Compte Rendu de Projet :

Table des matières

[I. Introduction : 2](#_Toc40563963)

[II. Réalisation du projet : 2](#_Toc40563964)

[a) Rappel de la tâche de l’étudiant 2](#_Toc40563965)

[b) Communication entre les différents éléments 3](#_Toc40563966)

[c) Communication 4](#_Toc40563967)

[d) Câblage et choix des PIN 4](#_Toc40563968)

[e) Chronogrammes 5](#_Toc40563969)

[f) Système socket 6](#_Toc40563970)

[g) Code du client 7](#_Toc40563971)

[III. Réalisation de l’application : 7](#_Toc40563972)

[a) Récupération de l’UID 7](#_Toc40563973)

[b) Allumage des LEDs et déclanchement du module relais 8](#_Toc40563974)

[c) Bibliothèque RFID 9](#_Toc40563975)

[IV. Technologie RFID 10](#_Toc40563976)

[V. Fiches recettes 11](#_Toc40563977)

[VI. Conclusion 11](#_Toc40563978)

[a) Communication avec le groupe 11](#_Toc40563979)

[b) Difficultés liées au confinement dû au COVID-19 12](#_Toc40563980)

[c) Connaissances acquises 12](#_Toc40563981)

[d) Ce qui reste à faire 12](#_Toc40563982)

[e) Poursuite d’étude 12](#_Toc40563983)

[VII. Annexes 12](#_Toc40563984)

[a) Annexe 1 13](#_Toc40563985)

# Introduction :

#### **Rappel du cahier des charges :**

Afin que la partie se déroule correctement un système pour les « médaillons » est à fournir.

Ce système sera au cœur même de la salle de l’escape game et devra prendre en charges les demandes suivantes :

* Le lecteur RFID doit pouvoir lire un UID (*User Identifier*) à travers une plaque de contreplaqué de 3mm.
* Utiliser le bus SPI (*Serial Peripheral Interface*) de la Raspberry pour le contrôle des lecteurs RFID.
* Le superviseur doit pouvoir récupérer UID (*User Identifier*) afin de le changer facilement pour le programme principal dans le cas d’une perte d’un des médaillons.
* Le joueur doit pouvoir constater l’allumage ou non d’une LED dans la seconde en fonction de l’état d’un lecteur RFID (si un badge ou « médaillon » valide est posé dessus ou non).
* Dans le cas où le joueur place correctement quatre badges la gâche électrique doit s’activé permettant l’ouverture de la porte. Un ordre de fin de partie doit également être envoyer via un système de client socket.
* L’application de supervisions doit pouvoir détecter l’ordre de fin de partie via un système serveur socket.

# Réalisation du projet :

## Rappel de la tâche de l’étudiant

Au sein de ce projet d’escape game, ma partie consiste à créer un système permettant :

