

# Travail de diplôme

---

Documentation Interface Flask

Nicolas Oliveira

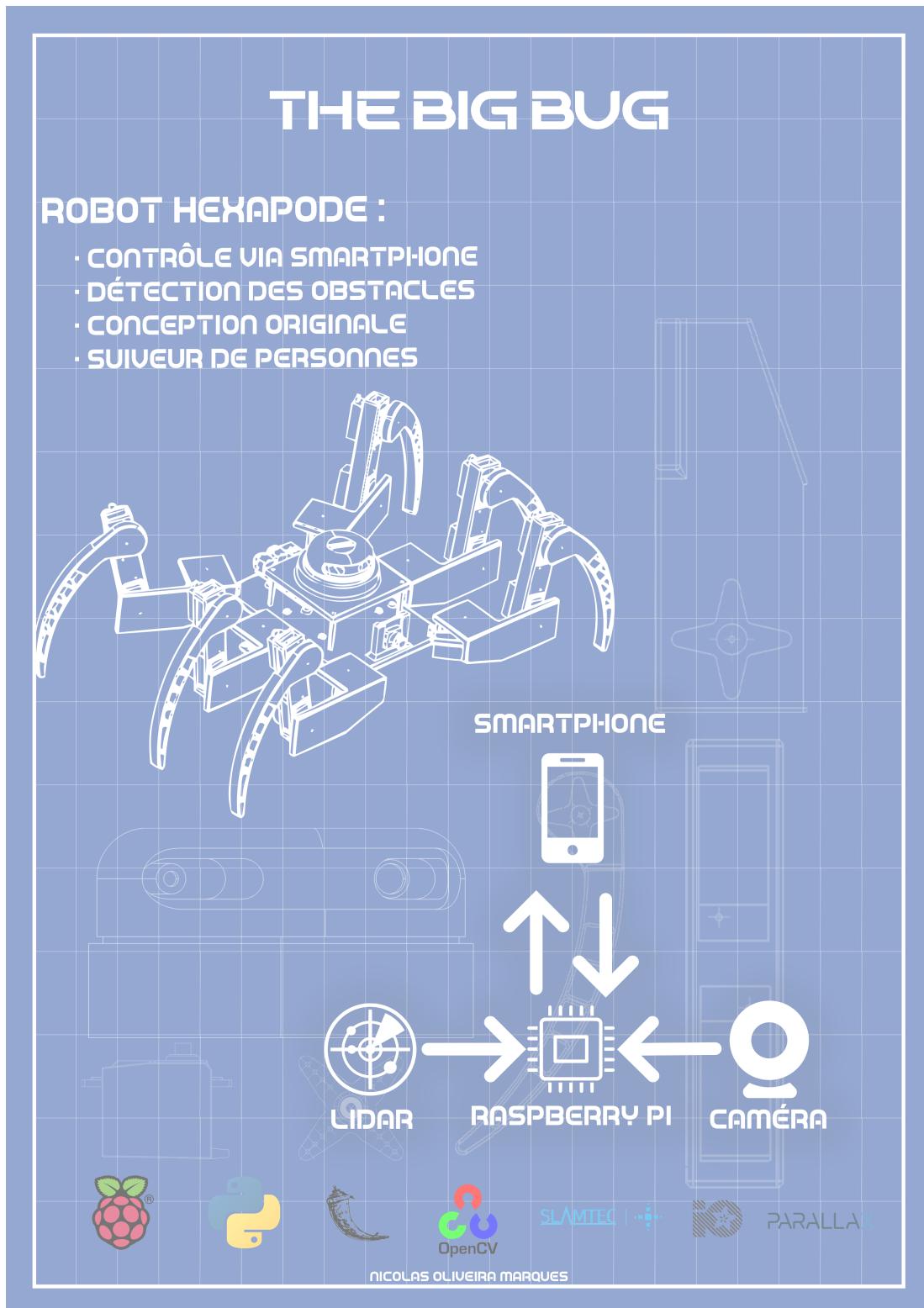
2021-2022 cfpt

## Table des matières

---

|  |    |
|--|----|
| 1. Poster                                | 3  |
| 2. Résumé                                | 4  |
| 2.1 Utilisateur                          | 4  |
| 2.2 Autonome                             | 4  |
| 2.3 Suiveur                              | 4  |
| 3. Abstract                              | 4  |
| 3.1 User-control                         | 4  |
| 3.2 Autonomous                           | 4  |
| 3.3 Followers                            | 5  |
| 4. Documentation technique               | 6  |
| 4.1 Introduction                         | 6  |
| 4.2 Gestion du projet                    | 9  |
| 4.3 Arborescence du projet               | 11 |
| 4.4 Composants                           | 12 |
| 4.5 Les protocoles                       | 19 |
| 4.6 Librairies et outils externes        | 23 |
| 4.7 Analyse des fonctionnalités majeures | 25 |
| 4.8 Tests                                | 34 |
| 4.9 Difficultés rencontrées              | 38 |
| 4.10 Conclusion                          | 40 |
| 5. Cahier des charges                    | 42 |
| 5.1 Sujet                                | 42 |
| 5.2 But du projet                        | 42 |
| 5.3 Spécification                        | 42 |
| 5.4 Restrictions                         | 43 |
| 5.5 Parties prenantes                    | 43 |
| 5.6 Analyse de l'existant                | 43 |
| 5.7 Environnement                        | 43 |
| 5.8 Livrables                            | 44 |
| 6. Annexes                               | 45 |
| 6.1 Manuel utilisateur                   | 45 |
| 6.2 Manuel d'installation                | 54 |
| 6.3 Journal de bord                      | 79 |
| 6.4 Code Source                          | 95 |

## 1. Poster



## 2. Résumé

---

The Big Bug est un robot hexapode (six pattes) pouvant se déplacer dans toutes les directions. Le robot est contrôlé par l'utilisateur via un smartphone, qui peut décider des actions, des animations ou le mettre dans différents modes. Les modes fournis sont :

- Le mode contrôle utilisateur
- Le mode autonome
- Le mode suiveur

### 2.1 Utilisateur

---

Le mode contrôle utilisateur est un mode dans lequel un utilisateur peut contrôler le robot via un site Web à l'aide d'un smartphone ou d'un ordinateur. Sur ce site, il dispose d'un retour caméra, de 4 boutons pour déplacer le robot vers l'avant/l'arrière ou vers la gauche/droite, d'une vue radar des obstacles que le lidar parvient à détecter, et d'une bannière avec les informations les plus importantes, comme l'inclinaison du robot, pourcentage de batterie et obstacle le plus proche.

