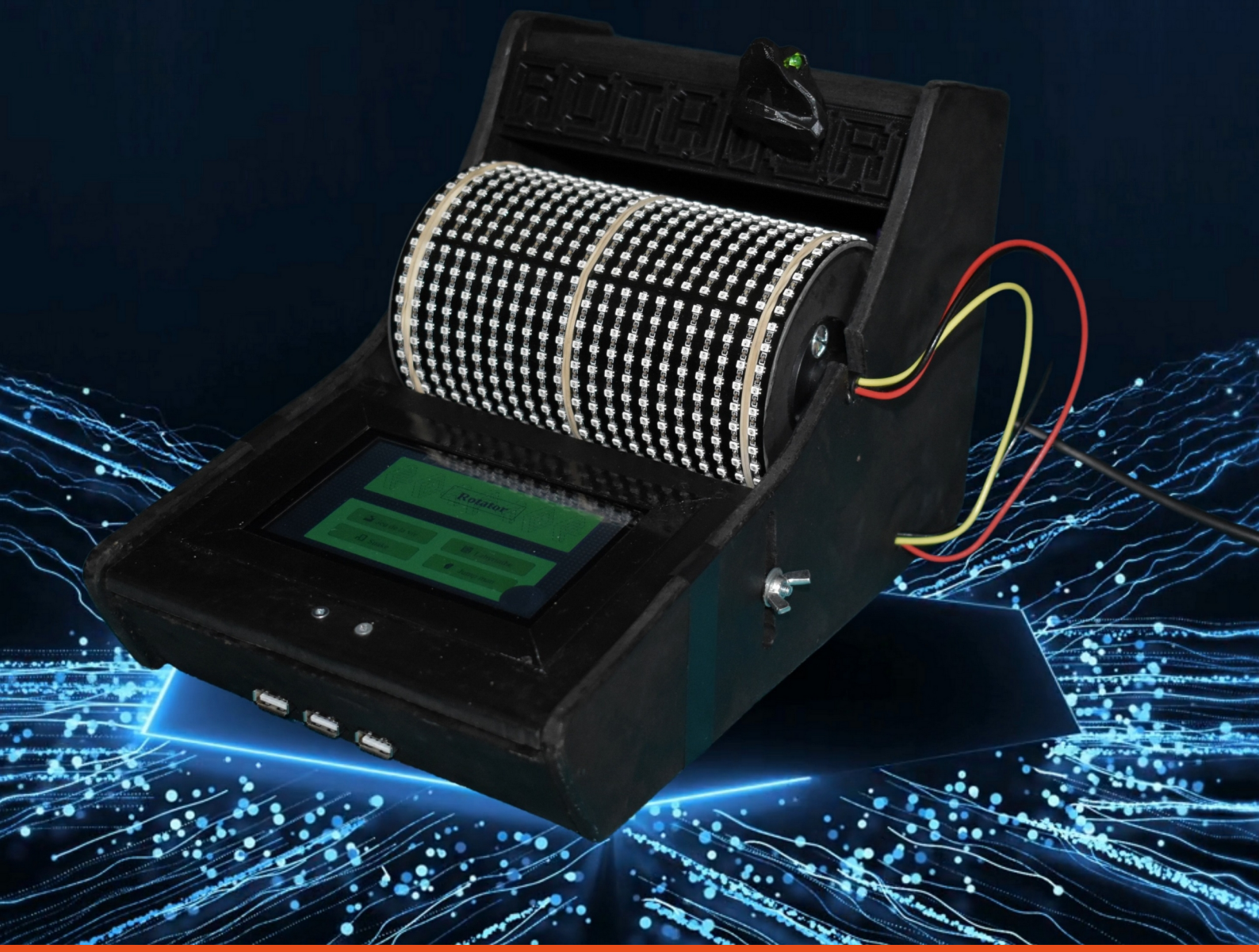
LES TROPHÉES NSI

Édition 2023

DOSSIER DE CANDIDATURE
PRÉSENTATION DU PROJET

ROTATOR



> PRÉSENTATION GÉNÉRALE[1] :

• *Idée et objectifs*
• *Origines et intérêts du projet*
• *(…)*

Le projet **Rotator** est un projet de création d'une **mini borne d'arcade** munie d'un **cylindre de LEDs rotatif**. Conçu dans le cadre des projets de la spécialité Numérique et Sciences Informatiques en classe de terminale, il a très vite été question de produire **différents jeux** interfacés par une Interface Homme Machine (IHM) "user-friendly", c'est-à-dire accessible.

La borne permet de jouer à différents **jeux intemporels d'arcades** comme le **Snake**, le **Labyrinthe**, et **Jumpman**. D'autres **algorithmes** et **utilitaires** sont mis à disposition comme **Image Maker** (convertisseur d'image sur matrice LED), **Scribble** (Dessinateur d'image) ou encore le célèbre **Jeu de la vie** d'Alan Turing.

Par le biais de jeux codés avec soin par les membres de notre projet, nous souhaitons permettre à des personnes de tout âge de **se remémorer le bon vieux temps** passé sur une borne d'arcade. De plus, grâce au choix de la forme cylindrique d'une part, et aux nouvelles options de jeu d'autre part, nous avons apporté notre petite touche d'**originalité** à notre borne d'arcade, en lui offrant une **dimension infinie et un gameplay varié**.

Suite au visionnage d'une vidéo de **cubes de LED**, les initiateurs du projet ont été émerveillés par la beauté et par les possibilités infinies d'une telle technologie. Il nous est venue l'idée de reproduire et d'adapter un **jeu de la vie en 3D**. Malheureusement, face aux limites matérielles, économiques et temporelles d'un projet scolaire, ce projet n'était pas viable. C'est alors que nous avons songé à créer **non pas un cube**, mais plutôt un **cylindre muni de panneaux LED** ! ***Voici comment naquit le projet Rotator.*** Muni de son cylindre rotatif, Rotator représente une **manière innovante de rejouer à ces jeux vidéo** qui peuvent nous rendre si nostalgiques. Il est ainsi **multigénérationnel, intemporel et novateur.**

Nous avons vu dans ce projet un moyen de **rassembler les générations**. En effet, les possibilités de programmation étant infinies et relativement simples, il est **possible de créer son propre programme**. Rotator peut donc servir de **support d'approche à la programmation**, mais aussi de **support d'approche à l'électronique**. Les débutants peuvent, par exemple, apprendre à déplacer une LED et faire tourner un moteur, les plus avancés quant à eux, développeront leur propre algorithme découvrant de nouvelles manières de programmation.

Rotator est aussi une **manière innovante de jouer aux jeux vidéos classiques**. C'est une manière de **faire revivre cette époque mythique du jeu vidéo que sont les années 1990**. C'est aussi une **performance artistique** visant à interroger le numérique et les limites de sa matérialisation dans le monde réel. En effet, si l'on prend l'exemple de la machine de Turing qui est composée d'un ruban infini, il est possible de se rendre compte que sa construction n'est pas physiquement réalisable sans l'utilisation d'un ruban mis sous forme cylindrique.

1 Ce texte est identique à celui de la présentation de Rotator dans notre documentation. Cependant, afin de suivre les consignes et faciliter le travail du jury, nous avons pensé qu'il était préférable de proposer une version correspondant au schéma demandé.

> ORGANISATION DU TRAVAIL :

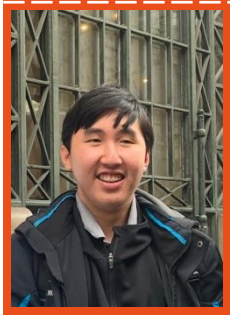
• Présentation de l'équipe (prénom de chaque membre et rôle dans le projet)
• Répartition des tâches
• Organisation du travail (répartition par petits groupes, fréquence de réunions, travail en dehors de l'établissement scolaire, outils/logiciels utilisés pour la communication et le partage du code, etc.)

DA COSTA SILVA Mathias

- Développeur Jeu de la Vie
- Design, modélisation et construction
- Rédaction de la documentation

JOSEPH-ANTOINE Max

- Développeur Snake
- Développeur IHM et serveur Apache
- Aide au débogage
- Montage de la vidéo



NGUYEN Van-Kevin

- Développeur Jumpman
- Vérification de l'interopérabilité
- Communication

SALVAING Louise-Anaïs

- Développeuse Labyrinthe
- Création de l'identité graphique
- Correction des textes

Nous nous sommes organisés de telle manière à ce que **chacun puisse travailler individuellement**. Lors de nos **réunions le vendredi**, nous ~~jouions aux différents jeux~~ **mettions en commun** le travail et nous nous **entraidions** afin que personne ne reste bloqué.

Nous avons travaillé avec **GitHub,** puis **Framagit,** pour le partage du code ce qui nous permettait d'une part, de le partager à chacun, mais aussi et surtout de pouvoir y avoir facilement accès depuis le terminal de la Raspberry PI.

De plus, nous avons choisi de communiquer sur le réseau social **Whatsapp** car ce dernier nous a forcé à garder notre sérieux pour ne pas perdre le fil de la discussion.

> LES ÉTAPES DU PROJET :

• Présenter les différentes étapes du projet (de l'idée jusqu'à la finalisation du projet)

I. Faire émerger l'idée
   A. Brainstorming autour de l'idée d'un Jeu de la Vie sur un cube en 3D
   B. Faisabilité et adaptation de l'idée à nos moyens et nos contraintes
   C. Brainstorming autour de l'idée d'une mini borne d'arcade avec un Snake sur des LEDs.
II. Établir un cahier des charges
   A. Comprendre les besoins de chaque jeux
   B. Dessiner des croquis
   C. Regarder les bibliothèques utilisables
   D. Définir une liste d'achats de produits correspondants à nos besoins
III. Prototypage
   A. Modélisation de la borne en 3D
   B. Développement des jeux avec un « simulateur » développé par nos soins
IV. Développement
   A. Construction de la borne
   B. Adaptation des jeux aux panneaux LEDs
   C. Développement de la partie motorisation, de la gestion des manettes et de la partie serveur

> FONCTIONNEMENT ET OPÉRATIONNALITÉ :

• *Avancement du projet (ce qui est terminé, en cours de réalisation, reste à faire)*
• *Approches mises en œuvre pour vérifier l'absence de bugs et s'assurer de la facilité d'utilisation du projet*
• *Difficultés rencontrées et solutions apportées*

1. Avancement du projet

Le **Jeu de la Vie**, le **Jumpman**, le **Labyrinthe** et le **Snake** sont terminés et jouables. L'interface et la partie serveur **fonctionnent correctement**. Il n'y a pas de bug à déplorer de ce côté là.

La **partie multi-joueurs** du Jumpman et du Snake est **en cours de réalisation**. Dans le cas du Snake, plusieurs serpents s'affichent mais certains bugs persistent.

Il nous reste à développer la possibilité de faire fonctionner Scribble et Image Maker en passant par l'interface graphique sur le Raspberry.

2. Approches mises en œuvre pour vérifier l'absence de bugs et s'assurer de la facilité d'utilisation

Nous avons fait tester les jeux à nos amis qui ont pu nous faire remonter des difficultés que nous n'avions pas prévues (exemple : le TimeOut du serveur Apache2). Après avoir répété l'exercice plusieurs fois, nous pensons maintenant **ne plus avoir de bugs**.

Tout au long du développement, nous souhaitions nous **assurer** de **la disponibilité** et de la **facilité d'utilisation**. C'est pour cette raison que nous avons adapté notre interface de manière à ce qu'elle soit claire et minimaliste. Afin de s'assurer de la réussite de cette opération, nous avons mis l'objet, débranché, **dans les mains de nos parents** sans leur donner d'explication. Ces derniers ont pu jouer sans difficulté grâce notamment au **script** qui lance la Raspberry Pi directement sur notre interface.

