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 du produit

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 du produit

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 un clavier USB connecté au port USB du Pi. Les 3 esp32 sont connecté avec 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 l’écran connecté au Pi.

Le clavier USB utilise une librairie spéciale qui permet de s’en servir en clavier ordinaire et en souris. Le clavier USB 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 fonctionnement 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.

**Rôles des Esp32 :**

* Chaque ESP32 reçoit les actions que l’utilisateurs effectues sur ses éléments interactifs via JSON.
* Les ESP32 utilises une communication I2C pour envoyer leurs données au Raspberry Pi, dans ce type de communication, les ESP32 sont des « Sub » écoutant les requêtes du Raspberry Pi et envoyant leurs données JSON au 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 » qui montre que le esp32 est actif au démarrage et durant la communication I2C entre le esp32 et le Pi.

**Rôles des 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.

**Rôles du Raspberry Pi :**

* Le Pi est le « Main » dans la communication I2C et envoie des requêtes d’informations au ESP32. Il reçoit un JSON comportant le nom du esp32 et l’état de ses éléments interactifs.
* Le Pi traite ces informations et affiche sur son interface utilisateur des énigmes qui intègre les éléments interactifs.
* 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 du bouton lumineux et peut faire allumer sa DEL.

**Mini écran :**

* Le mini écran affiche les énigmes du Pi.
* 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

**Rôles des 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 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 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 mettre le programme python en mode « sudo ».

**Rôle du bouton lumineux :**

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

**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.

***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 pour assembler le PCB.

A close up of a blue circuit board

AI-generated content may be incorrect.

Close-up of a circuit board with a black and silver connector

AI-generated content may be incorrect.

**Branchement du PCB :**

Brancher un bloc d’alimentation murale sur le port USB-C à l’extérieure de la mallette. Ce port USB-C 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 :

Créez votre environnement virtuel avec la commande « python3 -m venv votre/chemin/de/fichier ».

Activez votre environnement virtuel avec la commande « source ./venv/bin/activate »

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.

Pour utiliser votre environnement virtuel, allez dans Thonny, l’éditeur python par défaut de rasbian, appuyer sur « Configure interpreter » et sélectionner le fichier « python3 » dans votre environnement virtuel.

**Activer la communication I2C du Pi :**

1. Écrivez sudo raspi-config dans le CMD.

2. Utilisez la flèche vers le bas pour sélectionner « 5 options d'interface »

3. Flèche vers le bas jusqu'à « P4 SPI ».

4. Sélectionnez oui lorsqu'on vous demande d'activer SPI,

5. Sélectionnez également oui s'il est demandé de charger automatiquement le module du noyau.

6. Utilisez la flèche de droite pour sélectionner le bouton <Terminer>.

7. Sélectionnez Oui lorsqu'il vous demande de redémarrer.

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)

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

Mettre le fichier monautostart.desktop dans le répertoire etc/xdg/autostart/

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 » est l’emplacement de l’environnement virtuel

« /home/Pi/Desktop/Git/Projet\_Final\_TSO/Code/Mallette\_GUI/Prototype\_Fonctionnel\_sur\_PI/main.py » est le chemin absolu du programme python.

**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.

**Mode programmation du clavier:**

Pour entrer en mode programmation

Appuyer sur le bouton RESET

En restant appuyé, appuyer sur le potentionmètre

Puis relâcher le bouton RESET

Un répertoire devrait s'ouvrir sur l'ordi

Le fichier important est code.py qui peut être modifié selon les informations sur : <https://github.com/KMKfw/kmk_firmware>

Ou vous pouvez prendre le code.py sur le GitHub.

***Contenu matériel***

Alimentation:

A diagram of a circuit

AI-generated content may be incorrect.

Del:

A diagram of a circuit

AI-generated content may be incorrect.

bouton lumineux:

A diagram of a circuit

AI-generated content may be incorrect.

raspberry Pi:

A diagram of a circuit board

AI-generated content may be incorrect.

bornier(optionnel) :

A diagram of a computer

AI-generated content may be incorrect.

condensateur de découplage :

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 vos série énigmes en python.

*Ajout d’un module esp32 en c++ :*

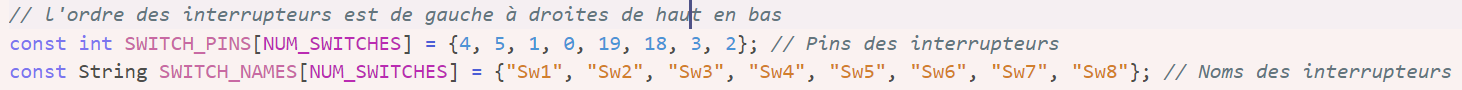
Prenez un des CODE\_ESP32\_XYZ et modifier les éléments suivants selon vos besoins.

1. Choisir une adresse I2C unique au esp32.

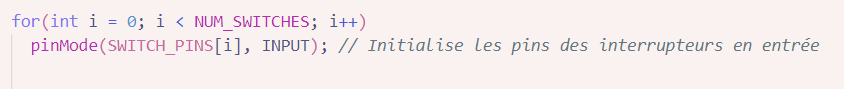


1. Associer les pattes et le noms des esp32 à vos éléments interactifs et initialiser les entrée des esp32.

La liste des pattes des éléments interactifs.