### 2.2 Autonome

---

Le mode autonome est l'endroit où nous ne contrôlons pas le robot. Il marche et une fois qu'il y a un obstacle, il décide de quel côté est le plus avantageux. Il boucle tout en faisant des animations aléatoires.

### 2.3 Suiveur

---

Le mode suiveur est un mode où grâce à un code QR affiché à la caméra, et à la détection de la distance grâce au lidar, le robot suit la personne à tout moment, si la personne avance, le robot avancera, s'il recule, il recule, etc.

## 3. Abstract

---

The Big Bug is a hexapod robot (six legs) that can move in all directions. This robot is controlled by the user with his smartphone, which can decide actions, animations or put it in different modes. The provided modes are :

- The user-controlled mode
- Autonomous mode
- The follower mode

### 3.1 User-control

---

The user-controlled mode is a mode where a user can control the robot with the website using a smartphone or a computer. On this website, there is a camera feedback, 4 buttons to move the robot forward/backward or left/right, a radar view of the obstacles that the lidar manages to detect, and a banner with the most important informations, such as the robot's inclination, the percentage of battery charge left and nearest obstacle.

### 3.2 Autonomous

---

The autonomous mode is a mode where we do not control the robot, it walks, until there is an obstacle. It decides which way is better. While deciding, it walks in a circle and doing some random animations.

### 3.3 Followers

---

The follower mode is a mode where with a QR code, shown to the camera, it will follow a person at any time and using the LiDAR, the range needed to catch up to the person. If the person goes forward, the robot goes forward, if they goes backward, it goes backward, etc.

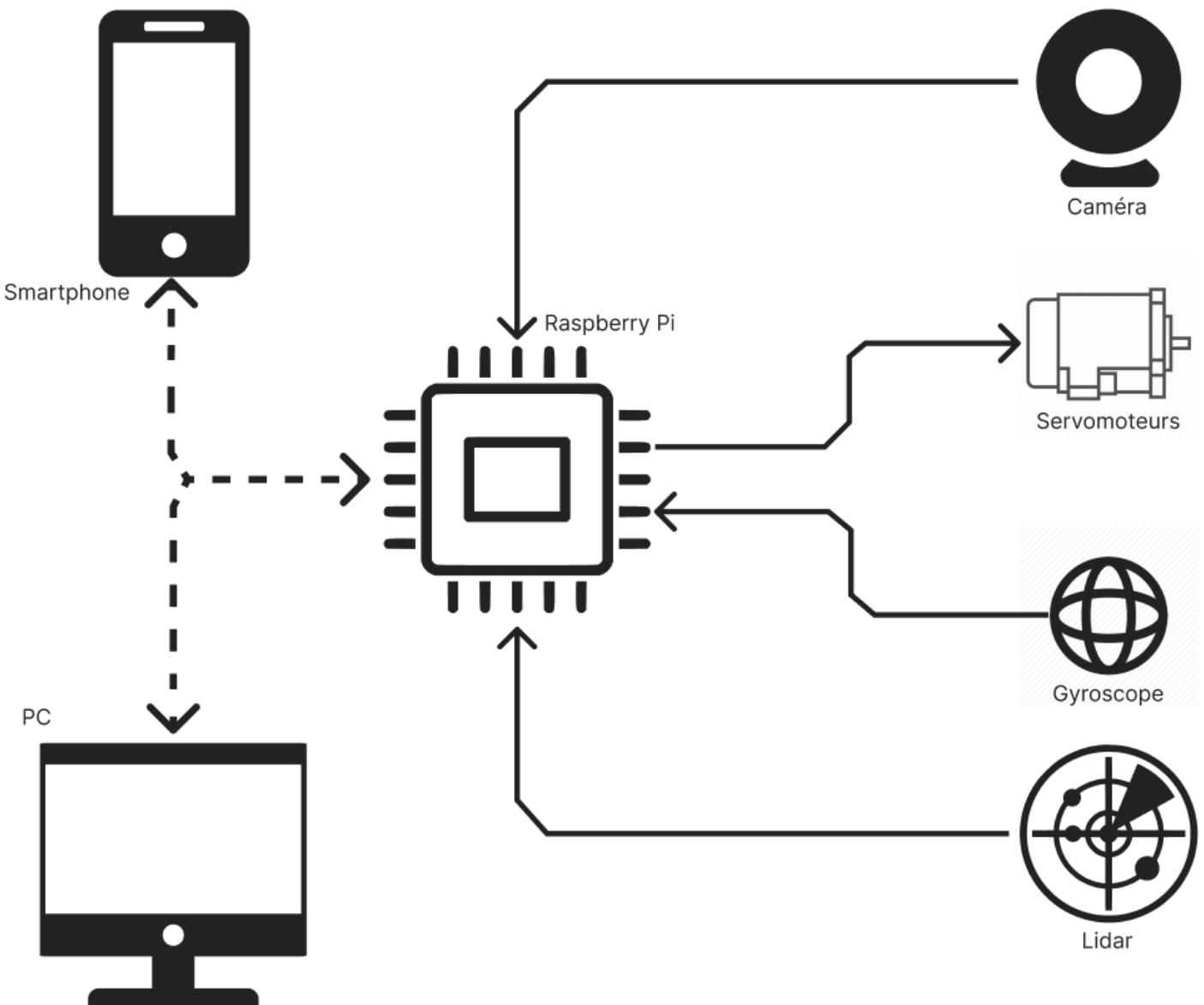
## 4. Documentation technique

### 4.1 Introduction

The Big Bug est un robot hexapode contrôlé à distance, qui grâce à plusieurs capteurs, peut se déplacer dans toutes les directions. Ce projet est séparé en deux grandes parties.

