." % nom
```

En Python, les blocs sont identifiés par l'indentation, au lieu d'accolades comme en C ou C++. Une augmentation de l'indentation marque le début d'un bloc, et une réduction de l'indentation marque la fin du bloc courant.

Exemple :

```
mois = raw_input
if mois == 'Décembre':
        print 'Joyeux Noël'
elsif mois == 'Janvier':
        print 'Bonne année'
else :
        print 'rien ce mois'
```

Le python devient plus complexe quand on veut interagir avec une base de données. C'est pour cela qu'on y ajoute la bibliothèque permettant d'utilisé des requêtes SQL. La bibliothèque utilisé pour notre base de donnée est Sqlite3. Il s'agit d'une

bibliothèque qui s'adapte très bien pour le Raspberry Pi.

Sqlite3 est un système de bases de données léger. Une base de données (*database* en anglais) est un conteneur dans lequel il est possible de stocker des données de façon structurée. Cette structure permet au programme informatique connecté à celle-ci de faire des recherches complexes. Dans notre cas, la base de donnée sera utilisée pour stocker toutes les cartes qui seront autorisées à entrer dans le garage à vélo.

Un langage standardisé -SQL- est dédié à cette structure et permet aussi bien de faire des recherches mais aussi des modifications ou des suppressions.

L'installation sous Raspbian est finalement simple, puisqu'elle se fait en quelques lignes de commandes :

```
apt-get update
apt-get install sqlite3
sudo Sqlite3 test2.db          // Cré et ouvre la base de données "test2"
```

Ensuite pour la table:

```
$sqlite3 test2.db
SQLite version 3.7.15.2 2013-01-09 11:53:05
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite>
sqlite> CREATE TABLE Card (
...>   Cartes varchar(40) primary key
...> );
sqlite>
```

Une base SQLite3 a la particularité d'être contenue dans un fichier qui porte le même nom. Pour l'utiliser en Python, nous avons donc juste importé la bibliothèque. Ensuite, il faut créer un curseur pour pointer les requêtes vers la base de données.

```
import sqlite3

conn = sqlite3.connect ('test2.db')    # création du curseur permettant la connexion avec la bdd.

conn.execute("INSERT INTO carte (card) values (EAD058EDR)")
# grace au curseur, on peut utiliser des requêtes SQL.

conn.commit()    # Permet de rendre effectif  les changements apportés à la bdd .

