Une image contenant Police, logo, Graphique, texte

Description générée automatiquement

Louis Boisvert

Alexis Létourneau

Groupe : 2317

**La clé du savoir**

Manuel technique présenté à :

Benoit Beaulieu

Nicolas Huppé

Louis-Philippe Gauthier

Étienne Desbiens

Département du génie électrique

Pour le cours :

247-67P-SH PROJET DE FIN D'ÉTUDES

Cégep de Sherbrooke

Mai 2025

***Description générale de la mallette:***

InXtremis est un centre d'évasion situé au cœur du centre-ville de Sherbrooke, dirigé par un ancien étudiant en Technologie de Systèmes Ordinés (TSO). Le propriétaire aspire à concevoir une nouvelle salle intégrant des éléments interactifs électronique grâce à des dispositifs électroniques dissimulés dans les énigmes. Plusieurs de ses salles actuelles utilisent déjà un ou deux dispositifs électroniques permettant d'ouvrir des portes et de suivre le progrès des joueurs. Cependant, il souhaite aller encore plus loin en créant une salle entièrement interconnectée, sans cadenas pour rythmer la progression.

Si les mécanismes simples, comme l'ouverture d'une énigme après la résolution de la précédente, sont faciles à mettre en place, le propriétaire ambitionne d'offrir une expérience encore plus impressionnante avec une énigme finale particulièrement spectaculaire. Pour concrétiser cette vision, il fait appel aux finissants du programme TSO.

L'objectif est de concevoir un module sous la forme d’une mallette renfermant une série de plusieurs énigmes. Le temps nécessaire pour résoudre ces énigmes devra être compris entre 5 et 10 minutes. Le module devra également offrir une remise en état très bas pour accueillir rapidement la prochaine équipe.

La mallette devra être robuste pour résister à une mauvaise manipulation ou à une utilisation intensive. Le circuit électronique devra être facilement accessible pour les interventions de maintenance, de dépannage ou de démonstration. De plus, il devra pouvoir communiquer avec les autres éléments de la salle pour suivre la progression des joueurs et détecter des problèmes éventuels, comme la tentative de contourner une énigme.

A diagram of a computer chip

AI-generated content may be incorrect.

***Fonctionnement de la mallette:***

A diagram of a computer hardware system

AI-generated content may be incorrect.

**Descriptions générales du schéma :**

La mallette est composée d’un Raspberry Pi connecté en I2C en «Main» avec 3 esp32 en mode «Sub» et d’un clavier esp32 connecté au port USB du Pi. Les 3 esp32 sont connecté à des « éléments interactifs » comme des boutons, des interrupteurs, des potentiomètres et des bornes bananes avec des connecteurs JST.

Présentement, un des esp32 est connecté avec 8 interrupteurs, un autre esp32 avec 3 potentiomètres coulissant et un esp32 avec 8 bornes bananes. Dans cette architecture, le Pi fait des requêtes I2C au esp32 à chaque intervalles en secondes fixé dans le code et reçoit un JSON contenant les valeurs des éléments interactifs des esp32. Ces valeurs sont ensuite intégrées dans une énigme créé en python qui s’affiche sur un mini écran connecté au Pi.

Le clavier esp32 utilise une librairie spéciale qui permet de s’en servir en clavier et en souris. Le clavier esp32 vient près-assemblé avec 12 touches programmables.

Le PI contrôle aussi des bandes de DEL adressable pour illuminer la mallette et indiquer la réussite des énigmes.

Le PI est aussi connecté à un bouton lumineux. Le Pi détecte l’état du bouton et peut contrôler sa DEL interne.

Comme un des buts principales de la mallette est d’être modulable, elle n’a pas de programmation prédéterminé, par exemple : le bouton lumineux pourrait tant allumer les bandes de DEL qu’être un élément de réponse dans une des énigmes. Les bandes de DEL pourrait illuminer la mallette, être un indice ou indiquer le temps avec des clignotement de plus en plus rapidement. Cependant, la mallette contient exemple un code python qui est une série d’énigme utilisant tous les éléments interactifs de la mallette.