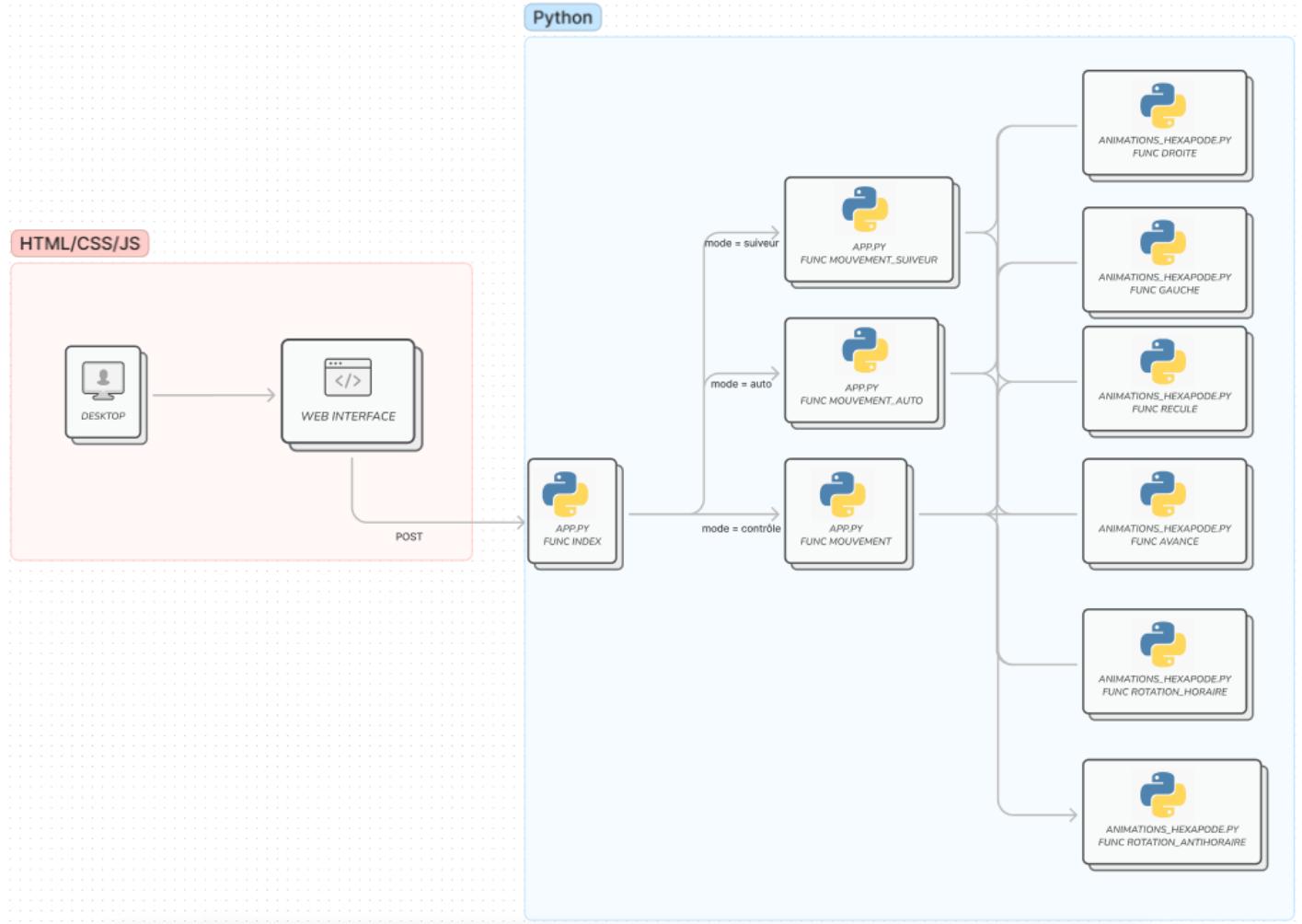
La première, la conception du robot. Cette partie englobe tout ce qui est la création mécanique du robot, l'assemblage, et programmer l'intelligence de celui-ci. La deuxième est la gestion du robot. Cette partie prend en compte l'interface web, la communication entre les classes, la récupération des données, et les utiliser.

Ci-dessous, voici un diagramme démontrant l'architecture de communication en général. L'utilisateur (représenté à gauche par un smartphone ou par un pc) se connecte au raspberry pi grâce à l'access point, puis à l'interface web qui sert à faire la gestion du robot, avec qui on peut contrôler la direction de la marche du robot, et récupérer les données des capteurs (représentés sur toute la partie de droite), pour ensuite les afficher sur l'interface. Pour chaque capteur, le raspberry pi communique avec le protocole approprié ce qui favorise la vitesse la communication. Puis, la communication entre l'utilisateur et l'interface se fait grâce à WSGI (Web Server Gateway Interface) qui sert à faire une interface entre des serveurs et des applications web pour le langage Python.