conn.close()
```

2.2. Prise en main du Raspberry Pi et de la carte Explore-NFC

2.2.1. Le Raspberry Pi

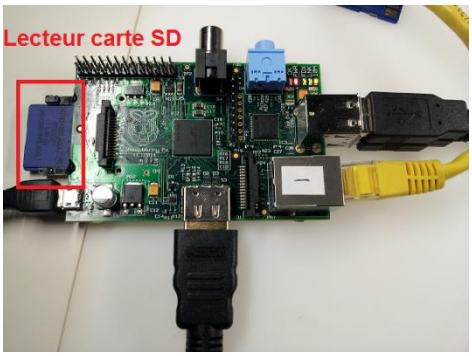
Qu'est ce que c'est ?

Le Raspberry Pi est un nano-ordinateur avec un processeur ARM[2]. Il a été conçu par la fondation Raspberry Pi dirigé par David Braben. Il s'agit d'un ordinateur qui à la taille d'une carte de crédit (85.60mm * 53.98mm*17mm). Il est destiné à encourager l'apprentissage de la programmation informatique. Il permet d'exécuter de nombreuses variantes du système d'exploitation Linux. La plus utilisée et celle que nous utilisons pour notre projet est Raspbian.

Raspberry Pi modèle B

Il est fourni nu. C'est-à-dire qu'il n'y a pas de clavier, pas d'alimentation, pas de souris et pas d'écran. Ce qui en fait son énorme succès est son prix de vente qui est d'environ 25 euros ce qui est relativement faible contrairement à un ordinateur "classique". Plus de trois millions de Raspberry Pi ont déjà été écoulé depuis début 2013.

Le modèle que nous allons utiliser pour notre projet est le modèle B (voir photo). Il est composé d'un port GPIO[3]. Ce sont des ports d'entrée/sortie très utilisés dans le monde des microcontrôleurs, en particulier dans le domaine de l'électronique embarquée. Ce port va permettre au Raspberry Pi d'interagir avec l'extérieur. Il est aussi composé de 2 ports USB, d'un port réseau, 2 sorties vidéos (HDMI et composite), 1 sortie audio et aussi un port micro-USB pour l'alimentation dont on se passera dans notre projet puisqu'on alimentera le Raspberry Pi avec la pin 1 du GPIO. Il faudra donc alimenter le Raspberry Pi en 5V. Le Raspberry Pi dispose aussi d'un port pour la carte SD qui constitue la mémoire de masse du nano ordinateur vu qu'il ne dispose pas de mémoire de masse intégré.



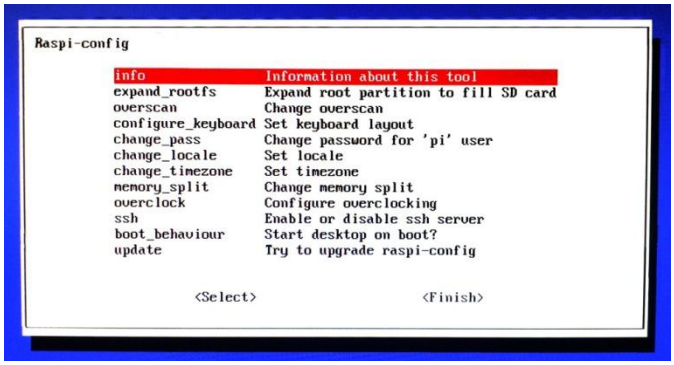
Le raspberry Pi avec tous ces connecteurs

[2] Les architectures ARM sont des architectures RISC 32 bits (ARMv1 à ARMv7) et 64 bits (ARMv8)[1] développées par ARM Ltd depuis 1990 et introduites à partir de 1983 par Acorn Computers.
[3] General Purpose Input/Output

Installation du Raspberry Pi

Tout d'abord avant de commencer l'installation, il faut se munir d'un écran disposant d'une prise HDMI, d'un clavier et d'une souris mais aussi d'une carte SD.

Comme cela est dit un peu plus haut, le système d'exploitation utilisé est Raspbian, une distribution de Linux adaptée pour le Raspberry Pi. Tout d'abord, il faut installer Raspbian sur une carte SD (minimum 8go) grâce à un ordinateur. Une fois l'installation terminé, il faut connecter le câble Ethernet au Raspberry Pi, le clavier et la souris, mais aussi l'écran à la prise HDMI. Il ne faut pas oublier le plus important, insérer la carte SD dans le port prévu à cet effet. Une fois tous les périphériques installés, on peut alimenter le Raspberry Pi.



Raspi-config

Maintenant que tout est connecter et après la mise sous tension du Rapsberry PI, il faut sélectionner Raspbian dans la fenêtre qui apparaît. L'installation se fera automatiquement. Au premier démarrage, on a le droit au lancement automatique de l'utilitaire de configuration "raspi-config" qu'il est possible de rappeler par la suite en saisissant la commande suivante dans un terminal : sudo raspi-config.

Voici un descriptif des différentes fonctions qu'il faut sélectionner les unes après les autres :

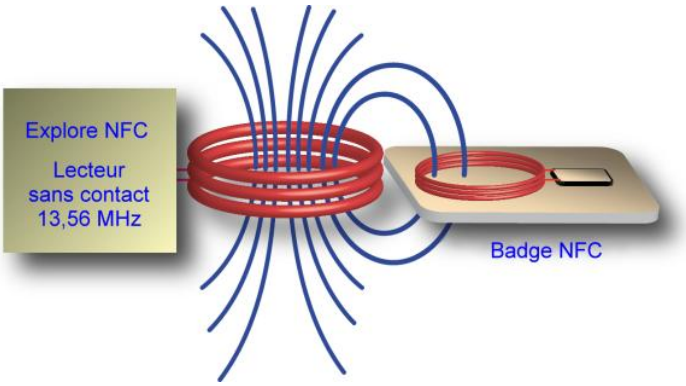
- **expand_rootfs** : Va permettre d'utiliser tout l'espace de la carte SD
- **configure_keyboard** : Pour mettre le clavier en français, on sélectionne "Generic 105-key (intl) PC", puis "Other", puis "French", puis "French", puis "The default for the keyboard layout", puis "No compose key" et pour finir "No".
- **change_pass** : modifier le mot de passe de l'utilisateur 'pi' (par défaut : raspberry)
- **change_locale** : changer la langue du système
- **change_timezone** : changer l'heure locale
- **ssh** : activer le SSH ce qui permet de contrôler le raspberry à distance