**Raspberry Pi :**

* Le Pi est le « Main » dans la communication I2C et envoie des requêtes d’informations au ESP32 très fréquemment (50 ms de base). Il reçoit un JSON comportant le nom du esp32 et l’état de ses éléments interactifs.
* Le Pi reçoit ces informations et intègre ces informations dans un programme python qui consiste d’un série d’énigme sur un interface utilisateur qui utilise la librairie PySimpleGUI. Chaque énigmes intègrent différents éléments interactifs.
* Lors du démarrage du Pi, le programme ouvre automatiquement le programme des énigmes.
* Le Pi prend en charge un clavier esp32 qui se connecte sur un port USB pour contrôler des lettres et la souris
* Le Pi reçoit l’état d’un bouton lumineux et peut faire allumer sa DEL.

**Esp32 :**

* Chaque ESP32 lit l’état des éléments interactifs et l’envoi en JSON au Pi via i2c. Les ESP32 sont des « Sub » attendant les requêtes du Raspberry Pi.
* Voici un exemple de JSON envoyé au PI :

{ «NomEsp32 »: « I2C \_SW », « JsonData » : {« SW1 »: « 0 » , « SW2 »: « 1 » , « SW3 »: « 0 » } }

Dans ce cas, le nom du esp32 est le « I2C \_SW », donc celui avec des interrupteurs, et le Pi reçoit que le bouton numéro 2 est appuyé. Le JsonData pourrait aussi contenir des valeurs analogues comme avec le esp32 « I2C \_POT »  qui s’occupe des potentiomètres.

* Le code de chaque esp32 est en c++ Arduino.
* Chaque esp32 est codé avec un « battement de cœur » sur sa DEL Neopixel qui montre que le esp32 est actif au démarrage et durant la communication I2C entre le esp32 et le Pi.

**Connecteurs JST :**

* Les connecteurs JST permettent de faire la connexion rapide et interchangeable entre la carte électrique (PCB) et le matériel externe tel que les éléments interactifs et les bandes de DEL.

**Mini écran :**

* Le mini écran est le moniteur du Pi et affiche la série l’énigme.
* Le mini écran peut aussi servir à offrir des indices, un message de fin et de début ainsi qu’une mise en situation.
* Le mini écran possède aussi des haut-parleurs intégrés permettant de jouer différent bruitage selon les actions effectuer par l’utilisateur

**Bandes de DEL adressable :**

* Les bandes de DEL adressable permette d’ajouter un élément visuel à la mallette.
* Les bandes de DEL adressable sont contrôlé par le Raspberry Pi et permettre présentement de signaler à l’utilisateur la réussite d’une énigme par la couleur verte ou l’échec de l’énigme par la couleur rouge. Les bandes de DEL adressable pourrait aussi illuminer la mallette, être un indice ou indiquer le temps qui découle avec des clignotement de plus en plus rapidement.
* Pour accéder la composante sur le Pi qui permet de contrôler les bandes de DEL adressable, il faut rouler le programme python en mode « sudo ».

**Rôle du bouton lumineux :**

* Le Pi détecte l’état du bouton lumineux et peut contrôler la DEL interne bouton lumineux .
* Le bouton lumineux pourrait permettre d’arrêter le décompte mettant fin à la série d’énigme. Pour l’instant il active uniquement la fonction Rainbow des DEL.

**Alimentation :**

* Un bloc d’alimentation murale fournira une tension de 5V par USB-C au circuit électrique (PCB). Il sera possible de connecter un chargeur portatif (pile de 5V avec prise USB-C) au lieu de le brancher au mur lors de démonstration où les prises murales ne sont pas facilement accessibles.
* Les esp32 et le Raspberry Pi, seront alimentés par le circuit électrique (PCB).
* Le mini écran ne peux pas être alimenter par le Raspberry Pi, car le Pi ne peut pas sortir assez de courant pour alimenter l’écran et tout le reste du PCB. Le mini écran par le bloc d’alimentation murale directement via un dédoubleur de câble USB-C.