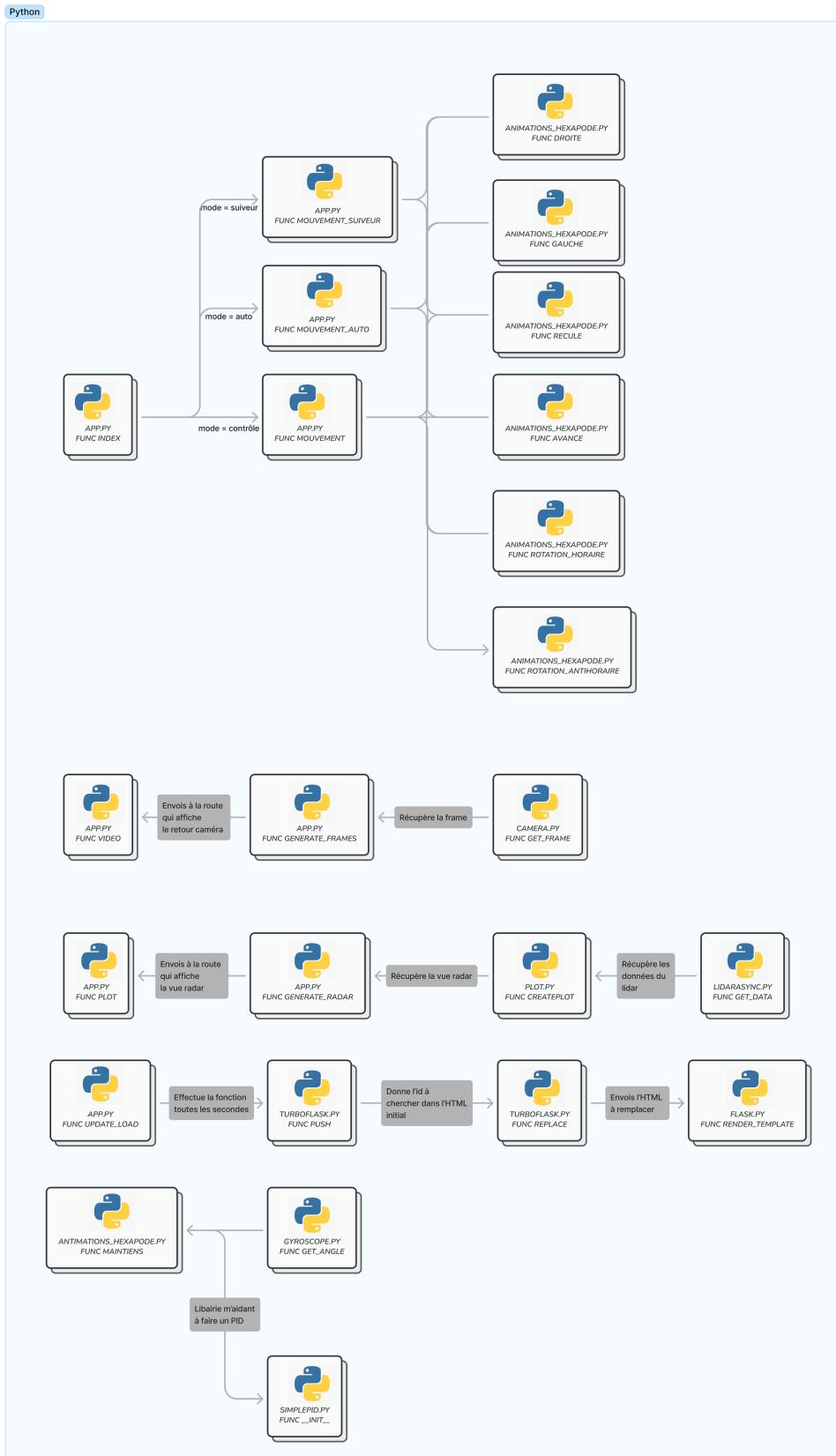
Le frontend de l'application est une page web, affichée localement. Pour cela, j'utilise les services HTTP de Flask présents sur le raspberry pi. La page web communique ensuite dans les deux sens à travers le code Python afin d'interagir avec le backend Python et afficher les résultats à l'utilisateur et/ou faire bouger le robot comme l'aura souhaité l'utilisateur.

Le backend est composé d'une fonction qui attend les requêtes Post et dépendamment de la requête, il réagira comme souhaité voici un diagramme qui représente cela:



Ce système permet de contrôler selon l'interaction de l'utilisateur, les modes changent, et dépendamment du mode, soit l'utilisateur contrôle les mouvements, avec le clavier ou en ayant un QRCode où le robot le suivra, sinon dans le mode autonome. Le raspberry pi se sert des données que lui rend le lidar, pour savoir dans quel sens il doit aller. Il fera une analyse des distances et verra où il a le plus de place pour marcher et si la plus grande distance pour marcher.

Dans ce projet, afin que ce soit plus simple pour s'y retrouver j'ai essayé de séparer un maximum les différents éléments du backend. Il y a les éléments que nous avons vus précédemment qui nous servent pour le mouvement de l'hexapode. Ensuite, nous avons la partie qui retransmet le retour caméra. À la suite, nous avons tout l'acheminement de la création de la vue radar, avec la récupération des données du lidar, qui sont envoyées à la classe qui créer le dessin de la vue. Ensuite, l'envoi sur le script principal qui l'enverra à la fin sur la page web. Ensuite, il y a toute la partie qui sert à mettre à jour les valeurs récupérées des capteurs qui sont affichées sur la bannière du haut. Et enfin, la partie qui gère le PID du robot, afin qu'il puisse se maintenir droit.

