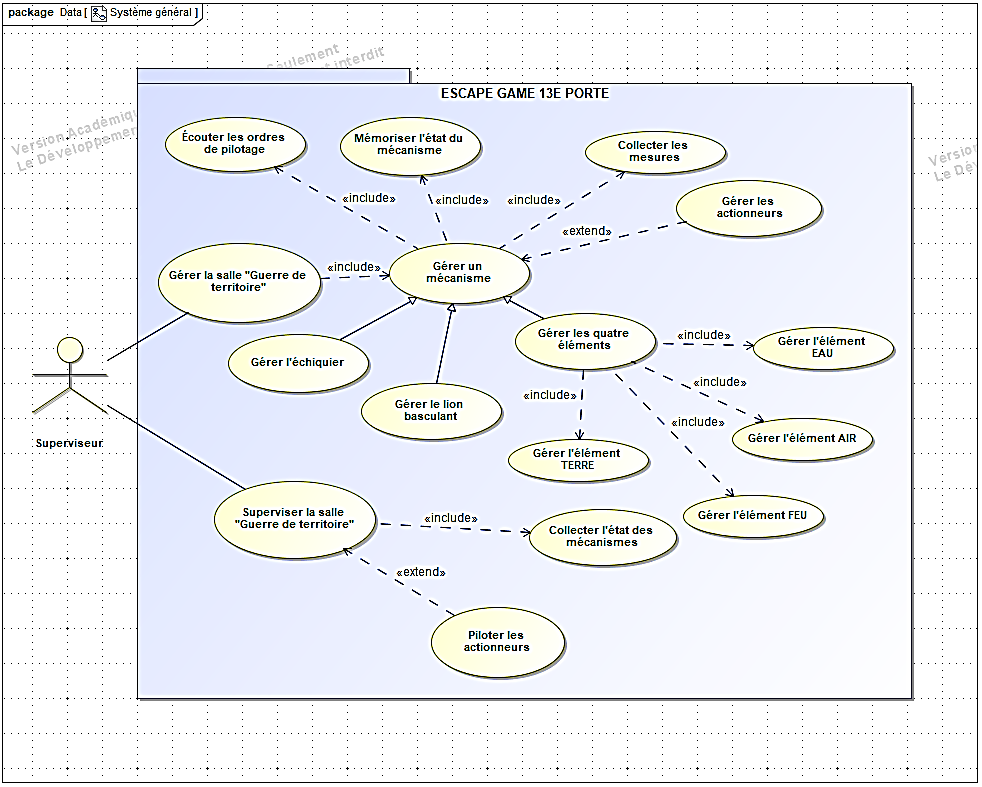
Rapport

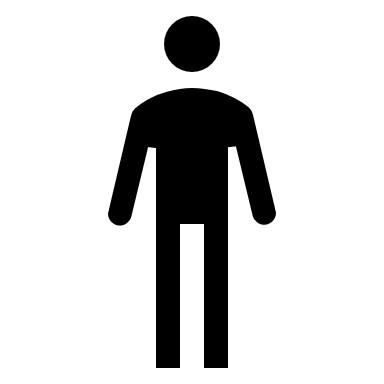
***Présentation du projet***

La société *La 13EME PORTE* est une enseigne d’Escape Game. Les gérants de la société souhaitent que le système technique d’une salle actuellement en place soit recréé entièrement afin de corriger les erreurs de conception et les différents bugs existants. De plus il est demandé d’ajouter plusieurs fonctionnalités permettant d’améliorer le travail du superviseur.  
  
Le système de la salle s’appuie sur 9 mécanismes et il devra associer à chacun des mécanismes un Arduino dédié.

Un mécanisme pourra collecter des mesures par l’intermédiaire d’actionneur (capteurs, boutons, etc…) placé sur la carte Arduino et devra écouter les ordres de pilotage.  
De plus chaque mécanisme transmettra régulièrement par liaison I2C à la Raspberry les valeurs de ses différentes entrées.   
Chaque information reçue par la Raspberry sera enregistrée en base de données et installée sur un serveur positionné dans le bâtiment par l’intermédiaire du poste de supervision.  
Sur ce poste de supervision on pourra visualiser l’état de la salle et piloter les actionneurs.

***Diagramme de cas d’utilisation :***

******



***Diagramme d’exigences :***



***Répartition des tâches :***

* Etudiant 1 : Constantin
* Gérer le Mécanisme 4
* Gérer le Mécanisme 8
* Gestion du bus I2c entre Arduino et Raspberry :

Envoie des ordres de pilotage de la Raspberry à l’Arduino et les traiter

Envoi des mesures et des états des actionneurs sur la Raspberry et les traiter

* Etudiant 2 : Corentin
* Créer la base de données
* Installer les serveurs Apache et MySql
* Gérer le Mécanisme 6
* Gérer le Mécanisme 7
* Récupérer la valeur des capteurs et l’état des actionneurs de la Raspberry au serveur
* Etudiant 3 : Joshua
* Configuration réseau du matériel
* Création d’une application Web sur l’affichage de l’état de la salle
* Gérer le Mécanisme 1
* Gérer le Mécanisme 2
* Gérer le Mécanisme 3
* Etudiant 4 : Thomas
* Création d’une application Web de pilotage des actionneurs
* Gérer le Mécanisme 5
* Gérer le Mécanisme 9
* Envoie des ordres de pilotage du PC de supervision à la Raspberry