3. Difficultés rencontrées et solutions apportées

Lors du brainstorming, nous nous sommes confrontés au problème de **l'enroulement des câbles** d'alimentation des panneaux LEDs dans le cylindre. Nous ne savions pas comment le résoudre et avions d'abord penser à l'utilisation de connectiques circulaires comme le jack. Finalement, nous avons découvert l'**existence des collecteurs tournants** qui répondaient à nos besoins.

Lors du développement des jeux sur les LEDs, nous avons fait face à de nombreuses erreurs venant de la bibliothèque *blinka* qui permet la gestion des panneaux. Après avoir étudié les bibliothèques dans leur intégralité, nous avons finalement compris que nous ne disposions pas du canal « de chaleur » des LEDs, ce qui causait une erreur. Nous avons donc supprimé cette fonctionnalité de la bibliothèque, ce qui a résolu notre problème.

Finalement, afin de rester au maximum dans le cadre du programme, **nous ne souhaitions pas utiliser de framework tel que Flask ou Django**. C'est pour cette raison que nous avons contourné le problème grâce au lancement de processus Python via l'interface codée en HTML.

> OUVERTURE :

• Idées d'améliorations (nouvelles fonctionnalités)
• Stratégie de diffusion pour toucher un large public (faites preuve d'originalité !)
• Analyse critique du résultat (si c'était à refaire, que changeriez-vous dans votre organisation, les fonctionnalités du projet et les choix techniques ?)

Pour améliorer notre produit, nous pourrions **développer d'autres jeux** comme Pacman, Casse brique ou un jeu de Pong.

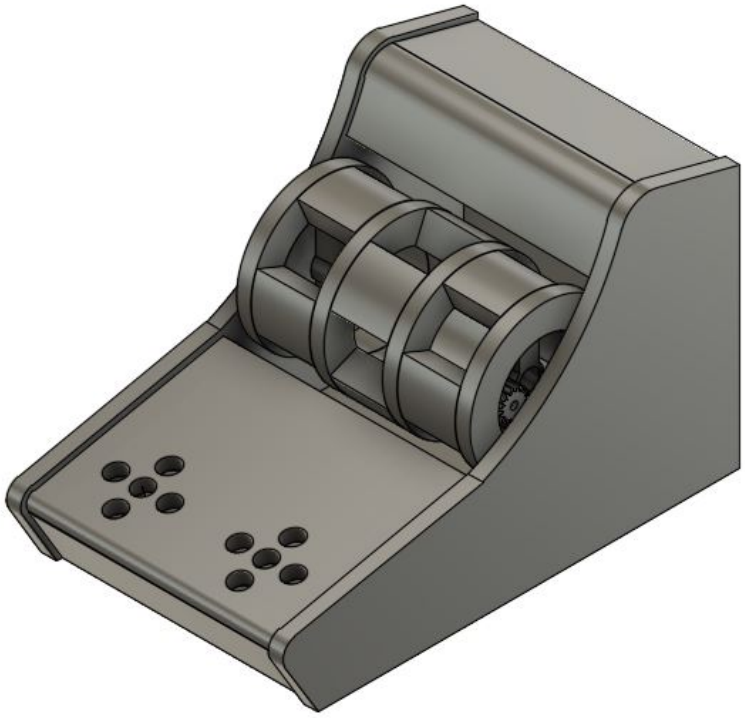
Afin de toucher un large public, nous pourrions développer une **plateforme de partage des jeux**. Ainsi, chaque personne pourrait développer son propre jeu et laisser la possibilité à **la communauté d'apporter des variantes**. Finalement, en créant des **ateliers dans des associations** d'apprentissage au code, Rotator serait aussi une manière ludique d'**apprendre le Python** et permettrait à de nombreux jeunes de découvrir le code.

Si c'était à refaire, nous nous tournerions sans doute vers une **construction** faite à l'aide d'une **fraiseuse à commande numérique**. En effet, le **résultat serait sans doute plus professionnel** avec moins de jeu entre les pièces. Aussi, nous pourrions chercher un moyen de **réduire la consommation électrique** du produit en passant par un écran E-Ink, ou en développant directement une interface sur les panneaux LEDs.

LYCÉE LOUIS-LE-GRAND

Numérique et Sciences Informatiques

ROTATOR : Un projet unique



Mathias DA COSTA SILVA
Max JOSEPH-ANTOINE
Van-Kevin NGUYEN
Louise-Anaïs SALVAING

Table des matières

1 Préambule

1.1 Licence d'utilisation

Même si nous pensons que Rotator a un potentiel commercial, nous avons choisi de rendre l'ensemble du projet open-source. En effet, nous sommes conscients de l'intérêt de la communauté du libre dans notre société. Dans notre projet, nous en sommes les premiers dépendants, que ce soit avec l'utilisation de Linux sur notre Raspberry PI ou nos ordinateurs personnels, Apache pour le serveur, ou encore avec l'utilisation de Framagit, LibreOffice et même TexMaker pour la rédaction de ce document. Nous sommes fiers de participer à, nous l'espérons, une émancipation numérique. C'est pour cette raison que le choix d'une licence CC BY-SA était primordial pour nous.

1.2 Concernant ce document

Nous souhaitions simplement vous informer sur le fait que la totalité de cette documentation est écrite en LaTeX. C'est ce langage qui nous a permis d'obtenir un document que nous espérons propre et bien présenté. Si vous disposez de la version PDF, il est à noter que l'ensemble du document contient des liens cliquables. En effet, nous avons tenu à ce que les références en ligne soit cliquables, qu'il y ait des point d'ancrage pour les sections et que les différents retours sur les figures soient accessibles. Ainsi, nous espérons que malgré les nombreuses pages, la lecture de cette documentation sera la plus claire et digeste possible.

1.3 Bonne lecture !

Cette documention est rendue publique car nous avons la volonté de partager. Il était primordial pour nous que vous puissiez reproduire et améliorer ce projet. Par conséquent, nous avons tenu à rédiger une documentation longue et complète que nous espérons précise et claire. Pour toute incompréhension, n'hésitez pas à nous contacter.

Nous vous souhaitons une très bonne lecture !

2 Projet Rotator

Le projet Rotator est un projet de création d'une mini borne d'arcade munie d'un cylindre de LEDs rotatif. Conçu dans le cadre des projets de la spécialité Numérique et Sciences Informatiques en classe de terminale, il a très vite été question de produire différents jeux interfacés par une Interface Homme Machine (IHM) "user-friendly", c'est-à-dire accessible.

La borne permet de jouer à différents jeux intemporels d'arcade comme le Snake, le Labyrinthe, et Jumpman. D'autres algorithmes et utilitaires sont mis à disposition comme Image Maker (convertisseur d'images sur matrice led), Scribble (Dessinateur d'images) ou encore le célèbre Jeu de la Vie de John Horton Conway (1937-2020).

Par le biais de jeux codés avec soin par les membres de notre projet, nous souhaitons permettre à des personnes de tout âge de se remémorer le bon vieux temps passé sur une borne d'arcade. De plus, grâce au choix de la forme cylindrique d'une part, et aux nouvelles options de jeu d'autre part, nous avons apporté notre petite touche d'originalité à notre borne d'arcade, en lui offrant une dimension infinie et un gameplay varié.

2.1 Le groupe

— Mathias DA COSTA SILVA
— Max JOSEPH-ANTOINE
— Van-Kevin NGUYEN
— Louise-Anaïs SALVAING

2.2 L'objet

Suite au visionnage d'une vidéo de cubes de LED, les initiateurs du projet ont été émerveillés par la beauté et par les possibilités infinies d'une telle technologie. Il nous est venue l'idée de reproduire et d'adapter un jeu de la vie en 3D. Malheureusement, face aux limites matérielles, économiques et temporelles d'un projet scolaire, ce projet n'était pas viable. C'est alors que nous avons songé à créer non pas un cube, mais un cylindre muni de panneaux LED ! Voici comment naquit le projet Rotator.

Muni de son cylindre rotatif, Rotator représente une manière innovante de rejouer à ces jeux vidéo qui peuvent nous rendre si nostalgiques. Il est ainsi multigénérationnel, intemporel et novateur.

2.3 Une vidéo de présentation

Afin de présenter notre projet aux "Trophées NSI", nous avons tourné une vidéo de présentation. Nous avions pour contrainte de présenter le groupe, les rôles de chacun et le projet en moins de deux minutes. C'est pour cette raison que nous nous sommes tournés vers une vidéo "commerciale". Avant de continuer, vous pouvez visionner la vidéo sur le lien suivant :
https://www.orion-hub.fr/w/5w5E6eEAsJLN2FnpHRYZT3?loop=1
Comme vous l'aurez sans doute remarqué, nous nous sommes inspirés de l'incontournable présentation de Steve Jobs de l'Iphone. Nous espérons que notre vidéo vous a plu et qu'elle a suscité votre intérêt !

2.4 Répertoire de code

L'entièreté de notre code est disponible sur le lien suivant :
https://framagit.org/mxth2xs/rotator

2.5 L'intérêt du projet

Nous avons vu dans ce projet un moyen de rassembler les générations. En effet, les possibilités de programmation étant infinies et relativement simples, il est possible de créer son propre programme. Rotator peut donc servir de support d'approche à la programmation, mais aussi de support d'approche à l'électronique. Les débutants peuvent, par exemple, apprendre à déplacer une led et faire tourner un moteur, les plus avancés quant à eux, développeront leur propre algorithme découvrant de nouvelles manières de programmation.

Rotator est aussi une manière innovante de jouer aux jeux vidéos classiques. C'est une manière de faire revivre cette époque mythique du jeu vidéo que sont les années 1990. C'est aussi une performance artistique visant à interroger le numérique et les limites de sa matérialisation dans le monde réel. En effet, si l'on prend l'exemple de la machine de Turing qui est composée d'un ruban infini, il est possible de se rendre compte que sa construction n'est pas physiquement réalisable sans l'utilisation d'un ruban mis sous forme cylindrique.

2.6 Ce que nous aimerions réaliser

Comme vous avez certainement pu vous en rendre compte, le projet est relativement complet, mais nos idées sont débordantes et nous continuons à avoir de nouvelles idées constamment. C'est ainsi que nous aimerions compléter ce projet avec de nouveaux programmes comme l'incontournable Pacman.

Avant toutes choses, nous aimerions finaliser les quelques options encore indisponibles mais en cours de programmation. Elles seront matérialisées par *une mise en italique suivie de la mention [en cours de programmation]*. Il peut parfois s'agir d'une simple mise en correspondance sur l'interface graphique du projet, mais il peut parfois s'agir de devoir compléter l'algorithme de l'option.

Aussi, nous aimerions proposer différentes versions du projet. Comme vous pourrez le constater dans le matériel nécessaire, nous avons fait le choix d'utiliser un écran LCD de 800 x 480 pixels. Cependant, nous aimerions proposer une version avec un afficheur LCD 16 x 2 caractères ou bien même un afficheur E-INK qui permettrait d'une part, de réduire la consommation et d'autre part, de reprendre le design original avec les boutons intégrés à la borne d'arcade. Aussi, nous aimerions travailler sur une bande-originale pour Rotator permettant ainsi de travailler sur l'électronique côté audio.

Il a aussi été question lors du projet de monter le cylindre sur un "bras arrière" permettant de le faire passer d'une position horizontale à une position verticale que ce soit à l'aide de moteur ou bien même manuellement. Cette fonctionnalité permettrait ainsi à Rotator d'afficher des messages ou bien des films. Dans ce cas, nous serions ravis de reprendre le travail du dessinateur Italien Osvaldo Cavandoli dans sa série télévisée nommée *La Linea*. En effet, ce type d'animation se prête parfaitement à l'affichage matricielle. D'une manière plus générale, l'existance de films en caractères ASCII peut permettre l'affichage sur Rotator, il s'agit là de développer un programme performant quant à la compression des données.