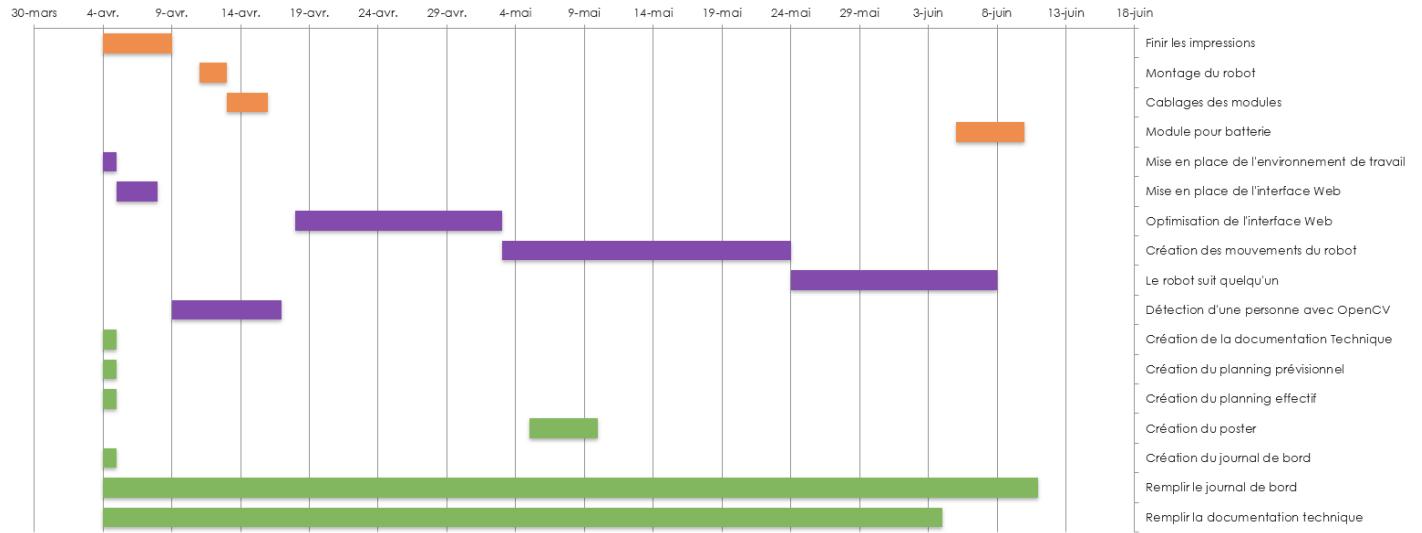


Ensuite pour plus de précision de comment fonctionnent mes classes indépendamment, je vous laisse allez voir [ici](#).

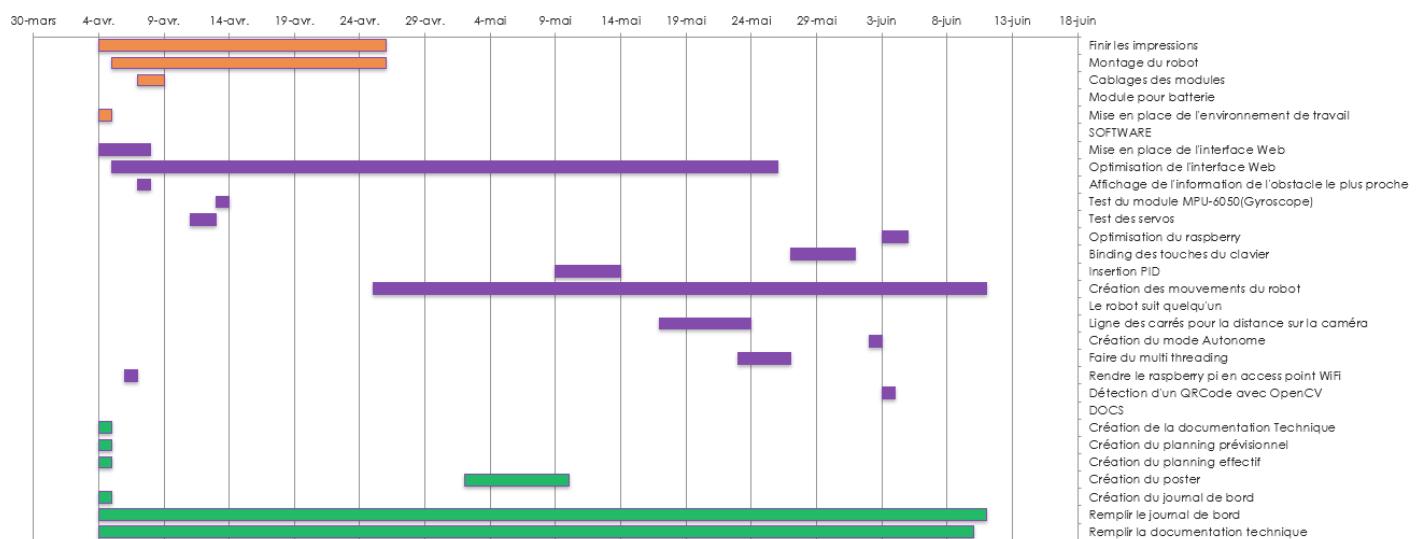
## 4.2 Gestion du projet

### 4.2.1 Planning

Afin de pouvoir faire le planning prévisionnel, j'ai utilisé un diagramme Gantt qui se présente comme ceci:



Dans ce planning, j'ai mis les grandes étapes que je pensais les plus utiles mais comme vous pouvez voir en comparaison avec le planning effectif, beaucoup plus de tâches ont été ajoutées. Et ce manque de préparation m'a porté préjudice durant la conception du robot.