- **boot_behavior** : Pour choisir si on veut arriver directement sur le bureau au démarrage du systeme et pas sur une console.

Une fois sur l'invite de commande, il faut mettre le login et le mot de passe. Il faut entrer « **pi** » pour le login et « **raspberry** » pour le mot de passe. On a choisi d'arriver sur la console au démarrage. Si l'on veut démarrer l'interface graphique, il faut taper **startx** sur la console.

Lorsque l'on est sur l'interface graphique, pour accéder à une commande et dans notre cas pour exécuter un programme, il faut utiliser LXTerminal. Pour éteindre le Raspberry Pi, il faudra passer par la commande : sudo half.

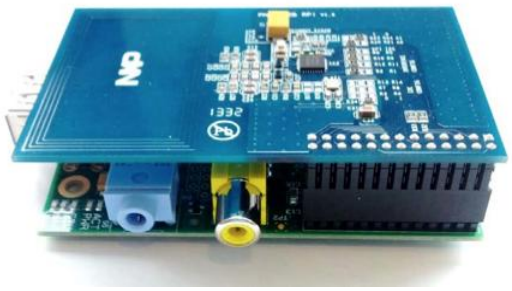
2.2.2. La carte Explore-NFC

Explore-NFC est le nom qu'a donné element14, la communauté de développement créée par le distributeur Farnell, à une carte d'extension destinée au Raspberry Pi . Cette carte va permettre de développer des projets utilisant la communication NFC qui est une technologie de communication à courte portée (environ 10 cm maximum). Cette carte offre une antenne et une puce intégrée. Elle se connecte sur les ports GPIO du Raspberry.



Schématisation du fonctionnement de la lecture NFC

Cette carte est celle que nous allons utiliser afin de gérer l'ouverture de notre porte de garage à vélo. Elle permet la communication en champ proche. Elle est dotée d'une antenne intégrée hautes performances. La bibliothèque que nous avons installé sur le Raspberry Pi afin que cette carte fonctionne est libnfc.



La carte Explore-NFC installée sur le Raspberry Pi

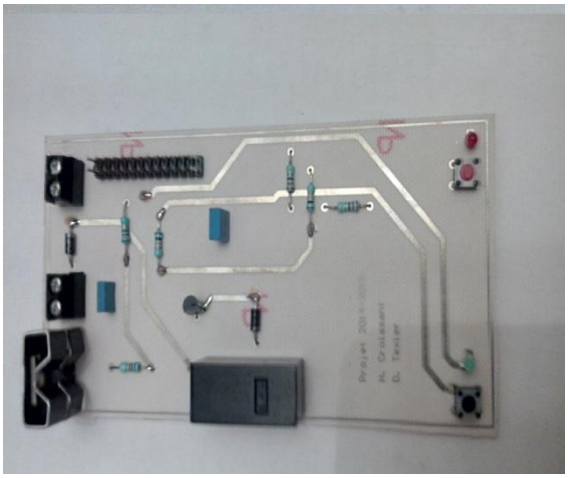
La lecture d'une carte nfc se fait par l'émission d'une onde électromagnétique par le lecteur. La fréquence du signal émis est de 13,56 MHz. L'énergie fabriquée sert a alimenter l'antenne du bagde récepteur. L'énergie va faire varier le courant dans l'antenne

du badge. Ces variations de courants sont transmises au lecteur qui peut en déduire le tag[4] associé
.

2.3. Réalisation de notre carte électronique

Les solutions techniques que nous avons choisies imposent l'utilisation d'une alimentation en 12V. Cette tension a été choisi car il s'agit de la tension qui alimente la gâche électrique de notre porte de garage à vélo. On a choisit de mettre sur cette carte un relais pour déclencher la gâche, un LM7805 pour abaisser la tension de 12V à 5V afin d'alimenter le Raspberry Pi, 2 boutons poussoirs pour l'ajout ou la suppression de carte NFC dans notre base de données et 2 LEDs pour avoir une interaction visuel lorsque le Raspberry Pi n'est pas connecté à un écran. Cette carte est aussi composé de deux borniers. Un pour l'alimentation de la carte électronique en 12 V et un pour pouvoir connecter le Raspberry Pi à la gâche.

Nous avons réalisé une première carte électronique (de test) afin d'ajouter ces composants, d'alimenter la gâche en 12V (via le relais) et le Raspberry PI en 5V. Nous avons rencontré quelques problèmes. L'un des problèmes majeurs qui nous a pris du temps est le relais qui se déclenchait de manière aléatoire. On s'est rendu compte, après avoir observé le Raspberry Pi, qu'une des piste qui était relié au relais était en contact avec le port USB du Raspberry Pi. Un autre problème a été l'utilisation d'une des pins par un de nos boutons poussoirs du GPIO qui était utilisé par le lecteur NFC. Cette pin étant donc utilisé, le bouton relié à cette pin ne fonctionnait pas. Le dernier problème que nous avons eu a été un problème d'alimentation au niveau du Raspberry Pi. En effet, le LM7805 fournissait la tension nécessaire mais il ne fournissait pas assez de courant pour le Raspberry Pi. Cela venait d'un problème de surchauffe au niveau du LM780. On a donc du ajouter un dissipateur thermique qui a résolu le problème. Après avoir corrigé ces erreurs, nous avons entreprit la réalisation d'une nouvelle carte avec un placement optimisé des composants pour ne plus avoir de faux contact. Nous voulions aussi faire un carte beaucoup plus petite que la taille de la première. On voulait une carte de la largeur du Raspberry Pi.