3 Arborescence du projet

Avant de vous présenter l'ensemble des programmes, nous vous laissons à disposition l'aborescence du projet. Vous pourrez vous référer à cette page pour trouver les algorithmes correspondants aux différents programmes mentionnés.

```
/
├── bonus
│   ├── image_maker ..... Max
│   └── scribble ..... Mathias
├── doc ..... Mathias
│   ├── src
│   │   ├── plan_arcade.pdf
│   │   ├── Rotator_accueil.JPG
│   │   ├── Rotator_perspective.JPG
│   │   ├── schema_elec.JPG
│   │   └── usb_pinout.JPG
│   ├── moteur.py ..... Mathias
│   ├── readme.pdf
│   └── readme.tex
├── fichiers_3D ..... Mathias
│   ├── borne_arcade_alternative.f3z
│   ├── borne_arcade.f3z
│   └── name_rotator_serpent.f3d
└── www ..... Max
    ├── data
    │   ├── cells ..... https ://conwaylife.com/wiki
    │   │   └── fichiers.cells
    │   ├── noms ..... Max
    │   │   ├── adjectifs.txt
    │   │   └── noms.txt
    │   ├── scores ..... Max
    │   │   └── score.json
    │   └── jdv_favoris.txt ..... Van-Kevin
    ├── images ..... Louise-Anaïs
    │   ├── flavicon.ico
    │   ├── jeuDelaVie.svg
    │   ├── jumpman.svg
    │   ├── labyrinth.svg
    │   ├── led_panel.jpg
    │   ├── logo.png
    │   └── snake.svg
    └── jeux
        ├── lib. ..... Mathias
        │   ├── adafruit_raspberry_pixelbuf.py
        │   ├── neopixel_arduino.py .... (Inutile sur Raspberry mais permet une autre configuration)
        │   ├── neopixel_raspberry.py
        │   ├── Pseudo_affichage.py ..... Max
        │   └── servo.py
        ├── config.py ..... Mathias
        ├── eteint.py ..... Mathias
        ├── jeudelavie.py ..... Mathias
        ├── jumpman.py ..... Van-Kevin
        ├── labyrinthe.py ..... Louise-Anaïs
        ├── libs_externes.py ..... Max
        └── snake.py ..... Max
```


- pages ..... Max
  - template_head_footer.html
  - template_playing.html
  - template_setup.html
- script ..... Max
  - index.js
- style ..... Max
  - index.css
  - template.css
- index_game.py ..... Max
- index.html ..... Max
- libs_index.py ..... Max
- .env.example ..... Max
- README.md ..... Mathias
- License.md ..... Mathias

4 Algorithme

4.1 Jeu de la vie

Le jeu de la vie est un automate cellulaire, inventé par le mathématicien britannique John Horton Conway en 1970. Il s'agit d'un jeu aux règles limitées qui se déroule sur une grille bidimensionnelle de cellules carrées, où chaque cellule peut être soit morte soit vivante.

4.1.1 Les règles

Le jeu de la vie fonctionne selon un ensemble de règles simples. À chaque étape, l'état de chaque cellule est mis à jour en fonction de l'état de ses voisines :

- Une cellule morte ayant exactement trois voisines vivantes devient vivante à l'étape suivante.
- Une cellule vivante ayant deux ou trois voisines vivantes reste vivante à l'étape suivante, sinon elle meurt.
- *Nous aimerions laisser la possibilité à l'utilisateur de choisir ses propres règles : c'est à ajouter sur l'interface grahique.*

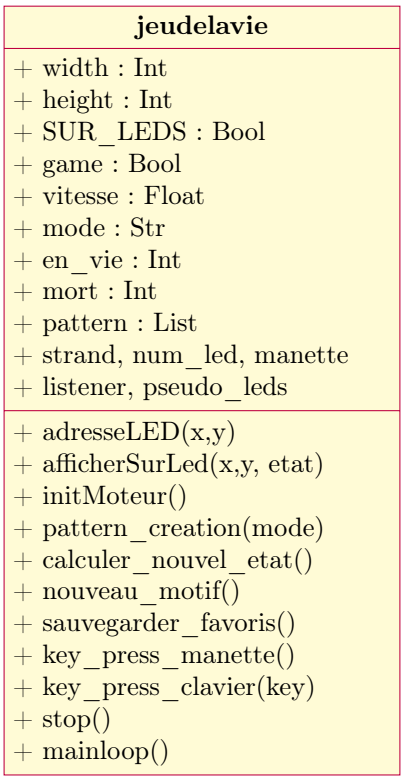
À partir de configurations initiales simples, il est possible d'observer une grande variété de comportements complexes et intéressants. Certains motifs se stabilisent en formant des structures récurrentes, tandis que d'autres évoluent de manière chaotique ou se déplacent à travers la grille.

4.1.2 Quelques positions de départ

Avec un panneau comme le notre de 22*44 soit 968 LEDs, nous pouvons considérer qu'il y a un nombre infini de combinaisons. En effet, chaque cellule pouvant prendre deux états, vivante ou morte (1 ou 0), il y a 2^{968} combinaisons. Ce qui fait $2^{968} = 2.4948e + 291$. Ainsi, il y a très largement plus de combinaisons de départ que le nombre estimé d'atomes observables dans l'univers (10^{80}).

Cependant, grâce au travail publié sur https://conwaylife.com/wiki/, nous avons pu télécharger un certain nombre de fichiers ".cells" (format de fichier lisible par l'homme, mais pas très efficace, qui est principalement utile pour les petits modèles. Les cellules mortes sont enregistrées sous la forme "." et les cellules vivantes sous la forme "O". Voir https://conwaylife.com/wiki/Plaintext). Ainsi, grâce à un algorithme Python, nous sommes en mesure de convertir le fichier en liste de liste, afin de donner des combinaisons de départs intéressantes.

4.1.3 Diagramme UML



5 Jeux

5.1 Jumpman

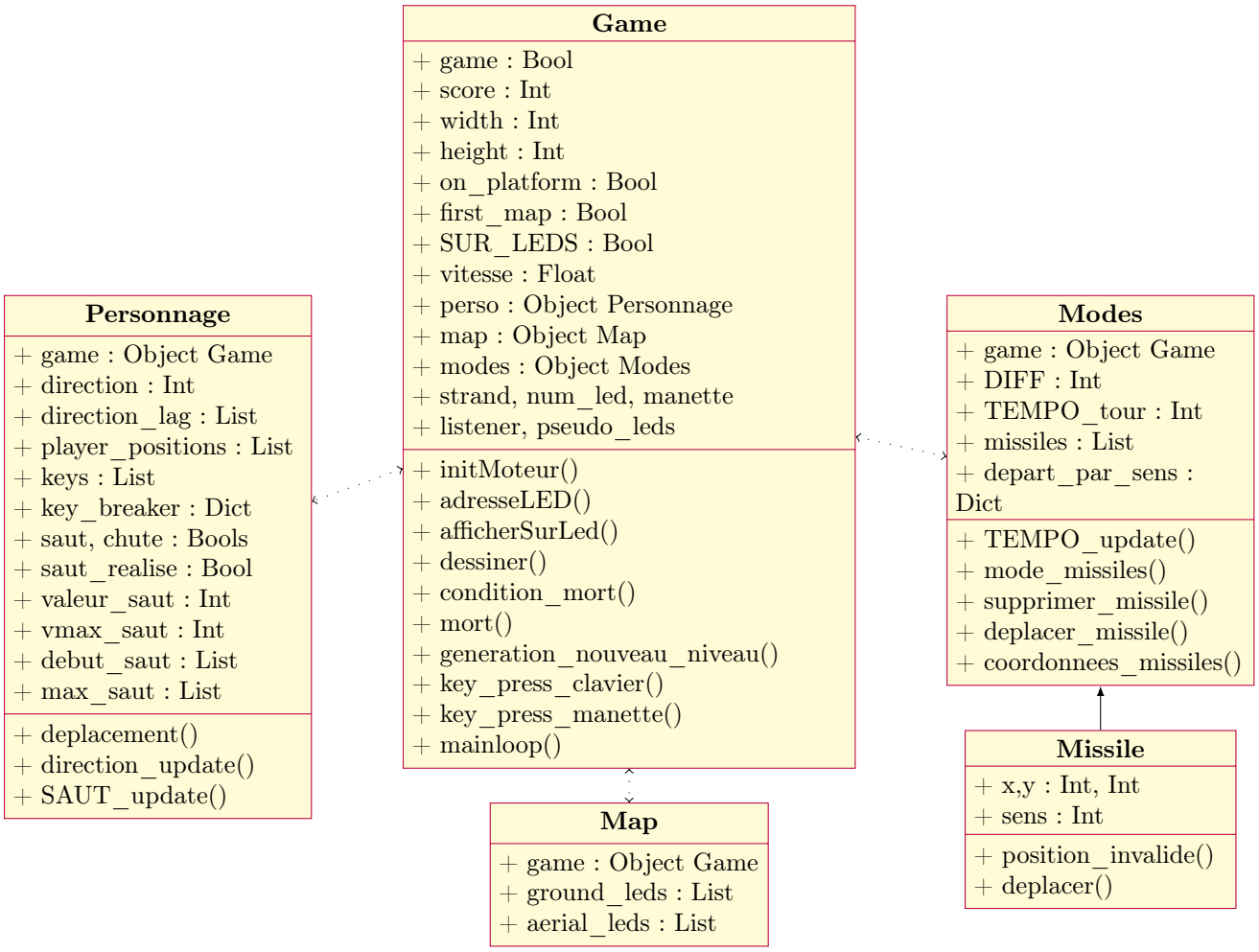
Jumpman est la variante ultime du jeu de plateforme classique Doodle Jump, offrant une expérience de saut sans fin qui teste les limites de votre agilité et de vos réflexes. Dans ce jeu captivant, vous incarnez une mignonne led bleu, qui doit sauter sur des plates-formes en évitant les obstacles pour atteindre des hauteurs vertigineuses. Le gameplay est simple mais incroyablement addictif, avec des sauts fluides et réactifs qui vous permettent de prendre le contrôle total de votre personnage. Les niveaux sont générés de manière aléatoire, offrant un défi sans fin pour les joueurs en quête de records. Si vous cherchez un jeu de plateforme compétitif et addictif qui teste vraiment vos compétences, alors Jumpman est le jeu qu'il vous faut. Alors, préparez-vous à sauter, bondir et voler plus haut que jamais auparavant !