**Programme de série d’énigme de base:**

* La mallette contient un programme en python qui part lors du démarrage du PI. Ce programme reçoit les états des objets interactifs dans un JSON via i2c puis l’intègre dans une énigme.
* Le programme affiche sur le mini écran l’énigme avec la librairie PySimpleGUI.
* Le programme contrôle les bandes de DEL adressable et le bouton lumineux.
* Le programme est constitué de plusieurs fichiers :
* « main.py » est le point d’entré du programme qui permet de démarrer les énigmes dans les autres modules ci-dessous.
* « Class\_Croco.py »  contient une énigme qui demande à l’utilisateur de connecter des bornes bananes numérotées ensemble pour compléter une équation. Exemple, la borne numéro 2 et la borne numéro 6 sont connectées ensemble et le programme veut une multiplication de deux chiffre qui égale 12, dans ce cas l’énigme est réussie car 12 = 2 x 6.
* « Class\_POT.py »  contient une énigme qui demande à l’utilisateur de manipuler des potentiomètres pour changer les paramètres d’amplitude, de période et de position en y d’un sinus dans le but de recopier un autre sinus généré aléatoirement.
* « Class\_SW.py »  contient une fenêtre de débogage qui affiche l’état des interrupteurs.
* « I2c\_Comm.py » contient la communication i2c.
* Voici le schéma fonctionnel de « Class\_Croco.py » et « Class\_POT.py » :

SCHÉMA FONCTIONNEL ÉQUATION

SCHÉMA FONCTIONNEL SINUS

***Procédure d’installation et d’opération :***

**Montage du PCB :**

Le PCB utilise principalement des composantes traversantes, sauf 5 composantes en surface. Ces pièces sont 4 transitoires de niveau pour les DEL adressable et l’autre est le port USB vertical. Suivre le BOM et le *silkscreen* du PCB pour assembler le PCB.

IMAGE PORT-USB-C

IMAGE TRANSISTOIRES

**Branchement du PCB :**

Brancher un bloc d’alimentation murale sur le port USB-C à l’extérieure de la mallette. Ce port USB-C extérieur est connecté au port USB-C vertical du PCB qui alimente le circuit. Ce même port USB-C extérieur sert à alimenter le mini écran. Les esp32 et le Pi sont alimenté par le port USB-C vertical via le PCB.

Ensuite, connecter le port micro-HDMI du Pi au port HDMI du mini écran.

Ensuite, il faut connecter les connecteurs JST des éléments interactifs, des bandes de DEL adressable et du bouton lumineux sur le PCB. Les indications sur le PCB indiquent où mettre chaque composant. Veuillez mettre un indicateurs à chaque bout des fils JST, cela facilitera le débogage quand tous les fils seront en couette.

***Initialisation du Raspberry PI :***

**Initialisation d'une nouvelle carte microSD pour un Raspberry Pi :**

Veuillez initialiser votre Raspberry Pi avec le système d’opération Raspbian avec la documentation officiel suivante  : <https://www.raspberrypi.com/documentation/computers/getting-started.html>

**Créer, initialiser et utiliser l’environnement virtuel sur le Pi :**

Vous pouvez prendre l’environnement virtuel déjà fait disponible sur le GitHub du projet, mais si vous voulez faire votre propre environnement virtuel voici la procédure :

1. Créez votre environnement virtuel avec la commande « python3 -m venv votre/chemin/de/fichier ».
2. Activez votre environnement virtuel avec la commande « source ./venv/bin/activate »
3. Ajouter des librairies avec la commande « python3 -m pip install XYZ ». XYZ est le nom de librairies voulu, voici la liste des librairies du projet :

* 'smbus2' Pour la communication I2C .
* 'PySimpleGUI' Pour l’interface utilisateur.
* 'rpi\_ws281x' Pour la communication avec les DEL adressable.
* 'Adafruit-Blinka' Pour pouvoir accéder au GPIO du Pi.
* Les autres librairies sont installées par défaut.