Notre carte électronique finale

[4] Il s'agit d'une étiquette électronique

2.4. La répartition du travail

Nous avons réparti le travail en 4 parties:

- La prise en main du Raspberry PI
- Le Cahier des charges
- La Programmation
- Carte électronique

Nous avons pris en main le Raspberry PI ensemble, puis à la suite de notre premier contact nous avons pu établir le cahier des charges avec madame Jacob et monsieur Walter.

De ce cahier des charges a découlé deux grands axes de travail:

- La programmation avec la gestion de la base de données et le programme en Python ;
- La conception et la réalisation des deux cartes électroniques (prototype et carte final).

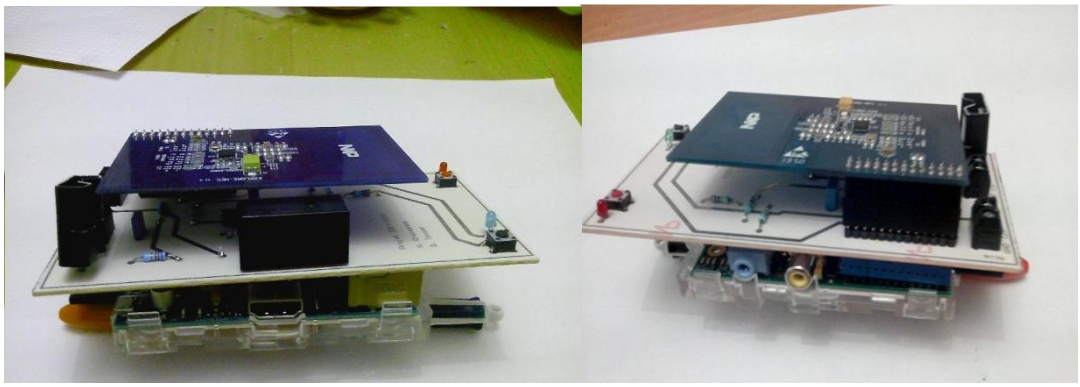
Au vue de nos qualités respectives, la partie programmation a été supervisé par Hervé et la partie réalisation des cartes électroniques a été supervisé par Dimitri.

3. Résultats obtenus

3.1. Notre projet final

Le résultat que nous avons obtenu est celui que nous voulions au départ. En plus de cela, une maquette de la porte du garage a été réalisé par les techniciens du département GEII de l'IUT. Cette porte a été réalisé rapidement afin que notre projet soit présenté lors des portes ouvertes qui ont eu lieu le 14 février 2015.

La carte que nous avons réalisé correspondait à nos attentes. Celle-ci vient s'intercaler entre le Raspberry Pi et la carte Explore-NFC. On peut le voir ci-dessous :



Raspberry PI, carte électronique et lecteur Explore-NFC

Notre porte de garage à vélo

Lorsque l'on passe une carte NFC enregistrée dans la base de donnée, la gâche se déclenche et la LED vertes s'allume. On peut ainsi ouvrir la porte du garage à vélo. Si une carte qui n'est pas enregistré dans la base de donnée est passé devant le lecteur NFC, la porte de s'ouvre pas et une LED rouge s'allume indiquant que la carte n'est pas autorisé à entrer dans le garage à vélo.

On peut aussi ajouter une carte en passant la carte NFC maître et en appuyant sur un des boutons poussoirs. Lorsque la LED verte se met à clignoter, on peut passer devant le lecteur la carte à ajouter. Si la carte est dejà autorisé, la LED rouge se met à clignoter. Il s'agit du même procédé pour supprimer une carte de la base de donnée. Cependant, on utilisera l'autre bouton poussoir.

3.2. Les améliorations possibles

Ce projet qui correspond à nos attentes peut être amélioré. Un boitier peut être créé à l'aide d'une imprimante 3D afin de donnée un aspect plus fini à notre projet.

Une autre amélioration aurait été la possibilité d'associer le TAG enregistré dans la base de données à un prénom et à un nom. Cela permettrait une gestion plus facile de la base de données. L'administrateur pourrait ainsi supprimer l'autorisation d'une carte sans avoir la carte à disposition et juste avec le nom et le prénom de la personne. Pour cela, il faudrait connecter le Raspberry Pi à internet pour pouvoir modifier la base de données.

Afin de facilité la gestion de cette base de donnée, un site internet pourrait être créer pour permettre à l'administrateur d'avoir une gestion de celle-ci plus intuitif. Cela éviterait de passer par des lignes de code.

La dernière amélioration serait l'intégration d'un écran sur l'ensemble Raspberry Pi, carte électronique et lecteur NFC. Cela pourrait permettre de voir le nom de la personne qui rentre dans le garage à vélo mais aussi de voir de façon plus visuel si il est autorisé ou non à entrer dans le garage. Cela est plus visuel que de simples LEDs.

Conclusion

Ce projet fût un projet très intéressant mais aussi pour nous une expérience enrichissante qui nous a permis de découvrir le Raspberry PI. De plus, nous avons découvert le langage Python que nous avons facilement appréhendé car il se rapproche du langage C que nous avons étudié en cours d'Informatique.

Ce projet a été un projet très motivant puisque c'est un projet qui pourrait dans un futur proche être intégré dans un garage à vélo, celui de l'IUT par exemple

Annexes

Table des illustrations

Notre programme entièrement commenté :