5.1.1 Les règles

Le but est de sauter de plateformes en plateformes afin d'atteindre les plus hauts cieux ! Si vous tombez... Vous perdez ! Il existe différents modes de difficultés :

- Difficultés :
  - S : Missiles
  - D : X

5.1.2 Diagramme UML



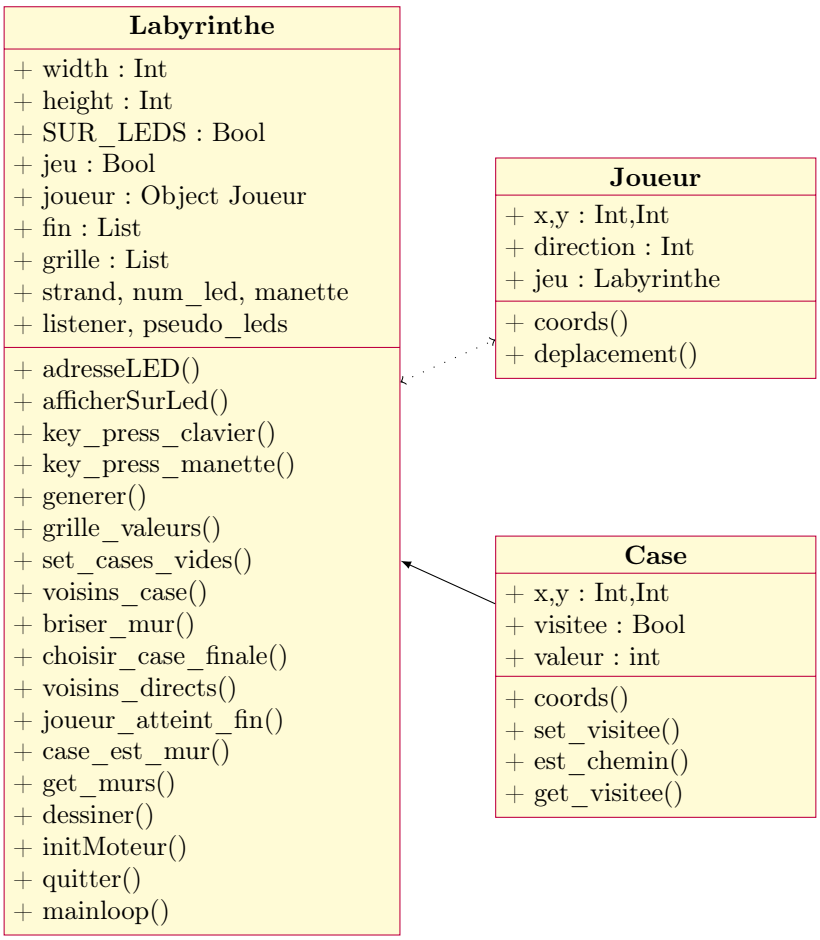
5.2 Labyrinthe

Un simple jeu de labyrinthe, tel qu'on sait en faire depuis des lustres, lesquels, encore, brillent comme ils peuvent. Une entrée, une sortie, des murs, et de ce fait des couloirs, plus ou moins tortueux : vous pourriez avoir de la chance, ou pas. C'est aléatoire (jacta est, etc). Pas de bête à terrasser, mais : un pixel rouge ! Rouge comme un ciel d'août ! Il vous le faudra rejoindre, pour que s'achève la partie par votre victoire, d'autant plus éclatante que vous y aurez passé peu de temps (le score se charge de vous l'indiquer). Le tout sur un cylindre, trouvez-y toutes les métaphores que vous voudrez – j'ai mon interprétation de la chose. Enfin, amusez-vous bien : faites de ce jeu ce que vous voulez tant que vous n'abîmez pas le matériel.

5.2.1 Les règles

Les règles du labyrinthe sont intuitives, une entrée et, peut-être... une sortie. Saurez vous la trouver, le plus rapidement possible, dans une grille à dimension rotative ?

5.2.2 Diagramme UML



5.3 Snake

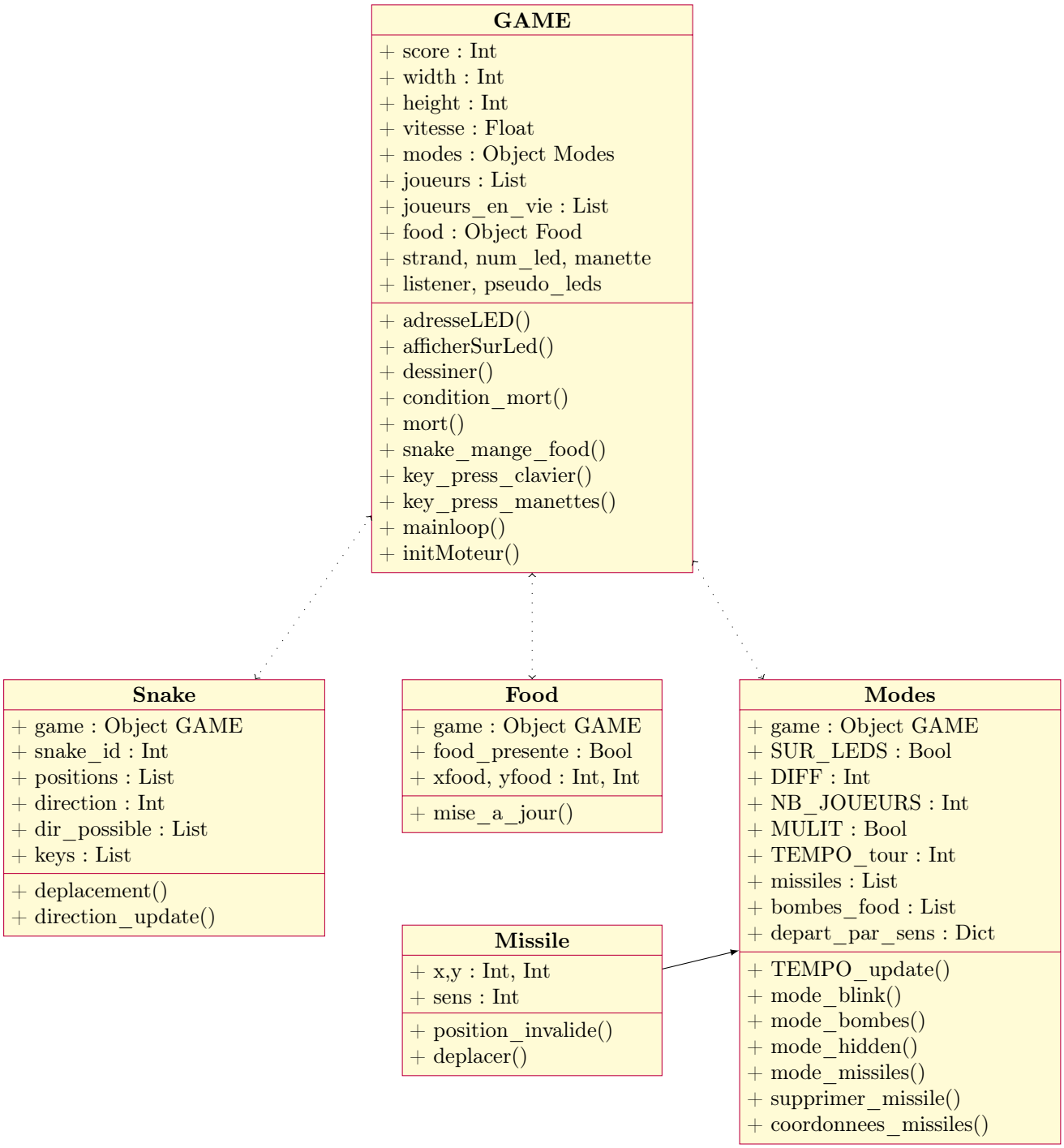
Le jeu Snake est un jeu classique des années 90. Il s'agit de se déplacer, sur une grille bidimensionnelle avec un serpent et de manger le plus de leds. Introduit sur les téléphones Nokia, il est et restera un incontournable du jeu vidéo. Mais qu'en est-il du Snake sur un cylindre ? En effet, grâce à la rotation et à la dimension infinie sur l'axe des ordonnées, le jeu peut parfois prendre une tournure différente. De plus, de nombreux modes inédits ont été développés afin de rendre ce Snake unique et inoubliable.

5.3.1 Les règles

Les règles sont identiques au Snake classique à quelques exceptions près... En effet, même s'il s'agit toujours de manger des pommes (définies par une led bleue) afin de faire grandir le serpent sans se toucher soi ou le bord vertical, différentes difficultés totalement nouvelles ont été développées :

- Liste des options possibles :
  - Missiles : Des missiles de taille 1x1 traversent aléatoirement l'écran horizontalement.
  - *Opposé : snake opposé qui copie les mouvements, [en cours de programmation]*
  - Clignotant : La pomme change de position au bout d'un nombre de tours aléatoire.
  - Bombe : La pomme se transforme en une bombe mortelle au bout d'un nombre de tours aléatoire.
  - Caché : Au bout d'un certain nombre de secondes, la position de la pomme n'est plus indiquée.
- Difficultés :
  - S++ : Missiles, *Opposé*, Clignonant, Bombe, Caché
  - S : Missiles, Clignonant, Bombe
  - A : Missiles, Clignonant
  - B : Missiles
  - D : Aucun mode activé.
- Autres :
  - *M : Super Star, ce mode implique une vitesse du serpent rapide matérialisé par un serpent multicolore. Il agit ou non de manière totalement aléatoire. [en cours de programmation]*
  - M : Mode multijoueur
  - N : Lancer sans LEDs (simulation du panneau avec PyGame). *Cette fonction n'est disponible qu'en lançant le programme depuis un éditeur de programmation, cependant il serait intéressant d'interfacer PyGame dans notre interface web avec Flask par exemple.*

5.3.2 Diagramme UML



6 Utilitaires

Les utilitaires suivants sont fonctionnels. Cependant, il est nécessaire de prendre en compte le fait qu'ils ne sont pas disponibles sur l'interface graphique. En effet, Image Maker et Scribble font appels à, respectivement, l'explorateur de fichier et Tkinter ; nous n'avons pas eu le temps d'optimiser l'interface graphique. L'utilisation de Flask serait par ailleurs préférable pour ces deux programmes.

6.1 Image maker

Image maker permet de transformer n'importe quelle image et de la compresser à la taille du panneau LED.
L'utilitaire se charge de redimensionner l'image à la taille du panneau LED. *Il devra, plus tard, faire attention au ratio de l'image. Pour l'instant, il fonctionne sans prendre cette information en compte.*

6.2 Scribble

Le scribble est un programme permettant de créer ses propres dessins à l'aide d'une fenêtre Tkinter, ses propres "PIXEL art" et de les afficher sur le panneau LED.

6.2.1 Tutoriel

Lorsque le programme se lance, une page Tkinter s'ouvre. C'est une grille, représentant le panneau LED. En haut, se trouvent quatre boutons. Ils permettent de changer la couleur, exporter l'image en png, afficher l'image sur les LEDs, ou supprimer le dessin de la page Tkinter.
L'utilisateur peut donc choisir une couleur et dessiner sur l'interface. La couleur noire servira de gomme. Une fois le dessin effectué, l'utilisateur peut exporter son dessin pour le sauvegarder et l'afficher sur les LEDs.

7 Le serveur web : un serveur local

7.1 Configuration du serveur HTTP

Tout d'abord, nous avons choisi d'utiliser Apache2 comme serveur web pour héberger notre projet. Cela signifie que le serveur peut recevoir des requêtes HTTP (et éventuellement HTTPS) à partir d'un navigateur Web, et qu'il peut répondre à ces requêtes en renvoyant des pages Web.

Ensuite, le lancement de jeux programmés en python, nécessitant un langage de script côté serveur, nous avons dû opter pour CGI (Common Gateway Interface), qui permet au serveur HTTP d'exécuter un script python : 'index_game.py', qui, lui, gère la majorité des opérations et joue le rôle principal dans cette organisation.

Afin de permettre l'exécution de fichiers python, il a été requis de modifier la configuration de Apache2, en ajoutant '.py' à la suite des fichiers exécutés comme 'cgi-script' :

```
AddHandler cgi-script .cgi .pl .asp .py
```

Ainsi, en fonction des requêtes HTTP, le script génère des pages Web dynamiques, en fonction des données fournies dans la requête, et exécute le jeu choisi en temps voulu.

7.2 Pages web

Nous avons également mis en place une page d'accueil statique en HTML. Cette page permet aux utilisateurs de choisir, parmi les 4 jeux proposés, le jeu auquel ils souhaitent jouer. Il est important de noter que quelque soit le choix de l'utilisateur, c'est toujours le même script 'index_game.py' qui s'exécute. Effectivement, c'est en fonction des paramètres GET fournis dans la requête HTTP que le script Python génère la page de configuration pour le jeu choisi.