**Application Web de pilotage des actionneurs**

**Application Web sur l’affichage de l’état de la salle**

Constantin

Thomas

Joshua

Corentin

Constantin

Thomas

Joshua

Corentin

***Matériels utilisés***



* Raspberry Pi 3B :

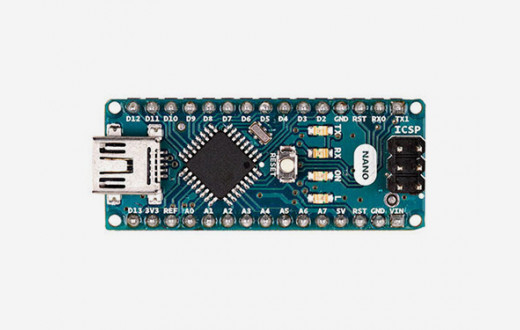
**Programmation sur Raspberry :**

Langage : Python

Nous utiliserons l’émulateur de Terminal PuTTY afin de pouvoir programmer directement depuis notre PC sur la Raspberry

* Arduino :

**Matériel Arduino pour la salle de jeu**

* Neuf Arduino Nano
* Quatre capteurs à effet hall A3144
* Module de commutateur de clé 3 pin 1NO1NC
* Afficheur 7 segments
* Capteur d’humidité
* Interrupteur à bascule
* Sept capteurs photosensibles
* Boutons poussoir
* Leds
* Capteur de fin de course
* Capteur de poids HX711