```
# Importation des bibliotheque

import RPi.GPIO as GPIO

import time

import nxppy

import sqlite3

import sys


# Connexion a la base de donnee

connection=sqlite3.connect('nfc2.db')

curs=connection.cursor()


# Definition des ports en entre/sortie

GPIO.setmode(GPIO.BCM)

GPIO.setup(4,GPIO.OUT)

GPIO.setup(17,GPIO.OUT)

GPIO.setup(18,GPIO.OUT)

GPIO.setup(22,GPIO.IN)

GPIO.setup(25,GPIO.IN)


# Debut du programme

while True :

uid = nxppy.read_mifare() # Lecture et insertion dans une variable du
numero de la carte NCF presentee

#Condition pour l'ajout d une carte

if uid == "04F1D561EE0280": # Si la carte maitre est detectee
```

```
if (GPIO.input(22)==0): # Si le bouton poussoir noir est appuye

print "Presente la carte a ajouter"

GPIO.output(17,GPIO.HIGH) # Clignotement LED

time.sleep(0.4)

GPIO.output(17,GPIO.LOW)

time.sleep(0.4)

GPIO.output(17,GPIO.HIGH)

time.sleep(0.4)

GPIO.output(17,GPIO.LOW)

time.sleep(0.4)

GPIO.output(17,GPIO.HIGH)

time.sleep(0.4)

GPIO.output(17,GPIO.LOW)

time.sleep(0.4)