### 4.2.2 Gestion du projet

Durant la réalisation de ce projet, j'ai utilisé l'outil Trello. Trello m'a servi à utiliser une méthode Kanban. Ceci marche en faisant des cartes qui représentent les tâches que l'on doit faire. Puis, les cartes, il faut les mettre selon l'avancement de la tâche. Pour mon cas, j'ai séparé ça en trois parties :

- À faire
- Commencée
- Finie

Voici comment est représenté mon Trello :

Tableau ▾ **Suivis de progression TD** ⚡ Ecole 🔍 Visible par l'espace de travail 📁 Partager ⏱ TimeCamp ⏲ Power-ups ⚡ Automatisation ⏮ Filtre ⏮ Afficher le menu

**À faire**

- Module pour batterie
- Repenser la solidité de certaines pièces
- Détection d'une personne avec opencv
- Robot suit une personne

**Commencée**

- REMPLIR le jdb
- REMPLIR la documentation technique
- Création des mouvements

**Finie**

- Mise en place de l'environnement de travail
- Mise en place de l'interface Web sur le projet TD
- Création de la documentation Technique
- Création du planning prévisionnel
- Création du planning effectif
- Câblages des modules
  - 1 04:15
- Rendre le raspberry pi en acces point WiFi
  - 1 00:49
- Insertion de l'information de l'obstacle le plus proche, sur la bannière
  - 1 00:14
- Création du journal de bord
- Test des servos

**Notes**

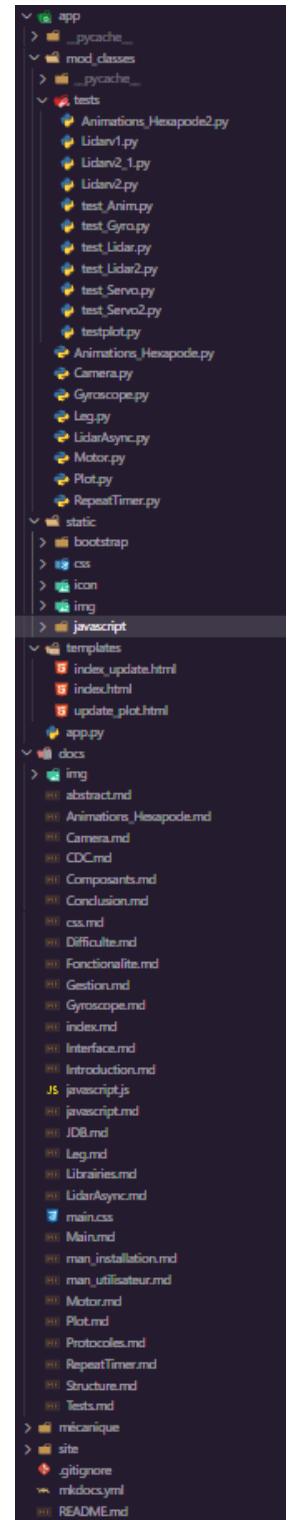
- HARDWARE
- SOFTWARE
- DOCS

## 4.3 Arborescence du projet

Du fait que le projet est réalisé grâce à la librairie [Flask](#).

Voici la structure pour ce projet:

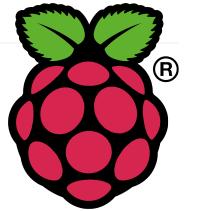
- **app/** Ce dossier contient le code de l'application.
- **app/mod\_classes/** Ce dossier contient les classes du projet.
- **app/mod\_classes/tests** Ce dossier contient les scripts de test.
- **app/static** Ce dossier contient tous les éléments qui ne seront pas modifiés.
- **app/static/bootstrap** Ce dossier contient tout ce qui est en rapport avec bootstrap sur les vues.
- **app/static/css** Ce dossier contient le css du site.
- **app/static/icon** Ce dossier contient l'icône du site.
- **app/static/javascript** Ce dossier contient le script javascript du site.
- **app/static/img** Ce dossier contient les images utilisées sur le site.
- **app/templates** Ce dossier contient les vues du site, puis le morceau d'html avec lequel on rafraîchit le site.
- **docs/** Ce dossier contient la documentation du projet. À la racine, il y a le journal de bord, la documentation technique, ainsi que le cahier des charges.
- **docs/img** Ce dossier contient les images utilisées pour la documentation, ou pour le cahier des charges.
- **mecanique** Ce dossier contient tout les fichiers qui ont servis à la mécanique du robot.
- **site** Ce dossier contient la documentation transformée en site web.



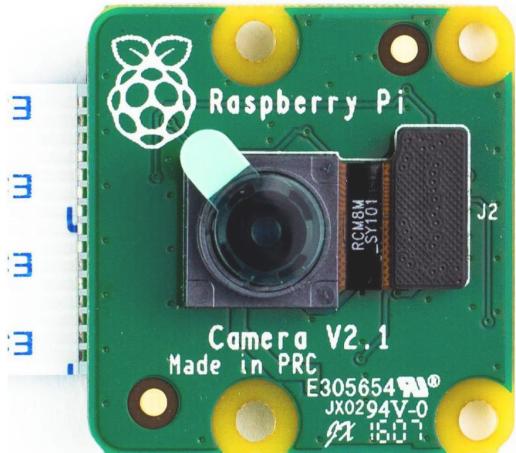
## 4.4 Composants

### 4.4.1 Raspberry Pi

Le Raspberry Pi est un nano-ordinateur monocarte à processeur ARM de la taille d'une carte de crédit conçu par des professeurs du département informatique de l'université de Cambridge dans le cadre de la fondation [Raspberry Pi](#).