À des fins de débogage, si vous voulez exécuter le programme des énigmes, il faut activer l’environnement virtuel dans votre éditeur de code Python. L’éditeur de python de base dans Raspbian est Thonny, pour activer l’environnement virtuel, appuyer sur « Configure interpreter » et sélectionner le fichier « venv/bin/python3 » dans votre environnement virtuel.

IMAGE THONNY CONFIG

**Les fichiers et l’exécution du programme de série d’énigme de base :**

Tous les fichiers pour le programme de série d’énigme de base est sur le GitHub dans le fichier CODE\_POUR\_PI\_FONCTIONNEL. Veuillez sélectionner tous les fichiers et le mettre dans un répertoire de votre Pi. Soyez sûr que les 4 librairies spécifiques au projet soit installé dans le même répertoire que le fichier main.py.

Pour exécuter le programme manuellement, il faut démarrer Thonny en mode administrateur avec la ligne « sudo Thonny » dans le CMD

**Activer le « autostart » d’un programme :**

Le fichier monautostart.desktop est différent car il demande d'être installé dans le fichier /etc/xdg/autostart/ et d’être modifié selon les indications suivante.

Voici le contenu du fichier :

[Desktop Entry]

Name=monautostart

Exec=sudo /home/Pi/path/to/venv/bin/python3 /home/Pi/Desktop/Git/Projet\_Final\_TSO/Code/Mallette\_GUI/Prototype\_Fonctionnel\_sur\_PI/main.py

Type=Application

X-GNOME-Autostart-enabled=true

La partie importante est la ligne « Exec » :

« Sudo » est pour mettre le programme en administrateur pour les bande de DEL adressable.

« /home/Pi/path/to/venv/bin/python3 » doit changer selon l’emplacement de votre environnement virtuel.

« /home/Pi/Desktop/Git/Projet\_Final\_TSO/Code/Mallette\_GUI/Prototype\_Fonctionnel\_sur\_PI/main.py » doit changer selon le chemin absolu de votre programme de série d’énigme python.

**Activer la communication I2C du Pi :**

* Exécutez sudo raspi-config.
* Utilisez la flèche vers le bas pour sélectionner « 5 Interfacing Options».
* Utilisez la flèche vers le bas pour sélectionner « P5 I2C ».
* Sélectionnez « yes » lorsque l'option « I2C » est demandée.
* Sélectionnez également « Oui » si l'option « Chargement automatique du module du noyau » est demandée.
* Utilisez la flèche vers la droite pour sélectionner le bouton « Finish».
* Sélectionnez « yes » lorsque l'option « Redémarrer » est demandée.

Source : [https://learn.sparkfun.com/tutorials/raspberry-Pi-spi-and-I2C -tutorial/all](https://learn.sparkfun.com/tutorials/raspberry-pi-spi-and-i2c-tutorial/all)

**Configurer les esp32 :**

Pour configurer les esp32 il faut ouvrir le dossier CODE\_ESP32\_XYZ dans vscode en s’assurant d’avoir l’extensions c++ et platform.io. Pour les acquérir, dans vscode, aller dans l’onglet extension et chercher les extensions dans la barre de recherche.

Puis upload le main.cpp de chaque CODE\_ESP32\_XYZ dans le esp32 correspondant.

IMAGE D’UN UPLOAD DE ESP32

**Mode programmation du clavier:**

Pour entrer en mode programmation :

* Appuyer sur le bouton RESET
* En restant appuyé, appuyer sur le potentiomètre
* Puis relâcher le bouton RESET
* Un répertoire devrait s'ouvrir sur l'ordinateur contenant un des fichiers.

Le fichier important est code.py qui peut être modifié selon les informations sur : <https://github.com/KMKfw/kmk_firmware>, mais vous pouvez prendre le code.py sur le GitHub.

**Mise en route, validation du fonctionnement, dépannage :**

Si tout est bien configuré, lors du démarrage, une fenêtre devrait s’ouvrir sur le mini écran vous permettant de sélectionner une énigme. Les trois énigmes devraient fonctionner comme dans la section « fonctionnement de la mallette ».