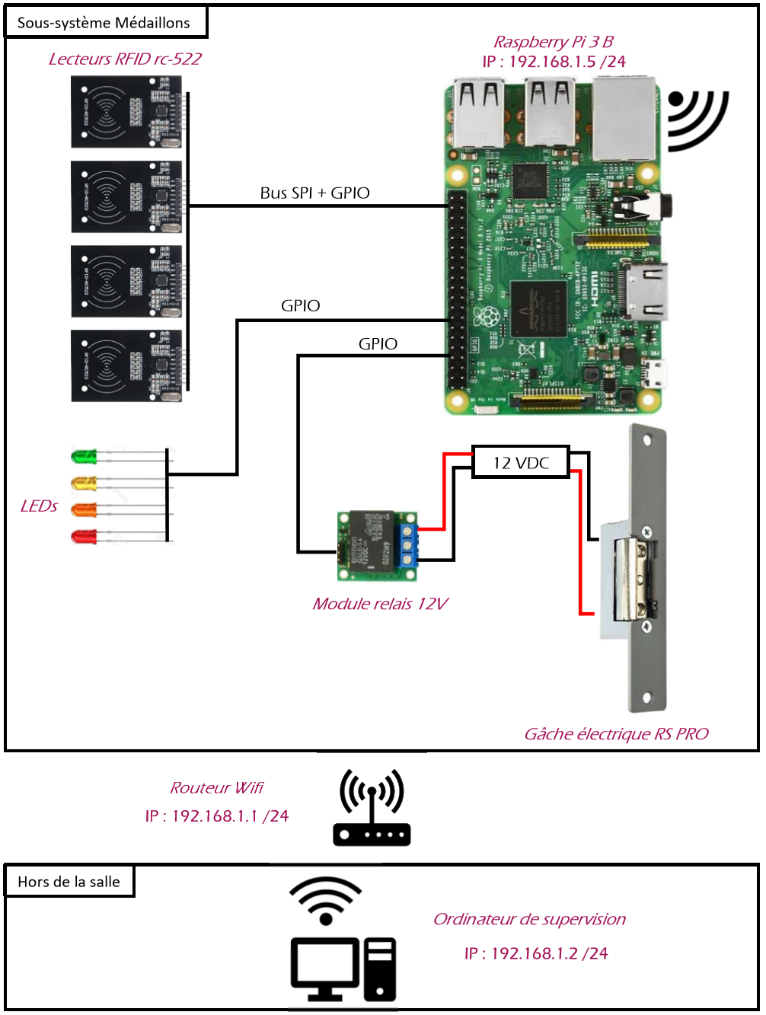
* + La détections des « médaillons ».
  + L’allumage des différentes LED.
  + L’ouverture de la porte.
  + La communication entre système « médaillons » et l’application de supervision

Le langage python est utilisé avec différentes librairies, comme « pi-RC522 » qui permet la gestion des lecteurs RFID, « socket » qui permet l’utilisation des services socket. Comme IDE j’ai choisi « Thonny » car très simple d’utilisation, ce choix m’était libre.

Le système doit être développé sur une carte Raspberry sous Raspbian. Le contrôle du superviseur de cette dernière sera possible via l’application « Anydesk » qui est compatible avec les deux environnement, Raspbian et Windows. Elle permet le contrôle à distance via un protocole TCP.

## Communication entre les différents éléments

***Synoptique***



L’ordinateur de supervision, le routeur wifi ainsi que Raspberry sont adressés en IP statiques

## Communication

Le bus SPI (Serial Peripheral Interface) est un bus de données série synchrone en full-duplex. Il fonctionne sous un schéma maître-esclave. Dans ce cas le maître est la Raspberry et les esclaves sont les lecteurs RFID RC522.

Ce modèle possède 8 interfaces dont 4 qui sont dédiés au bus SPI :

* SCK – Consacrer à l’horloge et généré par le maître.
* MOSI – Master Output, Slave Input (généré par le maître).
* MISO – Master Input, Slave Output (généré par l’esclave).
* NSS – La sélection de l’esclave (généré par l’esclave).

Les 4 interfaces restante sont :

* VCC – Alimente le module avec un voltage de 3,3 Volt.
* GND – La terre.
* RST – Permet la réinitialisation et la mise hors tension du module.
* IRQ – Permet une interruption, cela alerte le module quand un tag RFID se trouve à proximité

## Câblage et choix des PIN

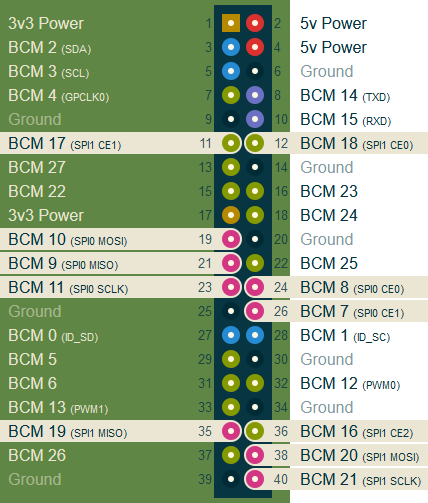
Cette partie du projet demande un câblage plus ou moins complexe, une recherche et des choix ont donc été nécessaire.