Les pages de configuration et de résultats sont basées sur des modèles séparés en deux : la partie 'body' qui diffère pour les deux types de pages, et la partie 'header/footer' qui est identique pour les deux. Ces pages sont alors groupées et complétées avec les paramètres nécessaires pour le jeu voulu.

7.3 Page de configuration

Afin de générer une page de configuration adéquate au jeu choisi, nous avons stocké les paramètres de chaque jeu dans un dictionnaire présent dans le script 'index_game.py'. Dans ce dernier, sont présents, pour chaque jeu :

- les noms des formulaires nécessaires (pour le nom du joueur, le mode et la difficulté)
- les formulaires nécessaires sous la forme d'une chaine de caractère (déterminés après initialisation du dictionnaire en fonction des noms donnés)
- les difficultés et modes possibles
- le chemin vers le fichier de score du jeu
- le chemin vers le script du jeu
- la présence d'une sortie pour le jeu

7.4 Note sur le nom des joueurs

Les noms sont générés aléatoirement, car il nous est impossible d'afficher un clavier virtuel considérant la taille de l'écran utilisé, et il nous est encore moins possible d'intégrer un clavier externe de manière permanente au projet. C'est ainsi qu'une liste de 150 noms de personnages fictifs, accompagnée d'une liste d'adjectifs, est utilisée pour créer de nouveaux pseudonymes. Chacun des personnages de cette base possède sa propre histoire et son propre charisme, ce qui les rend uniques et inoubliables qu'il s'agisse d'un héros légendaire, d'un personnage comique, sombre ou mystérieux. Il est donc du devoir du joueur d'adopter cette nouvelle identité et de lui redonner sa gloire en obtenant le meilleur score possible au jeu de son choix !

7.5 Méthode d'exécution des jeux

Une fois le jeu sélectionné et les paramètres entrés, le script 'index_game.py' se basera encore une fois sur les entrées GET pour lancer le jeu avec les bonnes options. Le paramètre 'run' décide de l'exécution : s'il est évalué comme vrai, le script Python lance le jeu en tant que 'sous processus', utilisant la bibliothèque 'subprocess', qui exécute la commande donnée (chaine de caractères) avec la fonction 'run'. Les options sont passées comme arguments dans le lancement du jeu. Ainsi, le script exécute systématiquement une commande de la forme suivante :

```
python [chemin_vers_jeu][jeu].py [difficulte] [mode]
```

D'où l'exemple :

```
python ./jeux/snake.py S++ M
```

Notons que cet appel est bloquant, ce n'est qu'à la fin du jeu que le script principal reprend la main, pour récupérer le score ou les erreurs, et générer une page de résultats adaptée.

8 Construction

8.1 Préambule de construction

Nous sommes heureux de vous présenter notre "structure" de Rotator, un design que nous avons conçu et modélisé avec fierté. Toutefois, nous aimerions vous encourager à apporter vos propres contributions à ce projet en proposant de nouveaux ajouts, que nous appelons les "perks". Nous encourageons les utilisateurs à soumettre leurs idées d'améliorations. C'est pourquoi nous avons placé le projet sous licence "CC BY-SA 4.0", ce qui signifie que vous êtes libre de modifier nos plans selon vos besoins. Veuillez noter que les plans n'ont pas été testés avec une imprimante 3D ou toute autre machine à commande numérique (CNC). Nous vous invitons donc à adapter les plans à votre propre équipement.
Dans la suite de la présentation, nous nous attarderons donc sur **notre** réalisation, en nous efforçant de soumettre des suggestions quant à la réalisation d'une possible nouvelle version.

8.2 Impact écologique

Notre montage a entièrement été réalisé à la main par nos soins. Nous avons utilisé du bois de récupération ainsi qu'une chute de PVC pour le cylindre. Ensuite, l'objet a été réalisé par nos soins à l'aide de scies et de différents autres outils comme par exemple une "Dremel". Cependant, l'impact écologique ne se limite pas qu'à la construction puisque l'utilisation d'un tel matériel est coûteuse en énergie.[3] En utilisant la formule suivante : nombre d'heures d'utilisation par jour x nombre de jours d'utilisation par an x (puissance en watts/1 000), on obtient la consommation annuelle de notre produit. Soit, pour une utilisation d'une heure par jour, on obtient 36,5kW/h, c'est à dire près de 8€ par an (février 2023). A titre de comparaison, la consommation de Rotator est comparable à la consommation d'un téléviseur LCD, il est donc préconisé de ne l'utiliser qu'avec parcimonie.

8.3 Matériel nécessaire

Nous ne sommes ni rémunérés ni affiliés aux différents sites proposés ci-dessous. Ces derniers ne sont proposés qu'à titre indicatif. Par conséquent, nous ne nous tenons pas responsables de leur opérationnalité.

- Raspberry Pi 4 [1]
- BTF-LIGHTING WS2812B Mini Led Individually Addressable Flexible LED Panel 22x22 [2]
- Alimentation 5V - 10A [3]
- Collecteur tournant [4]
- Module afficheur tactile 5", LCD 800 x 480 pixels [5]
- Manette
- Moteur
- Carte SD
- Câbles ("Jump wire" et autres)
- Visserie, Bois ou PLA (prévoir de la peinture pour l'esthétique si nécessaire)

D'autres choses peuvent être ajoutées pour compléter le projet. En effet, nous avons pensé à l'ajout d'enceintes, voir même de micros et de caméras pour de la détection sonore et de la détection de mouvements. Le tout est de repenser la structure afin d'avoir un objet adapté au HardWare.

1. L'utilisation de cartes du même type est possible. Cependant, il est nécessaire de vérifier les performances, en effet, si l'utilisation d'un Arduino est logiciellement possible, nos premiers essais avec une carte Arduino Uno n'ont pas permis d'allumer l'ensemble des deux panneaux. C'est pour cette raison et pour la flexibilité de la carte Raspberry Pi 4 que nous l'avons sélectionnée.
2. L'utilisation de n'importe quel panneau peut faire l'affaire. Il est essentiel cependant d'avoir des leds de type WS2812B afin d'être compatible avec la bibliothèque Neopixel
3. L'ampérage est calculé selon la formule $I = \frac{P}{U}$ soit un ampérage théorique de 10A selon notre panneau led, le montage étant en parallèle (voir schéma).
4. Le collecteur tournant permet d'éviter le phénomène d'enroulement des câbles. Afin de faciliter la construction du module rotatif, l'utilisation d'un collecteur tournant à arbre creux peut être prévilégié. Cependant, ces derniers restent difficiles à trouver et leur prix est très élevé.
5. Nous avons fait le choix de prendre un écran tactile communiquant avec l'interface DSI afin de nous laisser l'ensemble des ports GPIO disponibles.

8.4 Plans de construction

L'ensemble de l'objet a été modélisé sur la version éducative du logiciel FUSION 360 développé par AutoDesk.

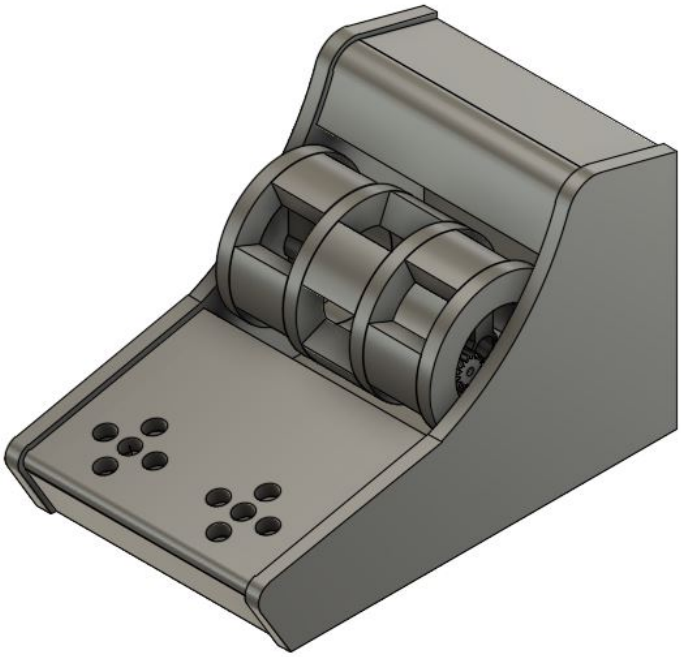


FIGURE 1 – Une modélisation de l'objet

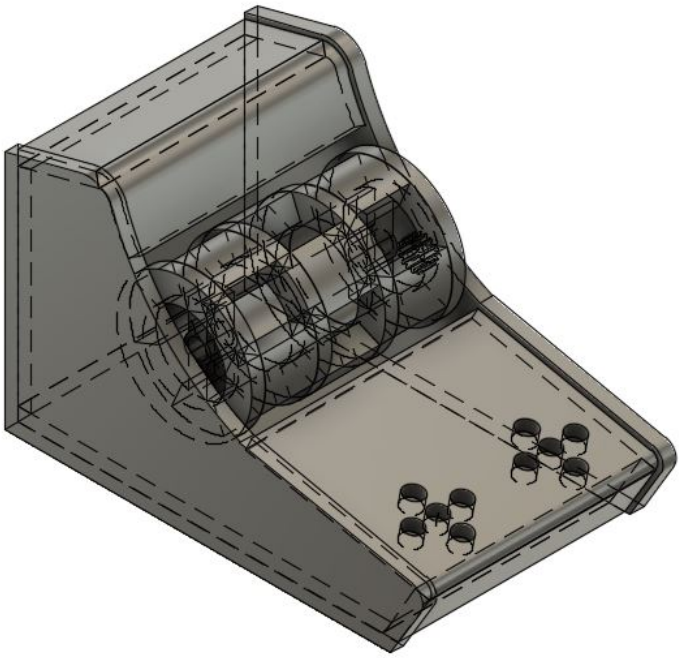
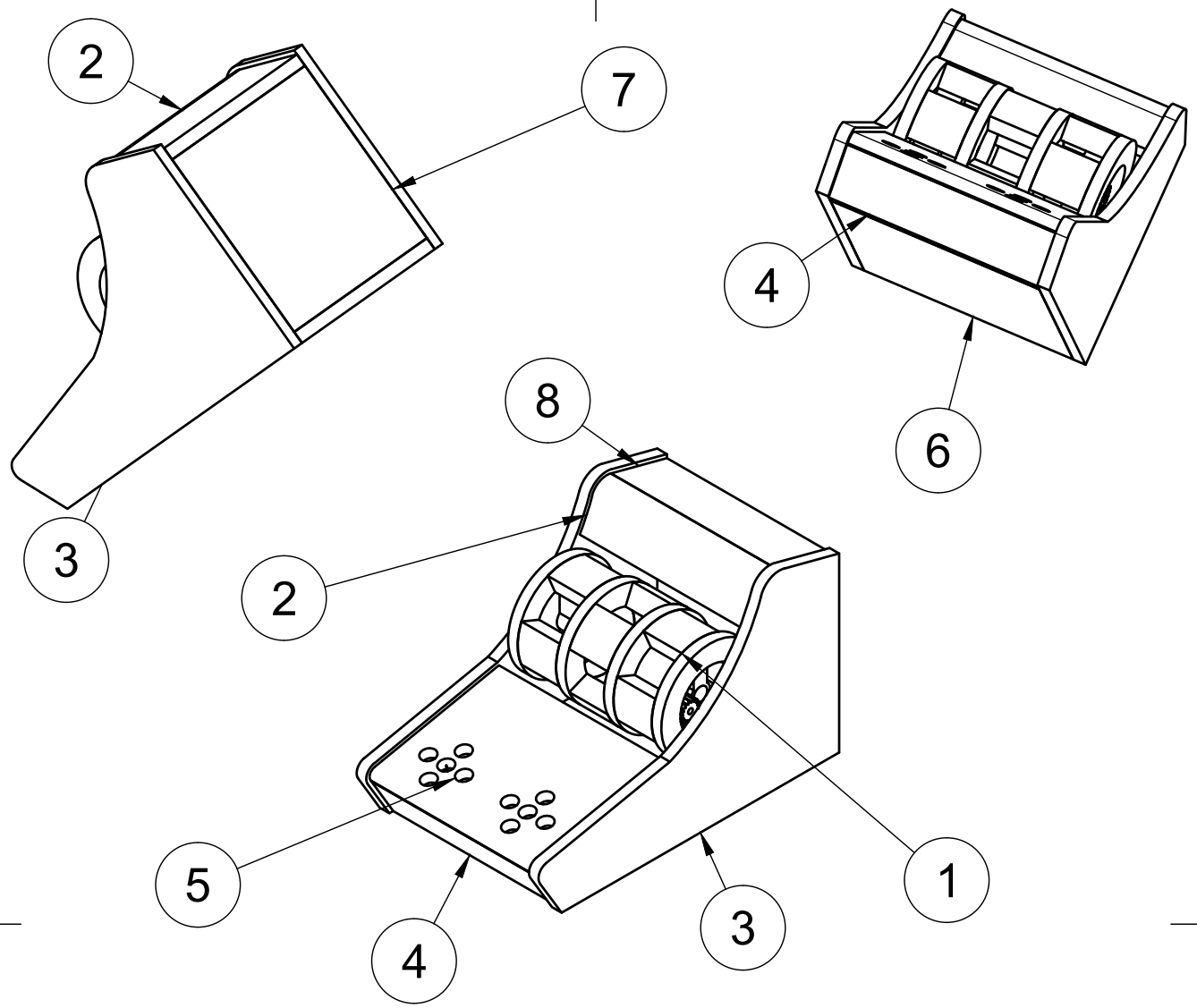