GPIO.output(17,GPIO.HIGH)

time.sleep(0.4)

GPIO.output(17,GPIO.LOW)

time.sleep(0.4) # Fin clignotement


ajou = nxppy.read_mifare() # Lecture et insertion dans une variable de la
carte NCF presentee

curs.execute('SELECT * FROM carte') # Lecture de la base de donnee

listebdd = curs.fetchall() # Insertion BDD dans la variable listebdd


if (str(ajou) in str(listebdd)): # Si la carte est dans la BDD

print ('la carte ' + str(ajou) + ' est deja autorisee')
```

```
GPIO.output(18,GPIO.HIGH) # Allumer LED rouge


elif (str(ajou)=="None"): # Si pas de carte detectee

print "Aucune carte detectee"

GPIO.output(18,GPIO.HIGH) # Allumer LED rouge


else:

curs.execute("INSERT INTO carte (card) values (?)",(ajou,)) # Ajout dans
la BDD

print ('la carte ' + str(ajou) + ' a ete ajoute')

connection.commit() # Enregistrement des modifications dans la BDD

GPIO.output(17,GPIO.HIGH) # Allumer LED verte

time.sleep(1)

GPIO.output(17,GPIO.LOW) # Eteindre LED verte

time.sleep(1)

GPIO.output(18,GPIO.LOW) # Eteindre LED rouge



#Condition pour la suppression d une carte

if GPIO.input(25)==0: # Si le bouton poussoir rouge est appuye

print "Presente la carte a supprimer"

GPIO.output(18,GPIO.HIGH) # Clignotement LED rouge

time.sleep(0.4)

GPIO.output(18,GPIO.LOW)

time.sleep(0.4)

GPIO.output(18,GPIO.HIGH)
```

```
time.sleep(0.4)

GPIO.output(18,GPIO.LOW)

time.sleep(0.4)

GPIO.output(18,GPIO.HIGH)

time.sleep(0.4)

GPIO.output(18,GPIO.LOW)

time.sleep(0.4)

GPIO.output(18,GPIO.HIGH)

time.sleep(0.4)

GPIO.output(18,GPIO.LOW)

supr = nxppy.read_mifare() # Lecture et insertion dans une variable du
numero de la carte NCF presentee

curs.execute('SELECT * FROM carte') # Lecture de la base de donnee

bas = curs.fetchall() # Insertion BDD dans la variable bas



if supr == "04F1D561EE0280": # Si on presente la carte maitre

print ("la carte maitre ne peut pas etre suprimmee")

GPIO.output(18,GPIO.HIGH) # Allumer LED rouge

time.sleep(1)

GPIO.output(18,GPIO.LOW) # Eteindre LED rouge



elif str(supr) in str(bas): # Si la carte est dans la BDD

curs.execute("delete from carte where card=(?)",(supr,)) # Suppression
dans la BDD

print ('la carte ' + str(supr) + ' a ete supprime')

connection.commit() # Enregistrement des modifications dans la BDD

GPIO.output(17,GPIO.HIGH) # Allumer LED verte
```

```
time.sleep(1)

GPIO.output(17,GPIO.LOW) # Eteindre LED verte

else :

print("La carte n est pas dans la base de donnee")

GPIO.output(18,GPIO.HIGH) # Allumer LED rouge

time.sleep(1)

GPIO.output(18,GPIO.LOW) # Eteindre LED rouge



lect = nxppy.read_mifare() # Lecture et insertion dans une variable du
numero de la carte NCF presentee

if (GPIO.input(22)==1): # Si aucun bouton poussoir n est presse

if (GPIO.input(25)==1):

curs.execute('SELECT * FROM carte') # Lecture de la base de donnee

base = curs.fetchall() # Insertion BDD dans la variable base

if str(lect) in str(base): # Si la carte est dans la BDD

print "allumer, c'est ouvert"

GPIO.output(17,GPIO.HIGH) # Allumer LED verte

GPIO.output(4,GPIO.HIGH) # Declencher relais

time.sleep(5)

GPIO.output(17,GPIO.LOW) # Eteindre LED verte

GPIO.output(4,GPIO.LOW) # Arret du declenchement du relais

print "on ferme"

time.sleep(1)

connection.close() # Fermeture de la connection avec la BDD
```