Le modèle du lecteur RFID RC522 a besoin de 8 PIN pour pouvoir fonctionner. Le bus spi nécessite 4 d’entre eux, le MOSI, le MISO, le SCK qui peuvent être commun pour tous les lecteurs et enfin le NSS (CE) qui est unique pour chacun d’entre eux, ces PIN on des emplacements dédiés sur la Raspberry. Les 4 interfaces restante peuvent être commune pour tous les lecteurs, une alimentation

Pour les LEDs un port GPIO (General Purpose Input/Output) est nécessaires pour chacune d’entre elles ainsi qu’une résistance de 90Ω dû voltage des ports GPIO de 3,3V. Elles ont cependant une terre commune.

Le module relais a besoin quant à lui d’un port GPIO pour le contrôler, un port d’alimentation de 5V et enfin une terre.

Un schéma préliminaire du câblage est disponible en annexe.



GPIO

SPI

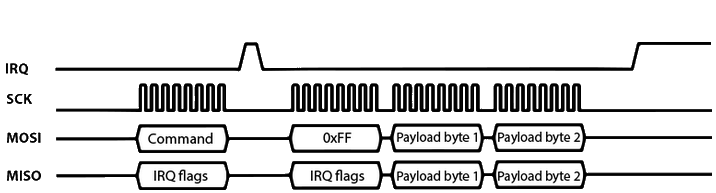
Terre

5V

3,3V

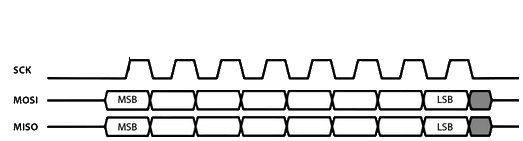
## Chronogrammes

Dans le premier chronogramme on voit que quand un badge passe à proximité donc quand l’IRQ signal le passage d’un tag RFID. La première transaction qui se passe avant le signal de l’IRQ, c’est l’octet (8 bites) de commande. Dans la second transaction 24 bits son transmis, les 16 derniers bits transmis sont l’UID (User Identifier) qui est propre à chaque taf RFID.



Chronogramme

Dans le second chronogramme on voit comment se passe la transaction d’un octet, le bite de poids fort (MSB) est transmis en premier.



Chronogramme

## Système socket

Pour envoyer l’ordre de fin partie un système de socket a été choisi avec comme « client » l’application du sous-système médaillons et comme serveur l’application de supervision. Ainsi le serveur sera en « écoute » en attendant que le client envoie un message qui confirmera la réussite de l’équipe qui participera à la session de jeu de l’escape game.

Pour ce faire les informations suivantes sont requises :

Pour le client :

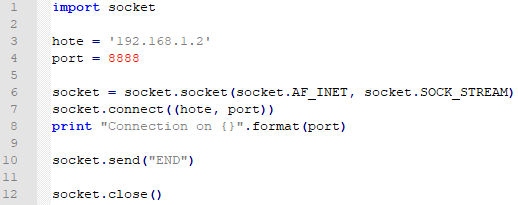
* Adresse IP : **192.168.1.2** (adresse IP l’ordinateur de supervision)
* Port : **8888**

Pour le serveur :

* Adresse IP : **192.168.1.5** (adresse IP de la Raspberry)
* Port : **8888**

## Code du client

Afin que l’application puisse envoyer l’ordre de fin de partie qui un simple message « END », il faut utiliser le code suivant en python :



La méthode *import*, ligne 1, permet d’accéder à différents modules, dans ce cas on importe le module socket.

Par la suite on définit l’hôte et le port de communication. On crée ensuite le socket avec la méthode *socket(socket.AF\_INET, socket.SOCK\_STREAM)*, ligne 6, puis on envoie une requête de connexion au serveur avec la méthode *socket.connect()*, ligne 7. Si la connexion n’est pas rejeté on envoie le message de fin avec la méthode *socket.send()*, ligne 10, puis on ferme le socket avec *socket.close()*, ligne 12.