**Programmation sur Arduino :**

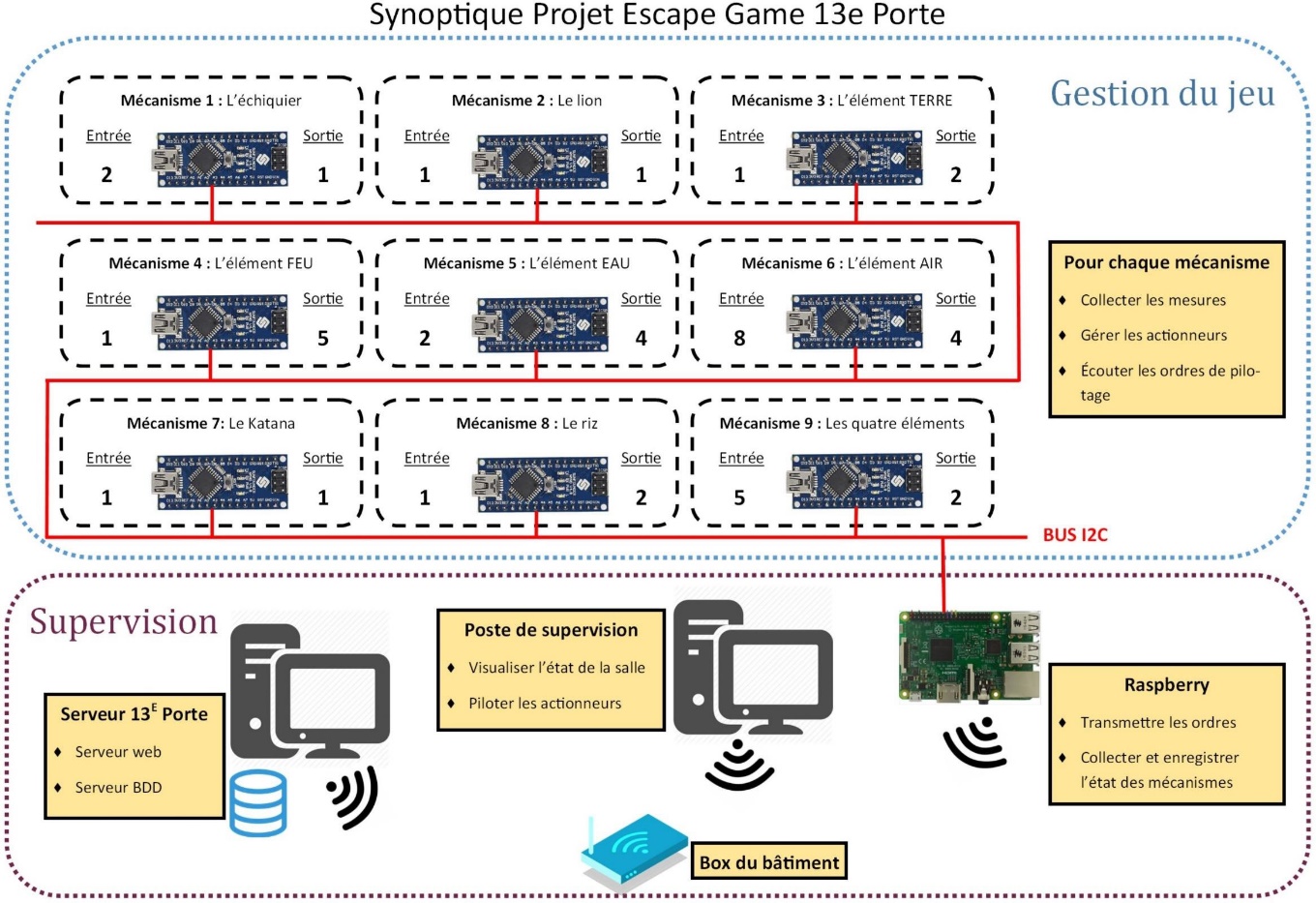
Langage : Arduino

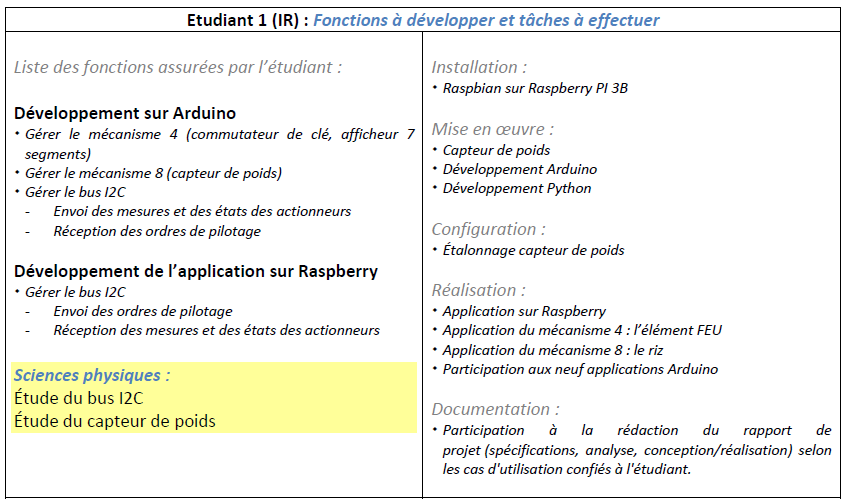
IDE : Arduino

• Un poste informatique pour le serveur *13e Porte*

• Un poste informatique de supervision (éventuellement le même poste que le serveur)

***Ma tâche***





***Procédure Raspberry***

Partie Hardware

1) Formater la carte micro-SD (SD Card Formatter)  
lien : <https://www.sdcard.org/downloads/formatter/>

2) Télécharger l’image système Raspbian (Version Desktop de préférence)  
lien : <https://www.raspberrypi.org/downloads/raspbian/>

3) Ecrire l’image Système sur la carte SD :

* Installer de Win32DiskImager.exe et lancer  
  lien : <https://win32diskimager.download/>
* Choisir l’image système Raspbian et la carte SD
* Write

4) Insérer la carte Micro-SD dans la Raspberry + réaliser le câblage I2C

5) Alimenter la Raspberry

Partie Software

1) Se connecter :

* Nom d’utilisateur par défaut : **pi**
* Mot de passe par défaut : **raspberry** (Attention clavier QWERTY raspberry = rqspberry)

2) Changer le clavier en AZERTY :

* Aller dans l’invité de commande
* Ecrire la commande **sudo dpkg-reconfigure keyboard-configuration**
* Conserver le choix par défaut « Generic 105-Key (Intl) PC » puis « Ok »
* Sélectionner le choix « French » puis « Ok »
* Sélectionner le clavier « The default for the keyboard layout » puis « Ok »
* Sélectionner le choix « No compose key » puis « Ok »
* Terminer la configuration par « Finish »
* Ecrire la commande **sudo shutdown -r now** (Pour redémarrer de la Raspberry)

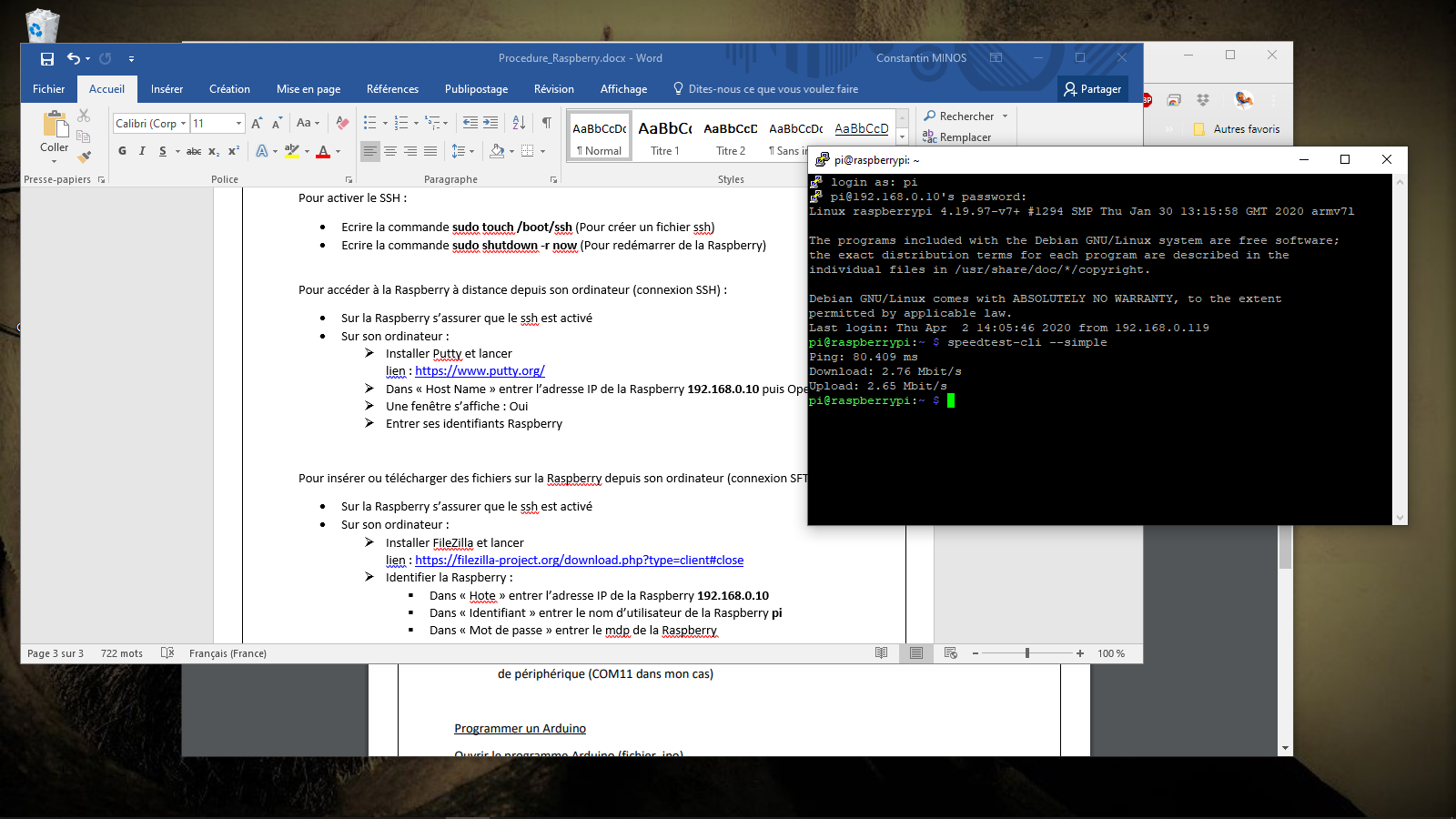
3) Changer le mot de passe :