Si l’énigme se ferme automatiquement, alors une exception est arrivée. Pour avoir plus d’information sur l’exception, il faut faire exécuter le code manuellement dans Thonny comme décrit plus haut et essayer de répéter les mêmes étapes qui ont mené à l’exception.

Si vous trouvez une exception dans le programme de base sans avoir l’avoir modifié, veuillez contacter un des techniciens du projet. Leurs contacts sont sur le GitHub.

***Contenu matériel :***

Énigme interrupteur:

Chaque interrupteur est connecté à une GPIO d’un esp32 en mode entrée et sont tous équipé avec une *pull-up*. Les GPIO 6 et 7 sont le SDA et le SCL respectivement pour la communication i2c. La patte de reset du esp32 est connecté au Pi pour pouvoir commencer la communication i2c lorsque le Pi est prêt à initier la communication.

A screenshot of a computer screen

AI-generated content may be incorrect.

Énigme bananes:

Chaque bornes bananes est connecté à une GPIO d’un esp32 pour permettre le changement rapide entre entrée *pull-down* qui écoute pour un signal haut et sortie qui envoi le signal haut, dans le but de savoir quelle bornes sont interconnectées. Les GPIO 6 et 7 sont le SDA et le SCL respectivement pour la communication i2c. La patte de reset du esp32 est connecté au Pi pour pouvoir commencer la communication i2c lorsque le Pi est prêt à initier la communication.

A diagram of a crocodile

AI-generated content may be incorrect.

Énigme Potentiomètre:

Chaque potentiomètre connecté à une GPIO d’un esp32 en mode entrée et sont tous équipé avec une *pull-up*. Les GPIO 6 et 7 sont le SDA et le SCL respectivement pour la communication i2c. La patte de reset du esp32 est connecté au Pi pour pouvoir commencer la communication i2c lorsque le Pi est prêt à initier la communication.

A screenshot of a computer

AI-generated content may be incorrect.

Bouton lumineux :

Le bouton lumineux a deux fonctions contrôlé par le Pi, il allume une DEL au centre du bouton et il donne l’état du bouton normalement ouvert.

Les pattes 1 et 2 sont pour la DEL. Lorsque *PIN\_PI\_26* est haut, la DEL est éteinte, Lorsque *PIN\_PI\_26* est bas, la DEL est allumée.

Les pattes 3 et 4 sont pour l’interrupteur, *PIN\_PI\_5* est en entrée avec une *pull-up*.

A diagram of a wiring diagram

AI-generated content may be incorrect.

Connecteur pour les bandes de DEL adressable:

Comme on utilise une bande de DEL adressable alimenté à 5v, on utilise un *level-shifter* pour envoyer une signal de 5v sur la patte de signal de la bande de DEL au lieu de 3.3v du Pi.

A diagram of a circuit

AI-generated content may be incorrect.

**Contenu logiciel :**

Liens vers le GitHub qui contient tous les codes commentés et autres composants nécessaires au fonctionnement de la mallette:

<https://github.com/A-Letourneau/Projet_Final_TSO>

Veuillez-vous référer à tous les Readme.md sur GitHub pour plus d’information

***Procédure de développement :***

Avec la base que vous avez créée dans « Procédure d’installation », vous pouvez modifier votre mallette avec vos propres modules esp32 connecté à des éléments interactifs et modifier le programme du série d’énigmes en python.

**Ajout d’un module esp32 en c++ :**

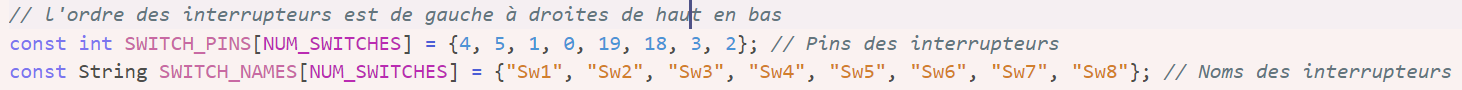
Prenez un des CODE\_ESP32\_XYZ et modifier les éléments suivants selon vos besoins.