### 4.4.2 Pi camera



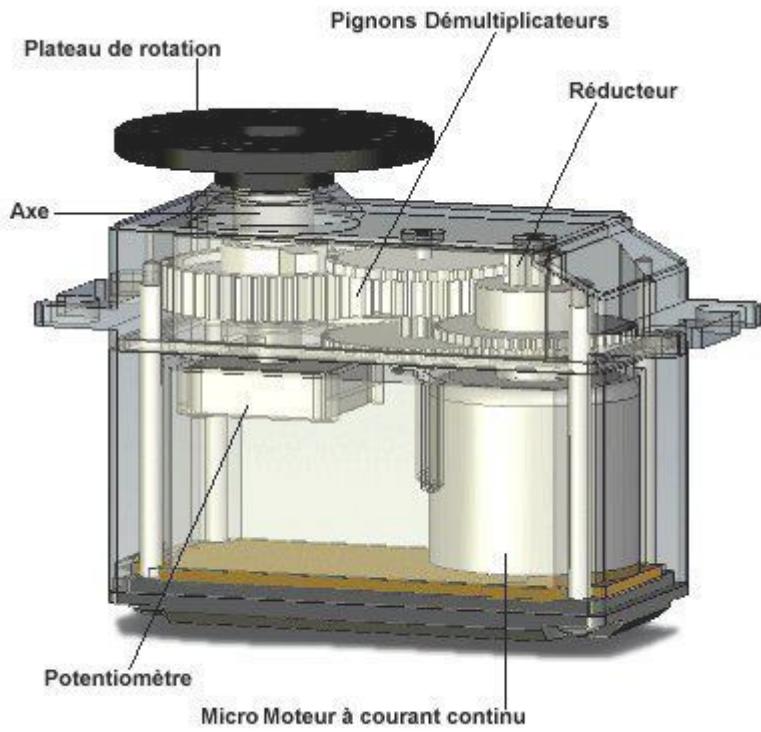
Ceci est un module de chez [raspberry pi](#) qui me permet d'avoir un retour vidéo constant sur le robot. Ce module comprend un capteur Sony 8-megapixel. Du fait que c'est raspberry pi qui l'a conçu, il n'y a qu'un ruban à brancher sur le raspberry pi.

## 4.4.3 Servomoteurs Parallax

PARALLAX

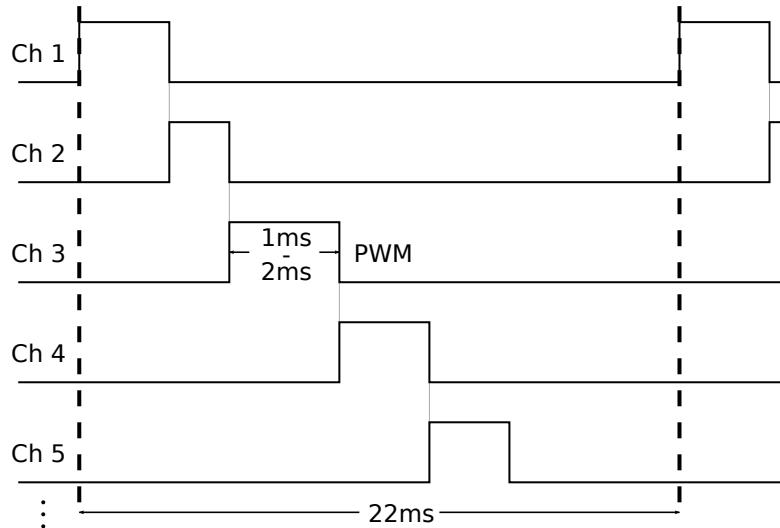


Un servomoteur est un moteur à courant continu de petite taille, avec un réducteur à la sortie, afin de diminuer la vitesse, tout en augmentant le couple. Puis, dans les servomoteurs standard, il y a un potentiomètre afin de contrôler la position, mais dans les parallax 900-00008 il n'y a pas, car ce sont des servomoteurs, à rotation continue.



Eric G

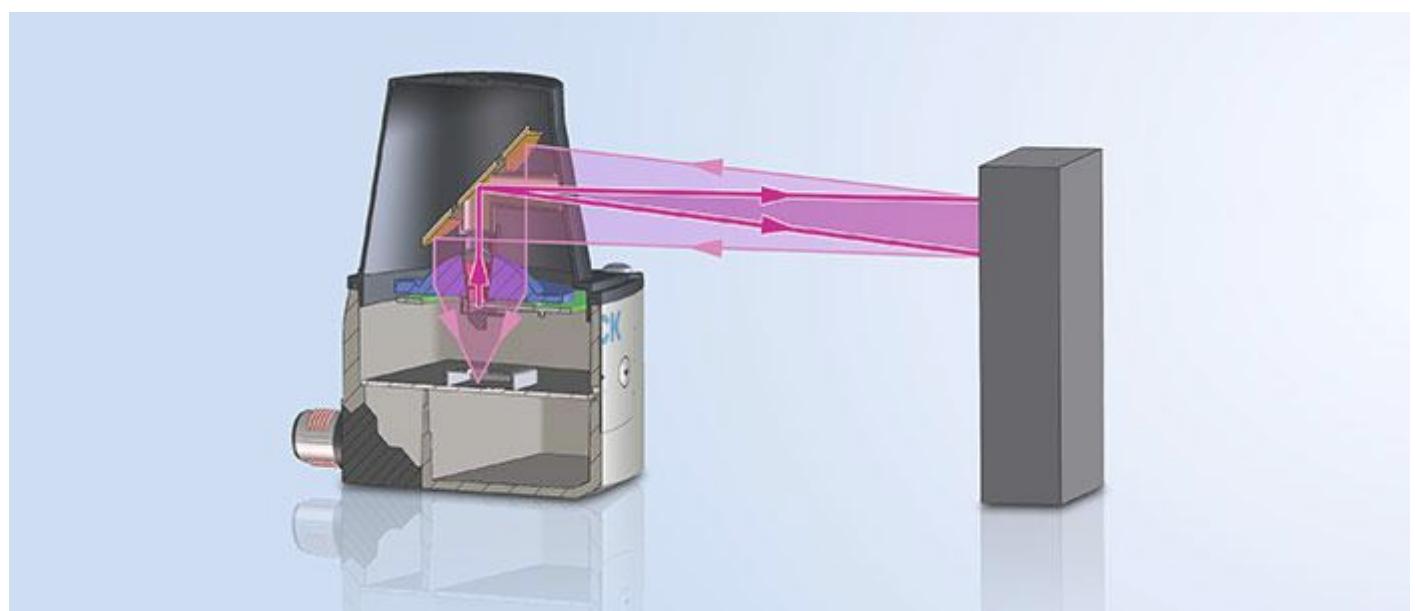
De ce fait, avec les servomoteurs standard on n'a qu'à mettre un angle, puis il y reste, alors que les nôtres, il faut le dire dans quel sens tourner, la force à avoir, et quand l'arrêter. La communication avec le servomoteur ce fait grâce au PWM, il faut lui envoyer une impulsion entre 1,3 [ms] et 1 890 [ms], 1 500 [ms] étant le milieu, en dessous ça fait tourner dans un sens, au-dessus dans un autre sens.