* Aller dans l’invité de commande
* Ecrire la commande **sudo raspi-config**
* Sélectionner « Change User Password » puis « Ok »
* Choisir son nouveau mot de passe puis « Ok »

4) Se connecter au Wifi :

* Ecrire la commande **sudo raspi-config**
* Sélectionner « Network Option » puis « Wi-fi »
* Entrer le SSID (nom du réseau) puis le mdp de la Wifi
* Sélectionner « Localisation Options » puis « Change Wifi Country »
* Dans la liste choisir « France » puis « Ok »
* Terminer la configuration par « Finish »

*Optionnel :* Pour vérifier la connexion Internet et le débit (speedtest) :

* Ecrire la commande **sudo apt-get install python3-pip** (Pour installer pip)
* Ecrire la commande **sudo pip3 install speedtest-cli** (Pour installer speedtest)
* Ecrire la commande **speedtest-cli --simple** (Pour lancer speedtest)  
  

5) Mettre à jour la Raspberry :

* Ecrire la commande **sudo apt-get update**
* Ecrire la commande **sudo apt-get upgrade** puis **y** pour valider

6) Créer une IP fixe (@192.168.0.10) :

* Ecrire la commande **sudo nano /etc/dhcpcd.conf** (Pour modifier le fichier dhcpcd.conf)
* Supprimer tout son contenu et le remplacer par :

Une image contenant capture d’écran

Description générée automatiquement

CTRL + O pour sauvegarder, ENTRER pour valider puis CTRL + X pour quitter

* Ecrire la commande **sudo shutdown -r now** (Pour redémarrer de la Raspberry)
* Ecrire la commande **ifconfig** (Pour vérifier les changements) :

Une image contenant capture d’écran, ordinateur

Description générée automatiquement

7) Activer l’i2c :