1. Choisir une adresse I2C unique au esp32.

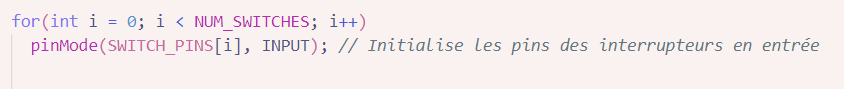


1. Associer les pattes et le noms des esp32 à vos éléments interactifs et initialiser les entrée des esp32.

La liste des pattes des éléments interactifs.

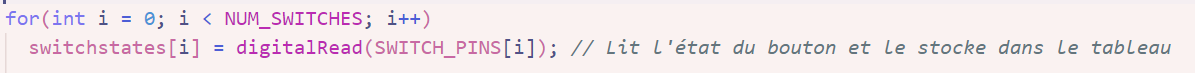


L’initialisation des éléments interactifs en entrée.



1. Dans « loop() », récolter les données de vos éléments interactifs.

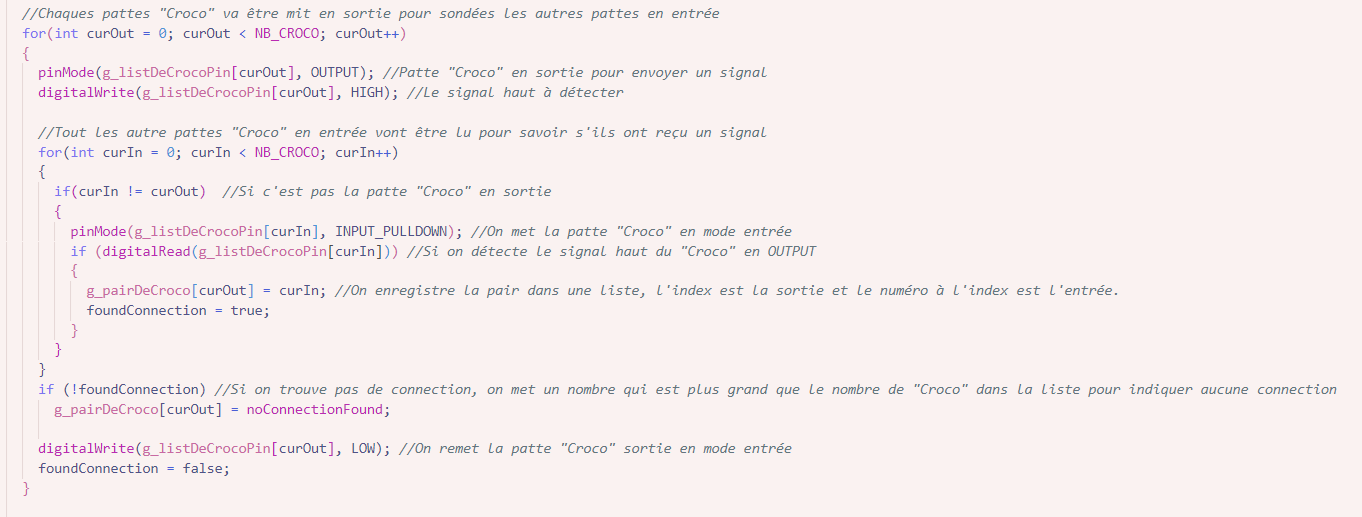
Dans le cas d’un élément interactif avec une sortie digital :



Dans le cas d’un élément interactif avec un sortie analogue :