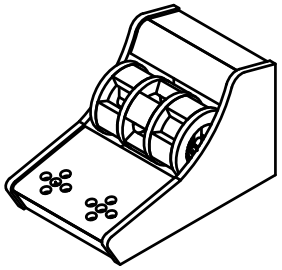
FIGURE 2 – Apparition des perspectives



élément	qté	numéro de pièce	description :
8	1	Plan supérieur	
7	1	Arrière	Arrière de la structure
6	1	Base	Base de la structure
5	1	Plan table de jeu	Écran ou boutons
4	1	Façade inférieure	
3	2	Plan latéral	Deux plans latéraux (voir feuille 3)
2	1	Façade supérieure	Possibilité affichage ROTATOR
1	1	Rouage V6	Support des leds (parties tournantes)

Liste de composants

Dept.	Technical reference	Created by	Approved by
		Da Costa Silva Mathias	



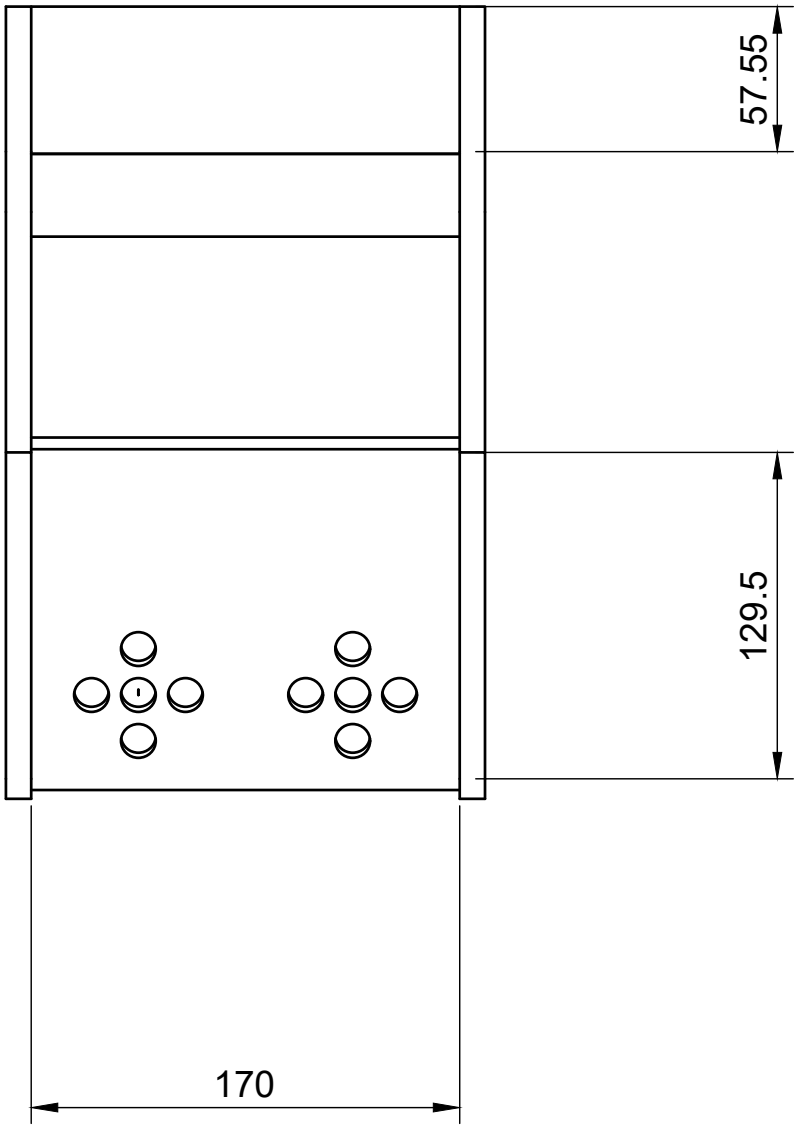
Document type	Document status
Datasheet	Vue d'ensemble

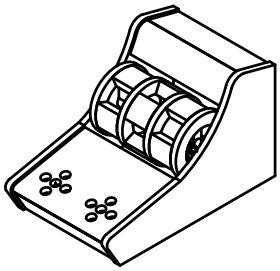
Title

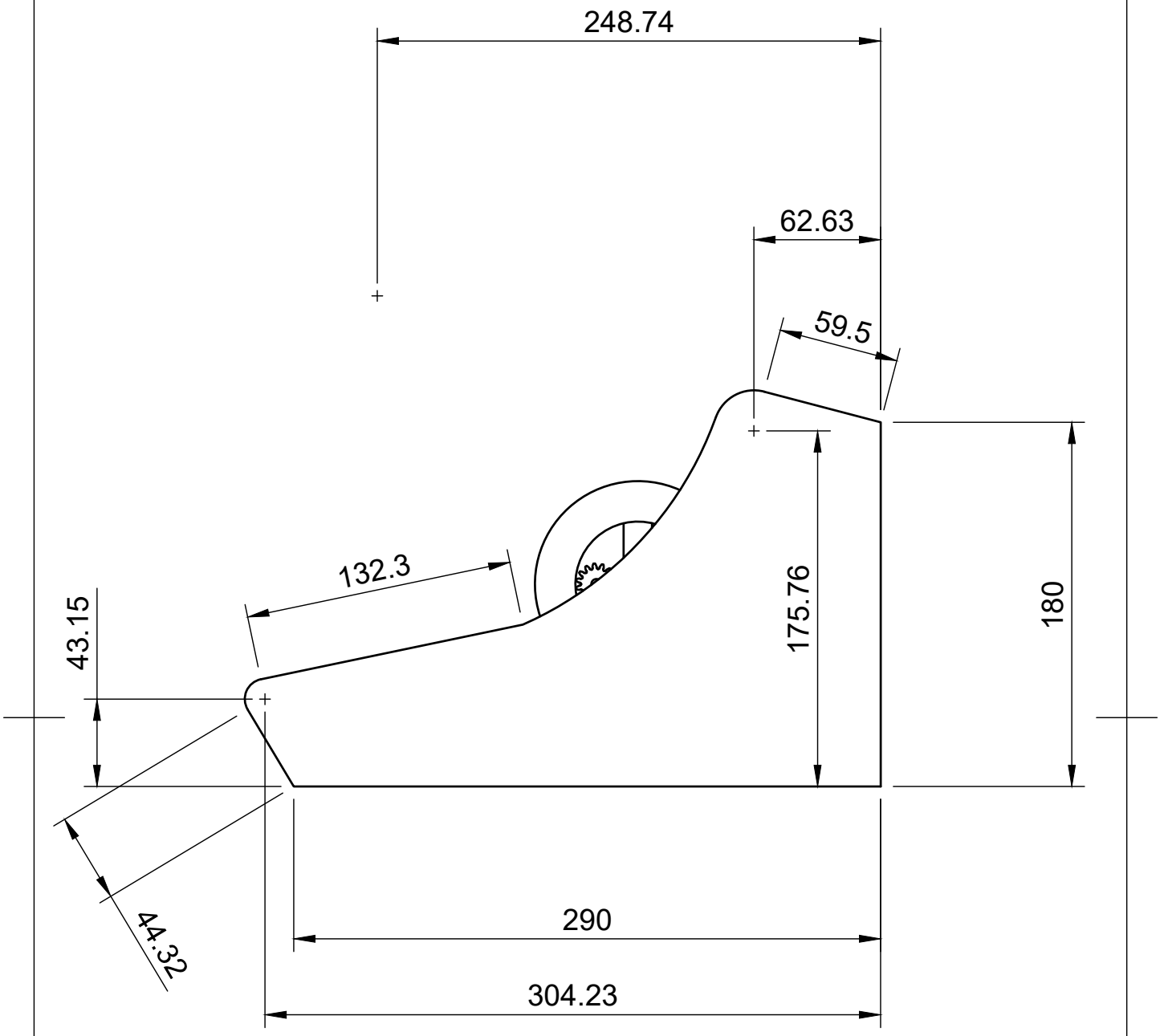
ROTATOR
Plan structure

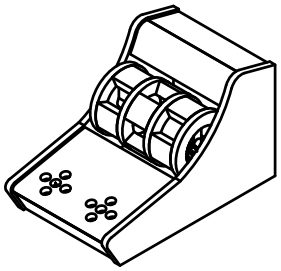
DWG No.

Rev.	Date of issue	Sheet
	22/02/2023	1/3



Dept.	Technical reference	Created by Da Costa Silva Mathias	Approved by	
		Document type Datasheet	Document status Vue de haut	
		Title ROTATOR Plan structure	DWG No.	
			Rev.	Date of issue 22/02/2023	Sheet 2/3



Dept.	Technical reference	Created by Da Costa Silva Mathias	Approved by	
		Document type Datasheet	Document status Vue latérale	
		Title ROTATOR Plan structure	DWG No.	
			Rev.	Date of issue 22/02/2023
				Sheet 3/3

Comme vous pouvez le voir sur les pages précédentes (exemple figure 1), les plans ont d'abord été prévus pour accueillir les boutons, directement sur la borne. Nous avons finalement opté, comme la liste du matériel nécessaire l'indique, pour des manettes. Ce choix a été privilégié pour accueillir l'écran. De plus, pour une question d'ergonomie, les deux joueurs sont plus à l'aise et peuvent se tenir à distance de la borne.

Aussi, il est nécessaire de prendre en compte la largeur de la borne ainsi que des périphériques. Dans notre cas, la Raspberry PI étant fixée sur l'écran via les fixations prévues à cet effet, nous n'avons pas la place pour brancher les deux manettes. C'est pour cette raison que nous avons récupéré un HUB USB non fonctionnel que nous avons réparé. La fragilité de ce type de matériel est souvent due aux soudures de mauvaises qualités et non protégées. Ainsi, il a suffi de dessouder l'ensemble des câbles et de les ressouder pour retrouver un HUB fonctionnel. Une fois les câbles dénudés du câble plus rigide, nous avons été en mesure d'effectuer un angle plus important, ce qui nous a permis de brancher nos deux manettes.