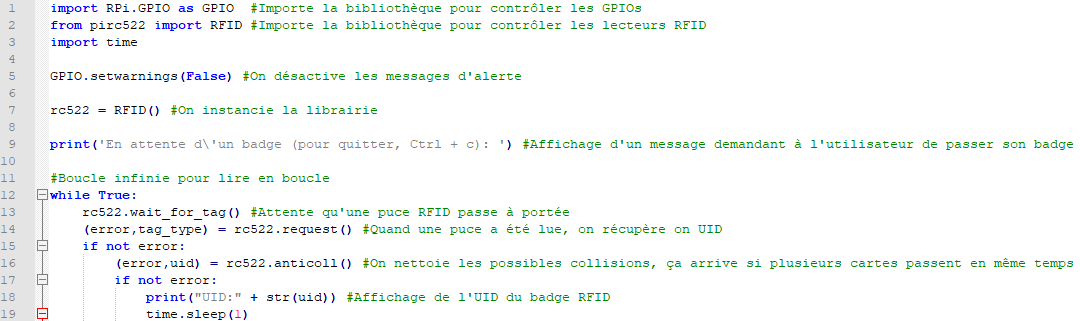
# Réalisation de l’application :

Afin que la partie la partie se déroule, il est nécessaire d’avoir une application qui permettra de détecter les médaillons, allumer les LEDs, déclencher le module relais et enfin envoyer l’ordre de fin partie comme si les médaillons sont bien placer.

## Récupération de l’UID

En premier lieu il faut pour le superviseur pouvoir récupérer l’UID d’un badge RFID dans le cas d’une perte et/ou vol.

Pour ce faire le code suivant est utilisé.

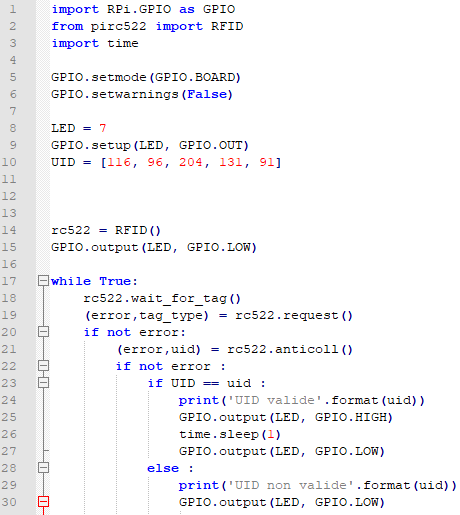


Dans un premier temps importe le module RFID depuis la librairie « pi-rc522 » et on instancie la librairie avec *rc522 = RFID()*, ligne 2.

Ensuite on crée une boucle infinie pour lire en boucle. Dans cette boucle on utilise la fonction *wait\_for\_tag()*, ligne 13, pour attendre qu’un badge RFID passe à portée de lecture. Quand un badge passe à proximité on récupère son UID avec *request()*, ligne 14 et on nettoie les possibles collisions avec *anticoll()*, ligne 16, ça arrive si plusieurs badge passent à porter en même temps. Puis un on affiche UID unique du badge avec *str()*, ligne 18. Pour la fin on attend 1 pour ne pas lire le badge des centaines de fois en quelques milli-secondes avec *sleep(1)*, ligne 19, 1 étant 1 seconde.

## Allumage des LEDs et déclanchement du module relais

L’allumage de la LED doit pouvoir se faire dans la seconde en fonction de l’état du capteur donc si un badge RFID est proximité ou non. Pour cela le code suivant est utilisé.