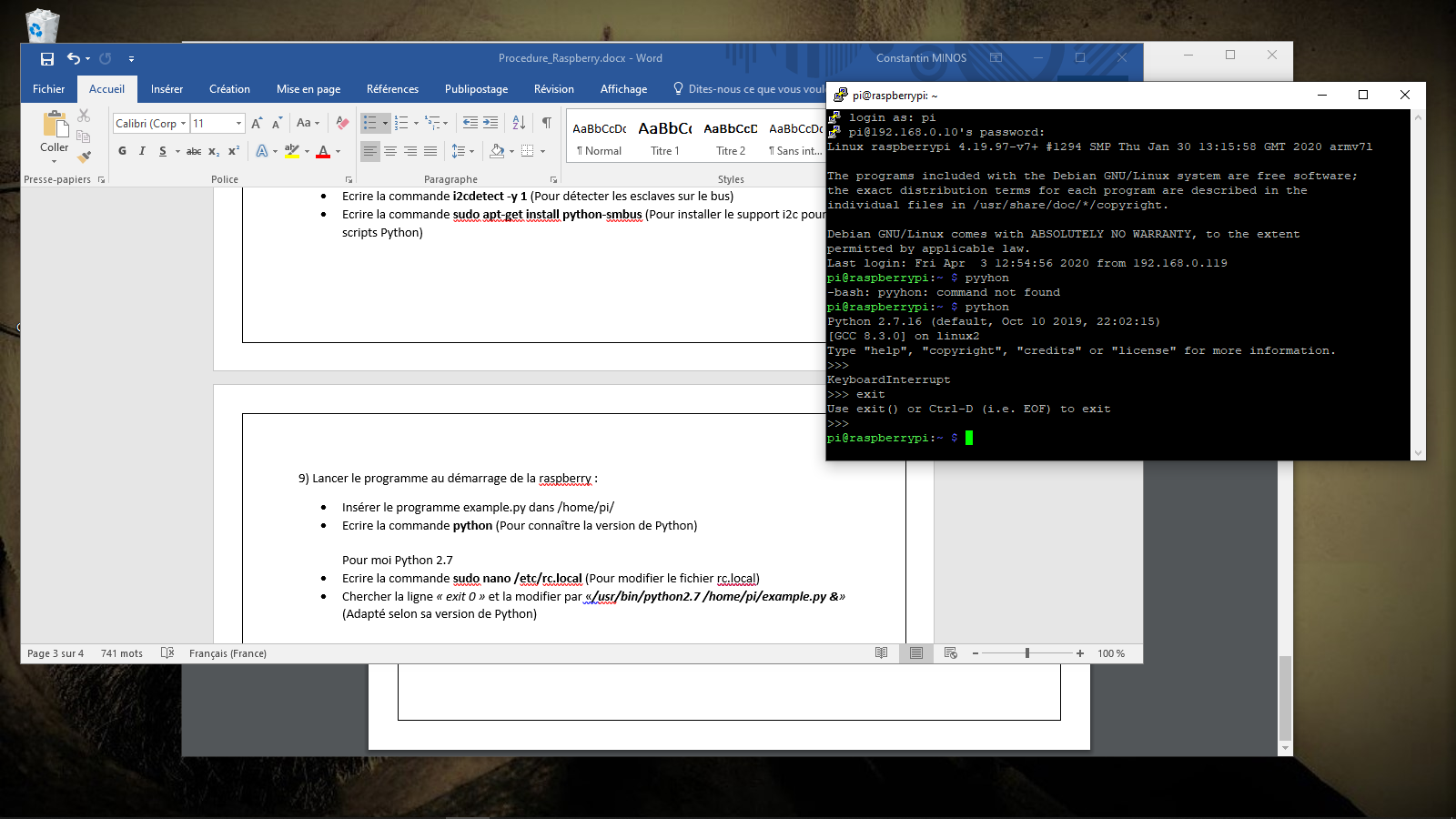
* Ecrire la commande **sudo raspi-config**
* Sélectionner « Interfacing Options »
* Sélectionner l’option (P5) I2C puis « Yes » pour activer
* Ecrire la commande **sudo apt-get install i2c-tools** (Pour installer les outils de gestion i2c)
* Ecrire la commande **sudo shutdown -r now** (Pour redémarrer de la Raspberry)
* Ecrire la commande **ls -l /dev/i2c\*** (Permet de vérifier l’activation i2c et identifier les bus i2c)

8) Communiquer avec l’Arduino :

* Ecrire la commande **i2cdetect -y 1** (Pour détecter les esclaves sur le bus)
* Ecrire la commande **sudo apt-get install python-smbus** (Pour installer le support i2c pour scripts Python)

9) Lancer le programme « i2c.py » au démarrage de la Raspberry :

* Insérer le programme « i2c.py » dans /home/pi/
* Ecrire la commande **python** (Pour connaître la version de Python) puis CTRL+D (Pour quitter)



Pour moi Python 2.7

* Ecrire la commande **sudo nano /etc/rc.local** (Pour modifier le fichier rc.local)
* Chercher la ligne *« exit 0 »* et la modifier par «***/usr/bin/python2.7 /home/pi/i2c.py &****»* (Adapté selon sa version de Python)

*Optionnel :*

Pour activer le SSH :

* Ecrire la commande **sudo touch /boot/ssh** (Pour créer un fichier ssh)
* Ecrire la commande **sudo shutdown -r now** (Pour redémarrer de la Raspberry)

Pour accéder à l’invité de commande de la Raspberry depuis son ordinateur (connexion SSH) :

* Avoir activé le ssh sur la Raspberry
* Sur son ordinateur :
* Installer Putty et lancer

lien : <https://www.putty.org/>

* Dans « Host Name » entrer l’adresse IP de la Raspberry **192.168.0.10** puis Open
* Une fenêtre s’affiche : Oui
* Entrer ses identifiants Raspberry

Pour insérer ou télécharger des fichiers sur la Raspberry depuis son ordinateur (connexion SFTP) :

* Avoir activé le ssh sur la Raspberry
* Sur son ordinateur :
* Installer FileZilla et lancer

lien : <https://filezilla-project.org/download.php?type=client#close>

* Identifier la Raspberry :
* Dans « Hote » entrer l’adresse IP de la Raspberry **192.168.0.10**
* Dans « Identifiant » entrer le nom d’utilisateur de la Raspberry **pi**
* Dans « Mot de passe » entrer le mdp de la Raspberry
* Dans « Ports » entrer **22**
* Connexion Rapide

***Liaison I2c***

Pour lier les Arduino à la Raspberry nous allons passer par le protocole i2C :   
- pour envoyer l’état des actionneurs et les valeurs des capteurs des Arduino à la Raspberry   
- pour envoyer les ordres de pilotage du PC de supervision de la Raspberry aux Arduino.

Le protocole I2C permet à plusieurs composants de dialoguer entre eux de manière bidirectionnelle mais en half-duplex uniquement. Et grâce à seulement trois fils : Un signal de donnée (SDA), un signal d’horloge (SCL), et un signal de référence électrique (Masse).

Les données sont transmises en série à 100Kbits/s en mode standard et jusqu’à 400Kbits/s en mode rapide.