Dans tous les cas, en suivant le schéma ci-dessous, il est possible d'effectuer à l'aide d'USB Mâle et d'USB Femelle, une rallonge à moindre frais. De plus, le démontage d'anciens appareils USB peut permettre de protéger la planète et par conséquent de réduire l'impact écologique de Rotator.

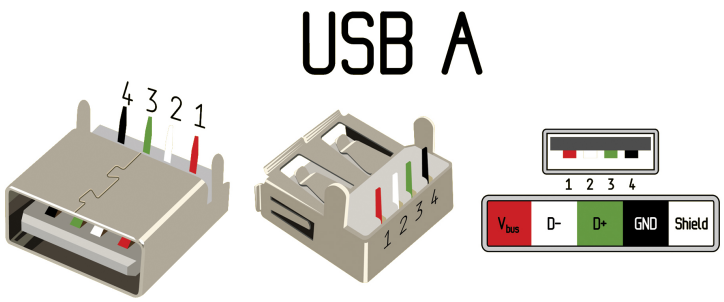


Figure 3 – Pinout USB - D'après un schéma de Rones / openclipart.org

9 Installation

9.1 Les branchements

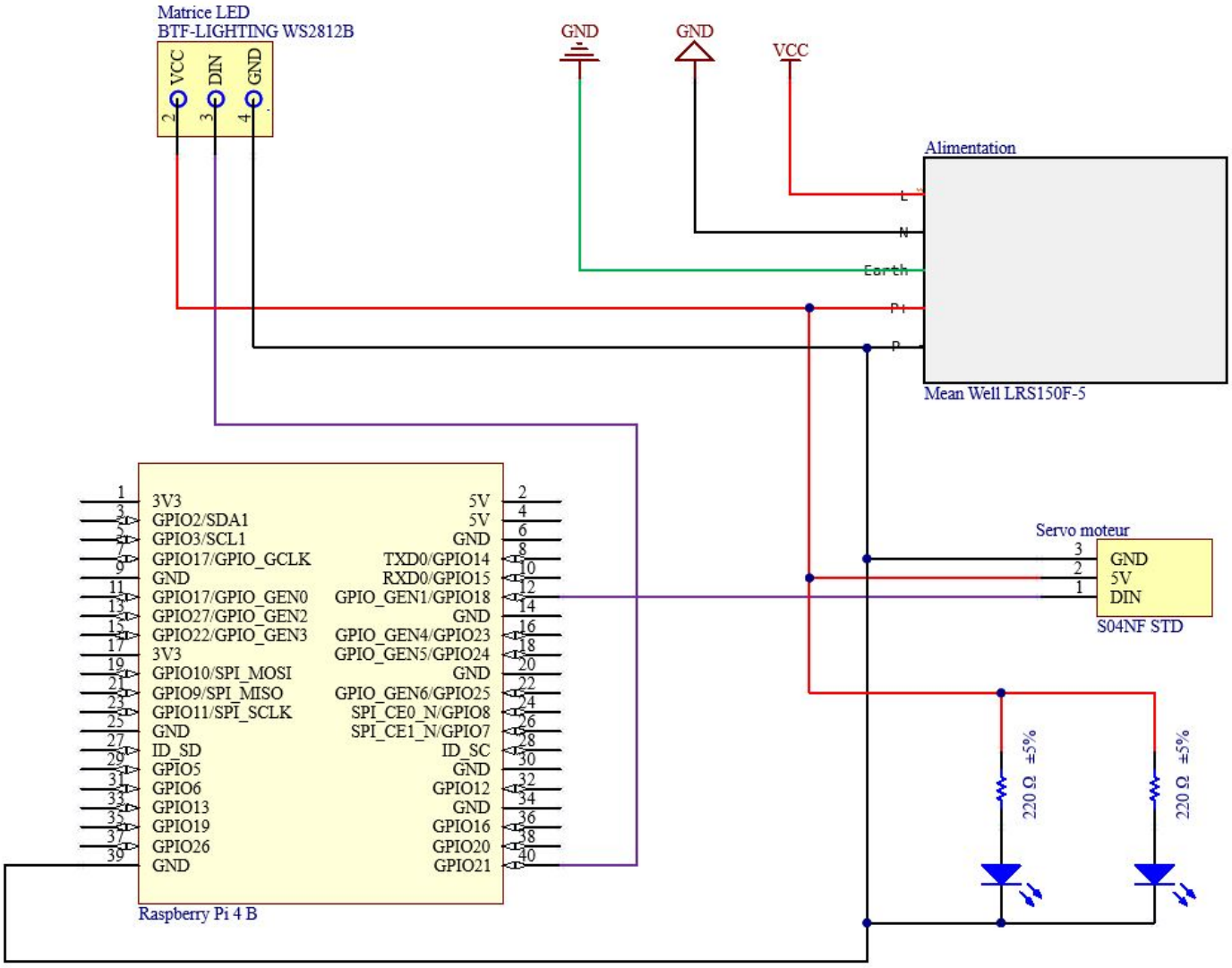


FIGURE 4 – Le schéma électrique de Rotator

À l'aide d'un voltmètre, vous vérifierez que l'alimentation délivre une tension égale à 5.0V. Pour ce type de montage, il est nécessaire d'avoir une alimentation de bonne qualité et de s'assurer que le courant est constant. En effet, la Raspberry PI étant alimentée par les ports GPIO, elle n'est plus protégée. Une sur-tension est donc synonyme de la mort de la carte. À l'inverse, une sous-tension peut nuire aux performances de la carte. Dans ce cas, et en dessous de 4.65V, la carte signale à l'écran être en sous-tension. Si ce problème survient alors que l'alimentation délivre une tension constante à 5V, il peut s'agir des câbles. En effet, les câbles adaptés aux GPIO (jump wire) sont généralement adaptés à du transport de données. Leur section n'est donc pas recommandée pour du transport de courant. Pour faciliter le montage, vous pourrez, comme nous l'avons fait, souder les pins prévus pour les GPIO à un câble d'une section plus épaisse.
En ce qui concerne les différents périphériques n'apparaissant pas sur le schéma comme l'écran et les deux manettes, nous avons choisi de les relier au Raspberry PI via les différents ports existants. En effet, les manettes sont branchées via les ports USB de notre HUB et l'écran quant à lui, est relié via le port DSI qui gère l'alimentation et le tactile !

9.2 Préparation du software : téléchargement des bibliothèques

Tout d'abord, il est nécessaire de "flasher" la carte SD du Raspberry PI. Pour cela, vous pourrez installer le logiciel, depuis un Windows, Mac ou Ubuntu, Raspberry PI Imager disponible à cette adresse : https://www.raspberrypi.com/software/ . Après avoir lancé le logiciel, on choisira, parmis les systèmes d'exploitation disponibles (OS), "Raspberry PI OS (64-bit)". Avec une Raspberry PI différente de la notre, la version 32-bit sera peut-être nécessaire. De plus, il est possible d'installer d'autres OS comme Ubuntu, cependant, Raspberry PI OS étant optimisé et prenant déjà en charge le tactile, nous avons préféré celui-ci et nous ne nous attarderons pas sur les autres. Pour finaliser l'installation, il suffit de choisir le périphérique de stockage. ATTENTION : Il est nécéssaire d'être certain du périphérique utilisé puisque cette opération efface toutes les données présentes dessus.

Après avoir inséré la carte SD et suivi les différentes étapes d'installation, des mises à jours peuvent être à effectuer. Vous pouvez pour cela vous rendre dans le terminal afin de lancer les deux commandes suivantes :

```
$ sudo apt update
$ sudo apt upgrade
```

Ensuite, pour faire fonctionner l'ensemble de nos programmes, il est nécessaire de vérifier que Python 3 et pip3 sont bien installés. Les deux premières lignes permettent d'installer Python 3 tandis que les deux suivantes permettent d'afficher les versions de Python 3 et pip3 vérifiant ainsi que ces deux packages sont bien installés.

```
$ sudo apt install python3
$ sudo apt install python3-pip
$ python3 --version
$ pip3 --version
```

Finalement, des bibliothèques Python sont nécessaires afin de lancer Rotator sans encombre. Vous pourrez les installer via les commandes suivantes :

```
$ sudo apt install git-all
$ sudo pip3 install pygame
$ sudo pip3 install python-dotenv
$ sudo pip3 install pynput
$ sudo pip3 install adafruit-blinka
$ sudo pip3 install adafruit-circuitpython-neopixel
```

Maintenant que l'ensemble des bibliothèques est installé, vous devrez télécharger le projet. Pour cela, vous pouvez, à l'aide du terminal vous placer sur le bureau et créer un dossier ROTATOR. Une fois dans ce dossier, vous pourrez, à l'aide de git, télécharger (ou, en d'autres termes, cloner) le répertoire où est hébergé notre projet.

```
$ cd Desktop
$ mkdir ROTATOR
$ cd ROTATOR
$ git clone https://framagit.org/mxth2xs/Rotator
```

9.3 Installation du serveur local : interface utilisateur

Une fois l'ensemble des bibliothèques Python installé et le répertoire du projet téléchargé, nous pouvons passer à l'étape d'installation du serveur local. Comme expliqué dans la section "Le serveur web : un serveur local", notre interface utilisateur fonctionne sur la base d'une interface web, fonctionnant sur un serveur local. Puisque nous avons décidé d'utiliser Apache2, il va falloir l'installer. Dans un terminal, exécutez les commandes suivantes :

```
$ sudo apt install apache2
$ sudo a2enmod cgi
```

Ensuite, il faut rechercher le fichier "000-default.conf" afin de le modifier. Vous pourrez exécuter la commande suivante. Si aucun fichier n'est trouvé, veuillez vérifier l'installation d'Apache2, ainsi que son répertoire.

```
$ sudo nano /etc/apache2/sites-enabled/000-default.conf
```

Puisque la commande a permis d'ouvrir le fichier avec le logiciel GNU nano, vous pourrez copier/coller les lignes commençant par "<Directory [...]>" et terminant par "</Directory>" à l'intérieur du fichier. ATTENTION : la première ligne incluant le chemin d'accès au dossier ROTATOR doit être modifiée selon votre propre chemin d'accès au dossier. De plus, la ligne "DocumentRoot" doit être modifiée selon les mêmes consignes. Ainsi le fichier ressemblera à :

```
<VirtualHost *:80>
    # The ServerName directive sets the request scheme, hostname and port that
    # the server uses to identify itself. This is used when creating
    # redirection URLs. In the context of virtual hosts, the ServerName
    # specifies what hostname must appear in the request's Host: header to
    # match this virtual host. For the default virtual host (this file) this
    # value is not decisive as it is used as a last resort host regardless.
    # However, you must set it for any further virtual host explicitly.
    #ServerName www.example.com

    ServerAdmin webroot@localhost
    DocumentRoot /home/user/Desktop/ROTATOR/www

    # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
    # error, crit, alert, emerg.
    # It is also possible to configure the loglevel for particular modules, e.g.
    # LogLevel info ssl:warn

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

    # For most configuration files from conf-available/, which are
    # enabled or disabled at a global level, it is possible to
    # include a line for only one particular virtual host. For example the
    # following line enables the CGI configuration for this host only
    # after it has been globally disabled with "a2disconf".
    #Include conf-available/serve-cgi-bin.conf

    <Directory /home/user/Desktop/ROTATOR/www>
        AddHandler cgi-script .py
        AllowOverride None
        Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
        Order allow,deny
        Allow from all
    </Directory>
</VirtualHost>
```

Pour pouvoir jouer aux jeux sans limite de temps, il faut modifier le temps au bout duquel le serveur émet une erreur de réponse "TimeOut". Pour cela, il faudra modifier la ligne TimeOut et mettre une valeur, en secondes, suffisamment grande (exemple : 9000). Pour accéder au fichier :

```
$ sudo nano /usr/local/apache2/apache2.conf
```

Il faudra finalement accéder au fichier "index_game.py" de notre projet afin de modifier le chemin d'accès à Python.

```
$ cd Rotator/www
$ sudo nano index_game.py
```

Avec GNU nano encore une fois, il vous suffira de changer le répertoire d'accès à Python. Par défaut, sur une Raspberry PI il se trouve dans "/usr/bin/". Ainsi, il n'y a pas de modification à effectuer. Cependant, si ce répertoire n'est pas le bon dans ce cas, il faudra le rechercher.