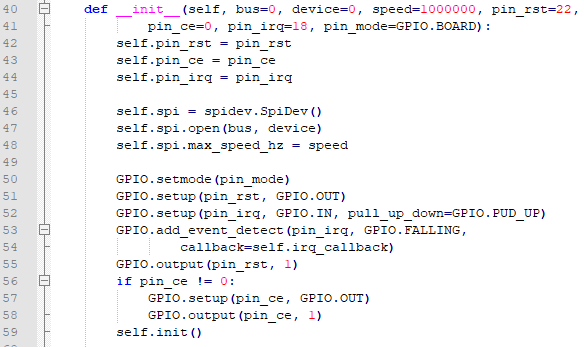
Une fois qu’un UID a été récupérer, ligne 10, on peut l’utiliser pour faire un comparatif avec l’UID récupérer par le lecteur RFID et ainsi allumer la LED. Pour ce faire une défini le numéro du port GPIO qui va contrôler la LED, ici port n°7 et on active le contrôle du GPIO avec *setup(LED,GPIO.OUT)*, ligne 9, puis on initialise ce port à l’état bas (LED éteinte) avec *output(LED,GPIO.LOW)*, ligne 15. Ensuite on reprend la boucle d’écoute décrit précédemment et on fait la comparaison des deux UID. Si l’UID est valide on allume la LED avec *output(LED,GPIO.HIGH)*, ligne 25, puis on attend 1 seconde et on éteint la LED. Si le badge reste à porter du lecteur RFID la LED restera allumer car la boucle est infinie.

Pour le déclanchement du module relais on utilise sensiblement la même démarche, seul le temps d’attente est différent pour permettre aux joueurs d’avoir un délai suffisant pour sortir de la salle.

## Bibliothèque RFID

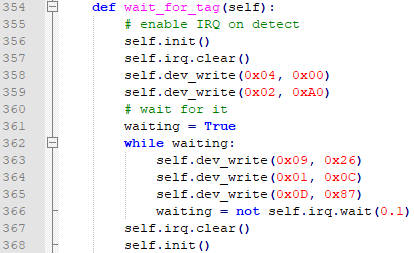
***Constructeur***

La bibliothèque principale utilisé pour le contrôle des lecteurs RFID est « pi-rc522 », cette libraire est basée sur une autre librairie, la « MFRC522 » qui n’est malheureusement plus à jour.



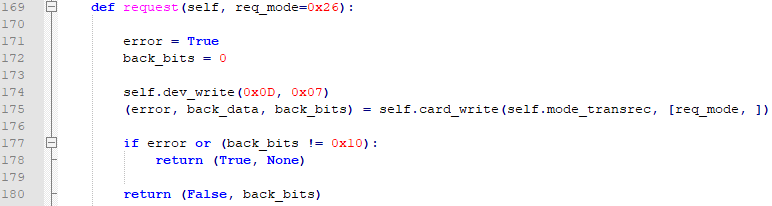
Sur la ligne 40 on crée notre constructeur avec *\_\_init\_\_*, *self* en premier argument permet d’avoir des méthodes non statiques, on peut donc les changer pour l’adapter à notre câblage. Le *bus=0* définit si on utilise le bus SPI 1 ou 2 car un Raspberry peu en gérer deux. Le *device=0* définit le pin de contrôle du lecteur RFID (NSS). Grace à ce constructeur on peu choisir quel lecteur on veut contrôler.

***Fonction wait\_for\_tag***



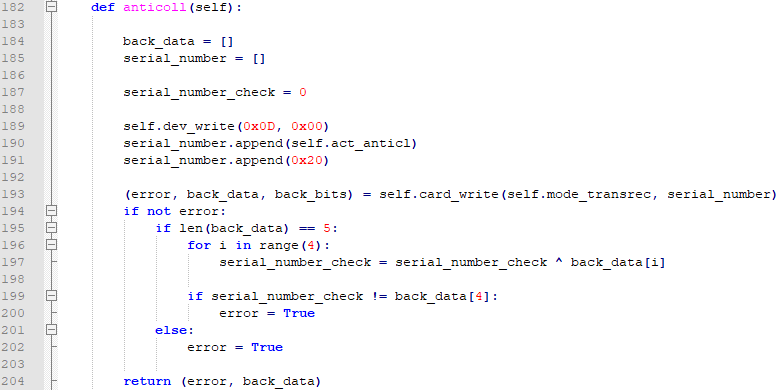
Cette fonction qu’on crée avec *def* permet l’utilisation de l’IRQ qui permet de détecter si un tag RFID passe à proximité.