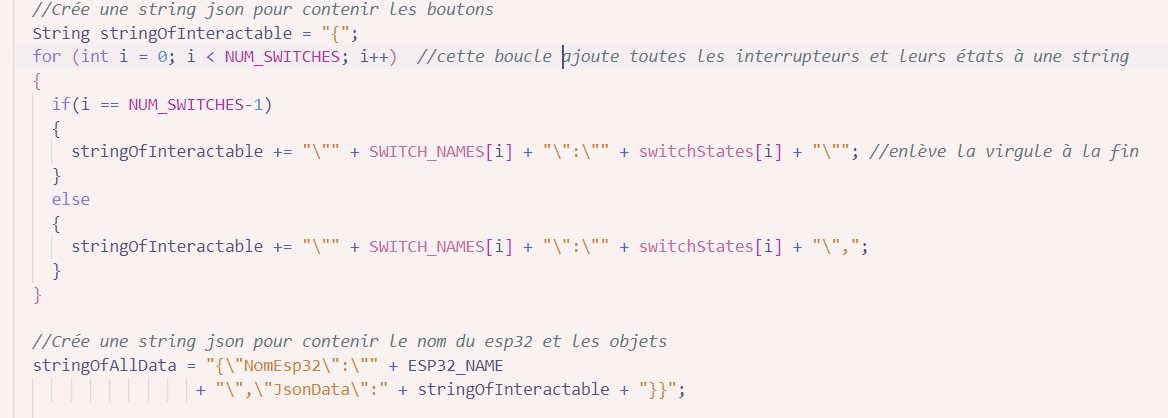


Il est possible que la récolte de donnée soit plus complexe, comme avec notre module esp32 de bornes bananes qui détecte l’interconnexion entre chaque bornes:



L’important est de mettre les données dans une liste ordonné pour la prochaine étape

1. Mettre les données dans un JSON :



Noter que le JSON ne doit pas dépasser 125 caractères,

Dans ce cas, le JSON résultant est :

{ «NomEsp32 »: « I2C \_SW », « JsonData » : {« SW1 »: « 0 » , « SW2 »: « 0 » … « SW8 : « 0 »  } }

Votre JSON doit suivre la même logique :

{ «NomEsp32 »: NOM\_DU\_ESP32 « JsonData » : { NOM\_DE\_L’ÉLÉMENT : VALEUR\_EN STRING, …} }

**Création d’une énigme en python :**

Nos énigmes sont mises dans des classe pour faciliter leurs intégrations

Voici un schéma simplifié d’une classe énigme :

IMAGE SCHÉMA SIMPLIFIÉ

La première étape de chaque énigme est d’ouvrir une fenêtre d’interface utilisateur, avec PySimpleGUI dans notre cas, pour afficher l’énigme.

IMAGE INTERFACE POT

La deuxième étape est de faire la communication i2c entre le Pi et le esp32 correspondant à l’énigme. Vous pourriez faire des requêtes pour plus qu’un esp32 si votre énigme demande plusieurs éléments interactifs différents.

Noter que vous devez avoir un système de prévention d’exception sinon une mauvaise communication i2c va faire planter le code.

Finalement, vous devez intégrer les états des objets interactifs dans une énigme. Vous pouvez vous inspirer de nos énigmes dans le GitHub.

Si vous voulez continuer avec PySimpleGUI, nous vous conseillons de lire le PySimpleGUI cookbook pour mieux comprendre comment cette librairie fonctionne.

<https://docs.pysimplegui.com/en/latest/cookbook/original/>

**Ajout d’une énigme en python :**

Maintenant que la classe contenant l’énigme est créé, il faut l’importer dans main.py avec la ligne « from NOM\_DU\_MODULE import NOM\_DE\_LA\_CLASSE » au début du fichier.

Premièrement, initialiser votre classe dans le main avec les paramètres nécessaires, exemple :

my\_Croco = Croco(SLAVE\_ADDRESS\_Croco, LIST\_OPERATIVE, DEBUG, strip1)

Deuxièmement, ajouter un « sg.Button("NOM\_DE\_L’ÉNIGME") » dans le « layout\_winSelect », cela rajoute un événement qui permettra de démarrer votre énigme.

Troisièmement, ajouter un événement après :

IMAGE READ\_ALL\_WINDOW

qui démarre votre énigme lorsque le bouton est appuyé. Exemple  :

IMAGE EVENT CREATE WINDOW

Et un événement qui détecte que la fenêtre a été fermé.

IMAGE EVENT DELETE WINDOW

***Listes de matériel et coûts :***

***Modifications et améliorations :***

Annexe

Schéma











Remerciement

On remercie TA avec notre équipe de Osentreprendre

On remercie Étienne d’InXtremis

On remercie les professeurs de TGE

(Faire à la fin)