Chaque périphérique sur le bus I2C est adressable, avec une adresse unique pour chaque périphérique du bus

Il y a un principe de maitre et esclave. Ici nous utiliseront le Raspberry Pi en tant que maitre et les Arduino en tant qu’esclave.   
Le maitre (la Raspberry) est le composant qui initialise un transfert, génère le signal d’horloge et termine le transfert. Dans notre cas il sera récepteur et émetteur.  
Les esclaves (Les Arduinos) sont les composants adressés par un maître. Dans notre cas ils seront récepteurs et émetteurs.



Script Python sur Raspberry :

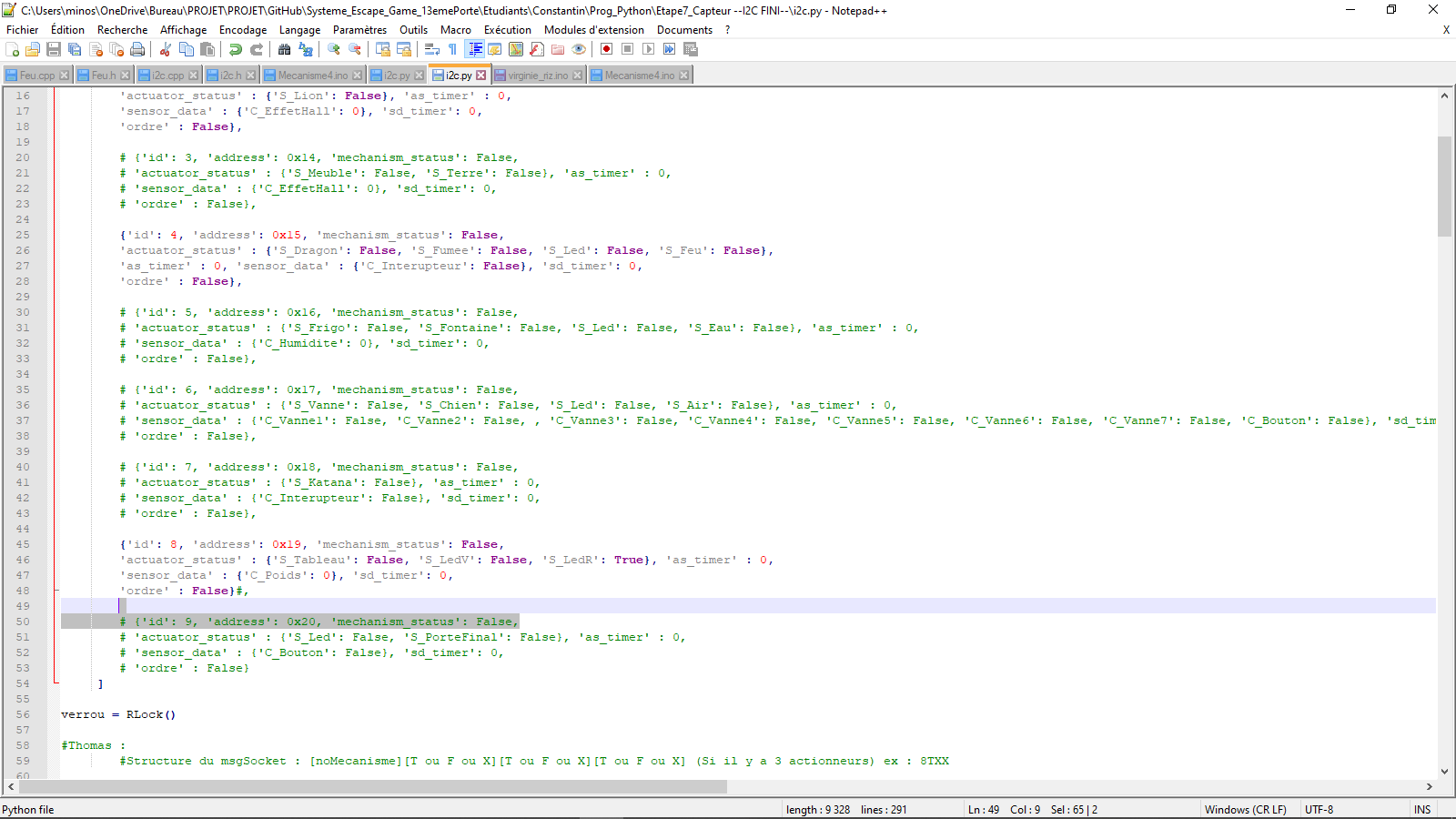
Ce script python consiste à :

* Recevoir la valeur des capteurs de chaque mécanisme à chaque changement ou sinon toutes les 5s max. Et ensuite envoyer les valeurs à la BDD.
* Recevoir l’état des actionneurs de chaque mécanisme à chaque changement ou sinon toutes les 60s. Et ensuite envoyer les valeurs à la BDD.
* Recevoir l’état de chaque mécanisme à chaque changement ou sinon toutes les 60s. Et ensuite envoyer les valeurs à la BDD.
* Envoyer les ordres reçus depuis l’application Web au mécanisme correspondant

Il a fallu créer une liste de 9 dictionnaires correspondant aux 9 Arduinos. Les clés de ces dictionnaires sont identiques.