On terminera par donner les autorisations d'exécution au fichier index_game.py avec la commande :

```
$ chmod +x index_game.py
```

Il faudra s'assurer que l'utilisateur www-data a accès aux ports GPIO. C'est en effet sur ces derniers que les LEDs et le moteur sont connectés. Ainsi, avec le terminal :

```
$ chown www-data:www-data -R ./
$ sudo add user www-data gpio
```

Finalement, on redémarrera le serveur grâce à la commande :

```
$ sudo systemctl restart apache2
```

9.4 Préparation des algorithmes : fichier de configuration

Chaque systèmes étant différents, nous avons pris le soin de faire un fichier ".env". Cette extension est utilisée afin d'enregistrer des variables d'environnement. Ces variables servent, par exemple, à défnir le système, ce qui est exactement notre cas.

Afin de les définir, nous reviendrons dans le répertoire parent de Rotator. Dans notre cas, on effectuera la commande :

```
$ cd /home/user/Desktop/ROTATOR
```

Ensuite, à l'aide de la commande Linux "cp", on va copier le fichier. A noter que l'utilisation d'un explorateur de fichiers peut parfois être compliquée lors de cette étape. En effet, les systèmes UNIX sont développés pour considérer ces fichiers comme étant des fichiers systèmes dont la modification pourrait endommager le fonctionnement des programmes. Ainsi, notre fichier .env pourrait ne pas apparaître. Afin de remédier à cela, l'utilisation du terminal est possible. L'utilisation de l'explorateur de fichiers quant à elle, se fera avec la fonctionnalité "Afficher les fichiers masqués".
Revenons à notre terminal. Vous pourrez copier le fichier .env.example servant de fichier modèle en le nommant ".env". Cette nomination particulière est nécessaire puisqu'elle permettra sa lecture par nos différents programmes.

```
$ cp .env.example .env
$ sudo nano .env
```

Grâce à l'ouverture du fichier avec GNU nano, on va pouvoir modifier le fichier. Les variables sont expliquées dans le fichier. Mais, par souci de compréhension, les voici récapitulées :

- DRY : La variable DRY permet de vérifier si un panneau LED est connecté. Cette fonctionnalité bien que peu utilisée, permet à l'utilisateur de profiter des programmes via un éditeur de code comme Visual Studio Code par exemple. En effet, une interface pygame sera, dans ce cas, lancée. Pour l'utilisation d'un panneau LED, l'utilisateur placera la variable à une valeur égale à False, sinon, on mettra la variable à True.
- BOARD : La variable BOARD permet de choisir entre l'utilisation d'une carte ARDUINO ou RASPBERRY. Cette variable a d'ores et déjà été mise en place par souci de prévoyance. En effet, l'utilisation d'une carte ARDUINO n'est pas encore totalement possible.
  La variable sera donc nécessairement égale à True.
- HEIGHT/WIDTH : Ces deux variables permettent de définir la hauteur et la largeur du panneau LED en nombre de LEDs. Dans notre cas c'est un panneau de 22*44.
- HEIGHT_SCREEN/WIDTH_SCREEN : Ces deux variables permettent de définir la hauteur et la largeur de l'écran en pixels. Dans notre cas, nous avons un écran de 880*480.
- PIXEL_PIN : Cette variable correspond au PIN GPIO utilisé pour connecter le panneau LED. Dans notre cas, et comme la Figure 4, l'indique, c'est le port 21. Cette variable est cependant soumise à la syntaxe de la bibliothèque "board". C'est pour cette raison que pour le port GPIO 21, la variable doit être égale à "board.D21".
- ORDER : Cette variable correspond à l'ordre des canaux de couleur de votre panneau. En effet, la majorité des panneaux Neopixel est configurée selon l'ordre GRB (Green, Red, Blue), tandis qu'une autre partie est configurée selon l'ordre RGB. Comme cette variable est soumise à la syntaxe de la bibliothèque Neopixel, la variable sera égale à, dans notre cas, neopixel.GRB.
- PORT : Cette variable n'est utilisée que dans le cadre d'un ARDUINO. Elle n'est donc pas nécessaire pour le moment.

9.5 Préparation de la Raspberry PI

9.5.1 Lancer l'interface au démarrage

Pour lancer automatiquement Chromium avec le lien "localhost/index.html" (qui correspond à l'interface de Rotator) lors du démarrage de la Raspberry Pi 4, il suffit de lancer le terminal avec la commande suivante :

```
$ sudo nano /etc/xdg/lxsession/LXDE-pi/autostart
```

Puis, ajoutez la ligne suivante à la fin du fichier :

```
$ @chromium-browser --kiosk http://localhost/index.html
```

9.5.2 Installation rapide

Si vous n'avez rien compris aux étapes précédentes, pas d'inquiétude, vous pouvez suivre les étapes suivantes :

1. Rendez-vous sur le site https://www.raspberrypi.com/software/
2. Téléchargez le logiciel Raspberry Pi Imager. Il y a plusieurs boutons en fonction de votre système d'exploitation.
3. Une fois le fichier téléchargé, exécutez-le, suivez les instructions de l'utilitaire d'installation.
4. Une fois l'installation effectuée, lancez Raspberry Pi Imager.
5. Téléchargez le fichier IMG disponible dans notre répertoire FramaGit.
6. Sur Rasberry Pi Imager, cliquez sur "Choisir l'OS".
7. Descendez sur la dernière option, "Utiliser une image personnalisée"
8. Choisissez le fichier IMG que vous venez de télécharger.
9. Choisissez la carte SD sur laquelle vous souhaitez installer votre système d'exploitation. Attention, toutes les données seront perdues, veillez à avoir une carte SD vide.
10. Enfin, cliquez sur écrire.

Après avoir inseré votre Carte SD dans la Raspberry PI, vous pourrez la démarrer. Si tout s'est passé correctement, vous verrez l'interface graphique de Rotator.

10 Comment créer son propre programme ?

L'ensemble de notre projet est basé sur la bibliothèque Neopixel, afin de ne pas causer de problèmes de compatibilité par les différentes mises à jours, nous avons choisi de mettre les bibliothèques, en local dans le dossier "lib". De plus, nous avons dû effectuer des modifications dans la bibliothèque dont dépend Neopixel. En effet, nous avons dû commenter la ligne suivante (l.123 - l.124) :

```
if byteorder.strip("GRB") != "":
    raise ValueError("Invalid Byteorder string")
```

Concernant l'affichage des leds, il est nécessaire de comprendre que la disposition des LEDs sur le panneau n'est pas forcément sous la forme d'une grille. En effet, dans notre cas les LEDs sont placés selon une disposition en zig-zag, ainsi la LED 23 correspond à la LED de coordonnées (22, 1). Pour pouvoir utiliser les LEDs avec des coordonnées X et Y habituelles, nous avons codé la fonction adresseLED qui permet de calculer le numéro de la LED en fonction de X et Y.

```
def adresseLED(x,y):
    """inverser a cause de la disposition en zigzag des LEDs.
    (lignes paires et impaires)"""
    if y%2 == 0:
        return ((y*22)+x)
    else :
        return ((y*22)+(21-x))
```

Finalement, l'affichage se fait selon le programme suivant :

```
import time #la bibliotheque time nous permet de faire des temporisations
from rotator.algo.config import strand, NUM_LED, WIDTH, HEIGHT '''cet import n'est pas
obligatoire, cependant, il permet d'importer les parametres du panneau LED'''

def test_led(): #on defini une fonction "test_led"
    for i in range(HEIGHT): #cette boucle permet d'iterer sur l'ensemble des lignes
        for j in range(WIDTH): #cette boucle permet d'iterer sur l'ensemble des colonnes
            strand.setPixelColor(adresseLED(j,i), 255, 0, 0)
            '''l'appel de cette fonction permet de definir la couleur de la LED,
            setPixelColor prend en parametre respectivement le numero de la LED,
            puis la decomposition de couleur au format RGB)'''
            strand.show() #une fois que l'ensemble des LEDs ont ete iteres, on affiche
                                                        le resultat
            time.sleep(0.01) #on temporise afin de voir ce qu'il se passe

test_led() #appel de la fonction
```

Pour ce qui concerne le moteur, il fonctionne selon une modulation de largeur d'impulsions (MLI ; en anglais : Pulse Width Modulation, soit PWM). Nous avons choisi de le contrôler avec RPi.GPIO. Vous trouverez, dans la documentation, un fichier moteur.py où nous avons pris le soin de commenter chaque lignes. Il suffira de faire du threading. En effet, les threads Python sont utilisés pour lancer de multiples processus ce qui correspond parfaitement à notre besoin. Cela évite l'interblocage.

Avec ces informations, vous êtes en mesure de créer, à votre tour, votre propre programme et de l'afficher sur les LEDs.

11 Remerciements

Nous tenions à remercier chaleureusement notre professeur, **Monsieur Loïc Josse**, sans qui le projet n'aurait pas pu voir le jour. Malgré la complexité du projet, les différents obstacles et nos demandes particulières, il a su nous faire confiance et nous accompagner jusqu'au bout. De plus, face à la crise des microcontrôleurs nous n'aurions pas pu nous procurer de carte Raspberry PI. Par conséquent, nous tenions à le remercier vivement pour le prêt de son Raspberry PI personnel.

Nous adressons nos sincères remerciements à l'ensemble de **l'administration du Lycée Louis-le-Grand** qui a accepté de financer notre projet. Aussi, nous les remercions de nous avoir fait confiance en nous accordant, pendant les absences de notre professeur, la salle où était entreposée notre projet.

Nous remercions **l'association des élèves du Lycée Louis-le-Grand**, Maison des Lycéens, pour sa contribution à l'achat des deux panneaux leds.

Enfin, nous exprimons toute notre reconnaissance à nos quelques **collègues** qui ont permis d'apporter un regard différent à ce projet. Avec la tête dans le guidon, il a parfois été difficile de se détacher de nos erreurs.

Nous vous remercions, vous, lecteurs et passionnés. Nous sommes fiers d'avoir fini ce projet et d'avoir pu vous le présenter. Nous restons disponibles pour toutes questions via nos profils LinkedIn.
Ce projet a été l'accomplissement d'un an de Numérique et de Sciences Informatiques. Nous avons pu réutiliser l'ensemble de nos connaissances et des différents langages que nous avons appris. Au-délà de tout, ce projet est la démonstration d'une démarche que nous espérons ingénierique. Nous sommes conscients que **Rotator** n'est pas parfait, mais nous espérons avoir dépassé le caractère divertissant du projet et avoir atteint une certaine forme de "professionalisme".
Encore une fois, nous vous remercions.

Mathias DA COSTA SILVA, Max JOSEPH-ANTOINE,
Van-Kevin NGUYEN et Louise-Anaïs SALVAING