***Fonction request***



La fonction

***Fonction anticoll***

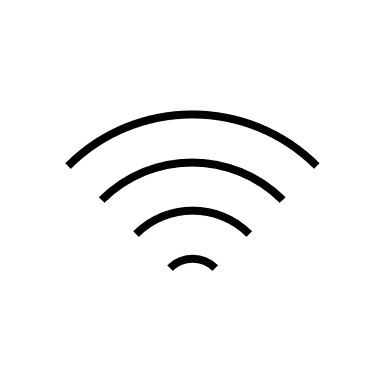


# Technologie RFID

La technologie RFID (Radio Frequency Identification) permet de récupérer des données à distance de différents « tag RFID » en utilisant un champ magnétique.

Pour cela le principe suivant est utilisé. Le système de radio-identification est composé de deux parties qui communique l’une avec l’autre. L’échange des données débute par un transfert d’énergie électromagnétique entre le tag et l’émetteur ou lecteur RFID.

Signal





Tag RFID

Energie

Lecteur RFID

Signal

Le modèle RC522 est un lecteur dit à haute fréquence, il fonctionne avec une longueur d’onde de 13,56 MHz qui a un déploiement difficile à travers les environnements métalliques.

Le tag est dit « passif » de classe 2, il permet la lecture et l’écriture de la mémoire, cette mémoire correspond à l’UID unique composé entre 7 à 14 caractères. Il répond à la norme ISO 14443.

Cette norme décrit plusieurs couches, 4 au total. La première couche a pour dénomination « Physical Layer », elle décrit la partie physique du tag. La deuxième couche, « Radio frequence signal interface », traite de la fréquence à laquelle l’émetteur et le récepteur doivent fonctionner, ici 13,56MHz ainsi que le débit de transfert, ici 25kb/s. La troisième couche, « Activation and anti-collision » qui décrit le principe d’anti-collisions qui permet de gérer le cas de plusieurs récepteurs qui se trouveraient en même temps dans le champ magnétique, d’où la fonction *anticoll()* décrit précédemment. La quatrième et dernière couche est appelé « Transmission protocol », on y décrit le format des données, la communication durant une transaction.

# Fiches recettes

# Conclusion

## Communication avec le groupe

Dès le début du projet nous avons mis en place différents moyen pour garder une cohésion de groupe, cette cohésion était d’autant plus importante pendant la période de confinement. Pour cela, plusieurs outils et méthode ont été utilisés :

* Gantt
* GitHub/GitKraken
* Charte graphique Word+power point
* Journal d’activités
* Teams
* Différentes messageries instantanées
* Réunions de groupe toutes les semaines avec Mr.Angibaud
* Office 365

## Difficultés liées au confinement dû au COVID-19

A partir de Mars 2020 nous confiné…

## Connaissances acquises

J’ai pu apprendre le langage python car nous n’avions jusque là jamais codé avec ce langage.

Développement sous Raspbian avec une Raspberry

## Ce qui reste à faire

Au moment de la rédaction de ce rapport, il me reste à faire :

* Mettre en place plusieurs lecteurs RFID en simultané
* La connexion entre la Raspberry et l’application de supervision via un système socket client/serveur
* Un test en grandeur nature

## Poursuite d’étude

Accepter à l’ENI pour la formation « Administrateur Système et Réseau » en alternance dans l’entreprise « TIGS » (Technique Informatique Global pour la Santé) pour une durée de 2 ans avec une possibilité de continué à l’ENI avec la formation « Expert en Sécurité Digitale ».

# Annexes

## Annexe 1