Voici un fragment de la liste de dictionnaires :

Une image contenant capture d’écran

Description générée automatiquement  
 (…) // 7 autres dictionnaires  


Chaque dictionnaire Arduino possède les clés suivantes :

* « **id** » : correspond au numéro du mécanisme assigné à l’Arduino
* « **address** » : correspond à l’adresse esclave assigné à l’Arduino pour l’i2c
* « **mechanism\_status** » : correspond à l’état du mécanisme (valeur True/False)
* « **ms\_noTimer**» : correspond à une valeur entière

Nécessaire à l’exécution du thread MSTimer pour l’envoie des valeurs actionneurs toutes les 60s

* « **actuator\_status** » : correspond à l’état des actionneurs du mécanisme

C’est une liste des différents actionneurs (ayant chacun une valeur True/False).

* « **as\_noTimer**» : correspond à une valeur entière

Nécessaire à l’exécution du thread ASTimer pour l’envoie des valeurs actionneurs toutes les 60s

* « **sensor\_data** » : correspond à l’état des capteurs du mécanisme

C’est une liste des différents capteur (ayant chacun une valeur entière ou True/False).

* « **sd\_noTimer**» : correspond à une valeur entière

Nécessaire à l’exécution du thread SDTimer pour l’envoie des valeurs capteurs toute les 4s.

* « **order** » : correspond à une valeur entière, à l’ordre envoyé par le PC de supervision.

Ce sont ces dictionnaires qui sont modifiés à chaque changement dans les mécanismes.

Afin de recevoir les informations des mécanismes en temps réel il a fallu utiliser des Threads. De sorte que toutes les Arduinos communique avec la Raspberry en même temps.  
Pour cela il a d’abord fallu créer le thread « **ArduinoCom** » qui correspond à la communication de la Raspberry avec une Arduino ; ensuite il a fallu assigner à chaque Arduino un thread ArduinoCom.

Le thread ArduinoCom consiste à effectuer en boucle les instructions suivantes :

Une image contenant capture d’écran

Description générée automatiquement

**1. Si un ordre a été donné depuis l’application Web l’envoyer à l’Arduino**

On appelle la fonction send\_order().

Elle exécute principalement les instructions suivantes :

Une image contenant capture d’écran

Description générée automatiquement

**2. Récupérer le message i2c depuis l’Arduino**

On appelle la fonction get\_message().

Elle exécute les instructions suivantes :

Une image contenant capture d’écran

Description générée automatiquement

**3. Gérer l’état des mécanismes**

On appelle la fonction get\_mechanism\_status().

Elle exécute principalement les instructions suivantes :

**Une image contenant capture d’écran

Description générée automatiquement**

S’il n’ y a pas de changement d’état pendant 60 secondes d’après le diagramme d’exigence il faut envoyer l’état en BDD.   
Pour ce faire il était judicieux de faire fonctionner un timer de 60s en parallèle. Il a donc fallu utiliser une nouvelle fois des threads.  
Ainsi chaque Arduino possède son propre thread qui correspond au timer de l’état mécanisme. Ce thread est appelé «**MSTimer** »

Le thread MSTimer exécute principalement les instructions suivantes :

Une image contenant capture d’écran

Description générée automatiquement

**4. Gérer l’état des actionneurs**

On appelle la fonction get\_actuator\_status().

Elle exécute principalement les instructions suivantes :

Une image contenant capture d’écran

Description générée automatiquement

De même que pour l’état d’un mécanisme S’il n’ y a pas de changement d’état pendant 60 secondes d’après le diagramme d’exigence il faut envoyer l’état en BDD.   
Pour ce faire il était judicieux de faire fonctionner un timer de 60s en parallèle. Il a donc fallu utiliser une nouvelle fois des threads.  
Ainsi chaque Arduino possède son propre thread qui correspond au timer de l’état des actionneurs. Ce thread est appelé «**ASTimer** »

Le thread ASTimer exécute principalement les instructions suivantes :

Une image contenant capture d’écran

Description générée automatiquement

**5. Gérer la valeur des capteurs**

On appelle la fonction get\_sensor\_data().

Elle exécute principalement les instructions suivantes :

Une image contenant capture d’écran

Description générée automatiquement

De même que pour l’état d’un mécanisme S’il n’ y a pas de changement d’état pendant 5 secondes max d’après le diagramme d’exigence il faut envoyer l’état en BDD.   
Pour ce faire il était judicieux de faire fonctionner un timer de 4s en parallèle. Il a donc fallu utiliser une nouvelle fois des threads.  
Ainsi chaque Arduino possède son propre thread qui correspond au timer de la valeur des capteurs. Ce thread est appelé «**SDTimer** »

Le thread SDTimer exécute principalement les instructions suivantes :

Une image contenant capture d’écran

Description générée automatiquement

Programme i2c sur Arduino :

La partie i2c du programme de chaque Arduino consiste à :

* Envoyer les informations du mécanisme à la Raspberry :

Une image contenant capture d’écran, moniteur, écran, ordinateur

Description générée automatiquement

* Exécuter les ordres reçus :

Une image contenant capture d’écran, moniteur, ordinateur, portable

Description générée automatiquement

***Description des mécanismes***

* Mécanisme 4 :

Gestion de la clé (Elément FEU)

Description du Sous-système : Un interrupteur à clef est caché dans le décor, les joueurs doivent trouver la clef et la tourner dans l’interrupteur

Résultat :