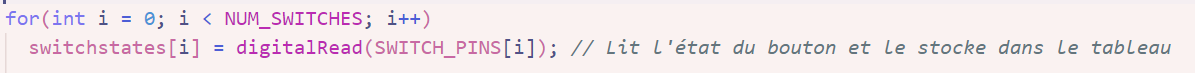


L’initialisation des éléments interactifs en entrée.



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

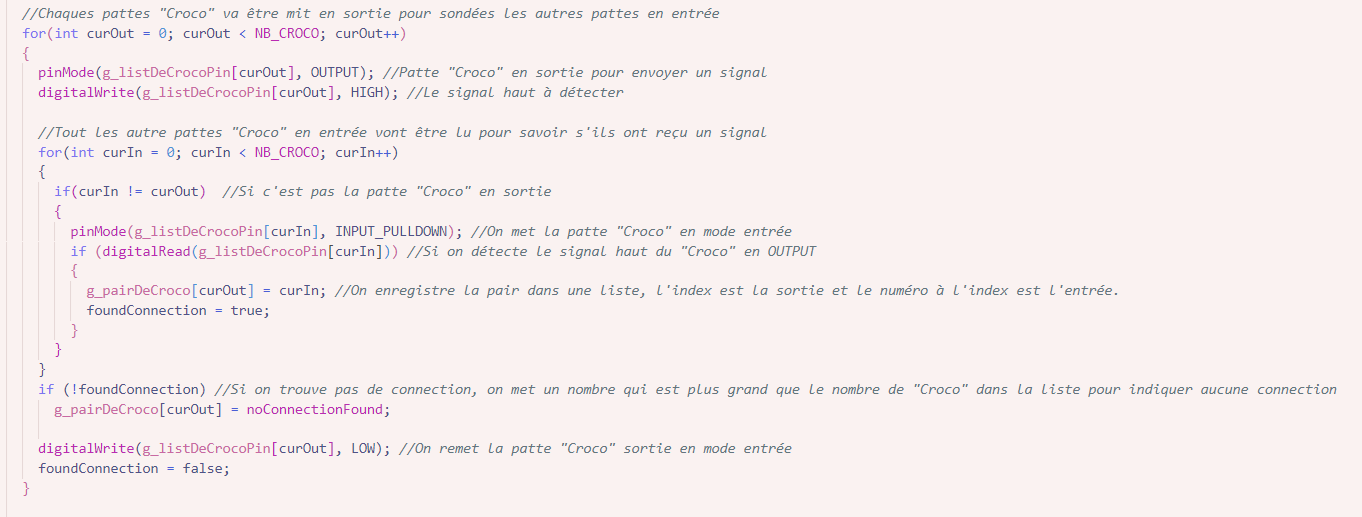
Dans le cas d’un élément interactif avec une sortie digital :



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

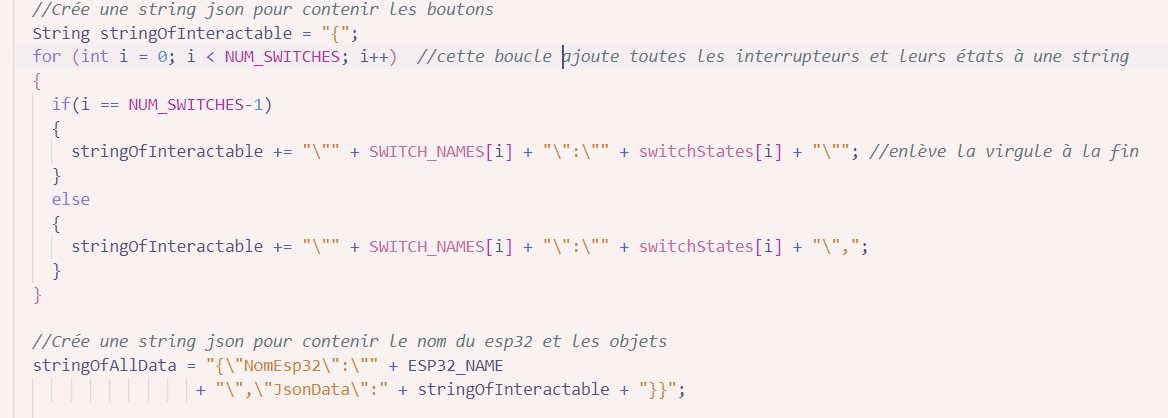


Il est possible que la récolte de donnée soit plus complexe, comme avec notre module esp32 de bornes bananes qui détecte l’interconnexion entre chaque bornes:



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, …} }

**Listes de matériel et coûts :**

**Modifications et améliorations :**

Parler des changements à faire pour une version 2 qui sont expliquer dans le plan de test

Annexe

Schéma











Remerciement

On remercie TA avec notre équipe de Osentreprendre

On remercie Étienne d’Inextremis

On remercie les professeurs de TGE

(Faire à la fin)