1. Une LED témoin s’allume sur le tableau de contrôle
2. Une « Tête de Dragon » sort du plafond (S2). En effet, la désactivation de l’électroaimant a pour effet de libérer une trappe au plafond
3. Une machine à fumée (220 Volt) est allumée via un relais. Cette machine à fumée est dissimilée dans la « tête de dragon » afin que ce dernier crache de la fumée lorsqu’il apparait aux joueurs
4. 5 LED blanches s’allument afin au-dessus afin d’éclairer cette dernière
5. L’élément FEU (LED) s’allume sur la tablette à destination des joueurs

## Entrées :

* C\_Interupteur
  + Type de capteur : Interrupteur à clef
  + Référence : Commutateur de clé 3 pin 1NO1NC
  + Signal : numérique
  + Voltage : 0-5V

## Sorties :

* S\_Dragon
  + Rôle : Activer/désactiver un électroaimant via un relais + LED de contrôle
  + Condition : Si E1 est à 1 alors activer la sortie, sinon, désactiver la sortie
  + Voltage : 0-5V
* S\_Fumee
  + Rôle : Activer/désactiver une machine à fumée (220Volt) via un relais
  + Condition : Si E1 est à 1 alors activer la sortie, sinon, désactiver la sortie
  + Voltage : 0-5V
* S\_Led
  + Rôle : Activer/désactiver 5 LED blanche câblées en parallèles
  + Condition : Si E1 est à 1 alors activer la sortie, sinon, désactiver la sortie
  + Voltage : 0-5V (5 Volts + résistance de 220 ohms)
* S\_Feu
  + Rôle : entrée de la gestion des quatre éléments

Programme sur Arduino :

Le programme exécute principalement :  
Une image contenant capture d’écran, ordinateur

Description générée automatiquement  
Une image contenant capture d’écran, ordinateur

Description générée automatiquement  
Une image contenant capture d’écran, ordinateur

Description générée automatiquement

On instancie d’abord un objet Feu qu’on nomme « mecanism ». On lui donne une configuration de base dans setupMechanism() puis on appelle en boucle la méthode « **execute()** » de la classe Feu.

Cette méthode exécute principalement les instructions suivantes :

Une image contenant capture d’écran, ordinateur

Description générée automatiquement  
Une image contenant capture d’écran, ordinateur, portable

Description générée automatiquement

* Mécanisme 8 :

# Gestion de la balance de riz et du tableau

Description du Sous-système : les joueurs doivent faire « une offrande » de 50 grammes de riz sur coupelle posée sur un capteur de poids. La condition est validée avec une précision de +/- 2 grammes.

Résultat :

1. La LED rouge s’éteint et la LED verte s’allume
2. Une LED témoin s’allume au panneau de contrôle.
3. Un électroaimant (12V) est désactivé via un relais (5V) libérant ainsi la chute d’un tableau (indice)

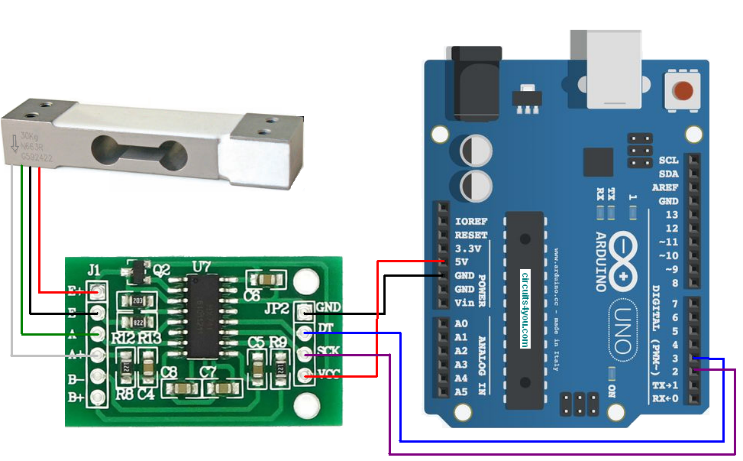
## Entrées :

* C\_Poids
  + Type de capteur : capteur de poids
  + Référence : HX711
  + Signal : numérique
  + Voltage :  0-5V

## Sorties :

* S\_Tableau
  + Rôle : activer/désactiver une électro aimant via un relais (tableau)
  + Condition : Si E1 est compris entre 48 grammes et 52 grammes alors activer la sortie, sinon, désactiver la sortie
  + Voltage : 0-5V
* S\_Led
  + Rôle : Désactivation LED rouge et activation LED verte (Indicateur de réussite)
  + Condition : Si E1 est compris entre 48 grammes et 52 grammes alors éteindre la LED rouge et allumer la LED verte, sinon c’est l’inverse
  + Voltage : 0-5V

## Remarques :

  
Schéma de branchement du capteur de poids HX711

Programme sur Arduino :

Le programme exécute principalement :

Une image contenant capture d’écran

Description générée automatiquement  
Une image contenant capture d’écran, ordinateur

Description générée automatiquement  
Une image contenant capture d’écran, ordinateur

Description générée automatiquement

On instancie d’abord un objet Riz qu’on nomme « mecanism ». On lui donne une configuration de base dans setupMechanism() puis on appelle en boucle la méthode « **execute()** » de la classe Riz.

Cette méthode exécute principalement les instructions suivantes :

Une image contenant capture d’écran

Description générée automatiquementUne image contenant capture d’écran, ordinateur

Description générée automatiquement