#### 4.4.4 RPLidar

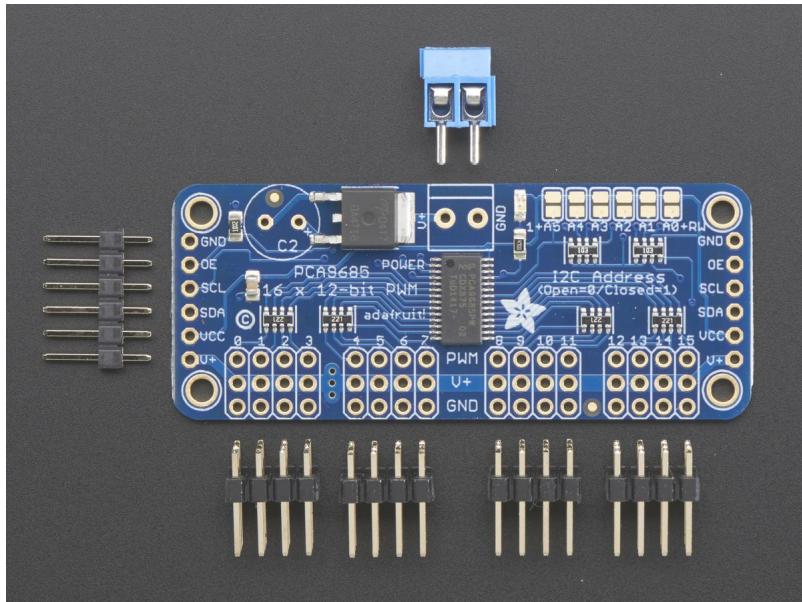


La télédétection par laser ou lidar est une technique de mesures à distance fondée sur l'analyse des propriétés d'un faisceau de lumières renvoyé vers son émetteur.



Le lidar que j'ai utilisé vient de chez [Slamtec](#). De la catégorie RPLidar, j'ai utilisé le modèle A2, qui a une portée de détection de 0,2 m à 16m, ce qui est utile afin d'avoir un robot qui se déplace de manière autonome. Tension à 5V, parfait pour le monter avec un raspberry pi. Enfin, la communication utilisée est le protocole [UART](#).

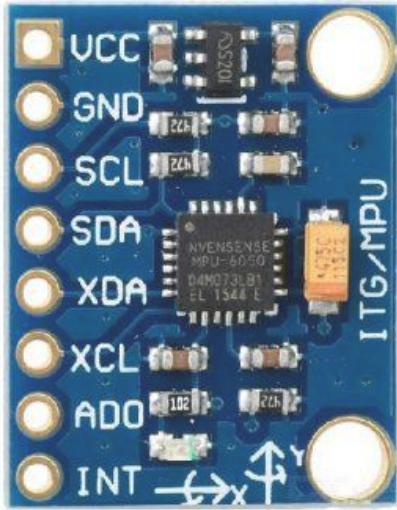
#### 4.4.5 PCA9685



Ce module est un contrôleur de Servomoteurs PWM à 16 canaux.

Ce module me sert à pouvoir plus facilement contrôler plusieurs servomoteurs en même temps. Il vient de la marque [adafruit](#) . L'un des points positifs et le fait que j'ai choisi ce module, c'est qu'il ne consomme pas sur le raspberry, pour les servomoteurs, il a une alimentation différente, chose qui n'est pas à négliger quand on travaille avec autant de servomoteurs. Ce module communique avec le raspberry grâce au protocole [I2C](#) ce qui me facilite le câblage, en pouvant mettre les câbles en série.

#### 4.4.6 MPU6050



Ce module MPU-6050 qui dessus à un gyroscope ainsi qu'un accéléromètre est improprement nommé gyroscope, car ce n'est pas un «vrai» gyroscope avec les anneaux, mais c'est un MEMS (MicroElectroMechanical Systems). Le fonctionnement de ce MEMS est qu'avec des mécanismes micrométriques réalisés sur silicium, elles sont mises en mouvement grâce aux forces générées par des transducteurs électromécaniques (dispositif servant de convertir un signal physique en un autre), et ce dernier fait l'interface entre la mécanique et l'électrique, et un circuit récupère ce signal et le transforme en signal numérique.

Ce module me permet d'avoir la position du robot à tout moment, ce qui m'est très utile afin de faire un PID pour le maintenir droit après les animations. Le gyroscope de ce module a une portée de mesure de  $+/-250^\circ$  à  $+/-2000^\circ$ , le robot n'étant pas censé se retourner complètement, j'ai choisi de mettre à la valeur la plus basse. Ensuite, l'accéléromètre du module a une portée de  $+/-2\text{ G}$  à  $+/-16\text{ G}$ , pour le projet je ne me sers pas de l'accéléromètre pour l'instant. Contrôler en I<sub>2</sub>C et prenant une tension de 3V3 ce module est parfait pour raspberry.

## 4.4.7 CR10-S pro v2

**CREALITY**

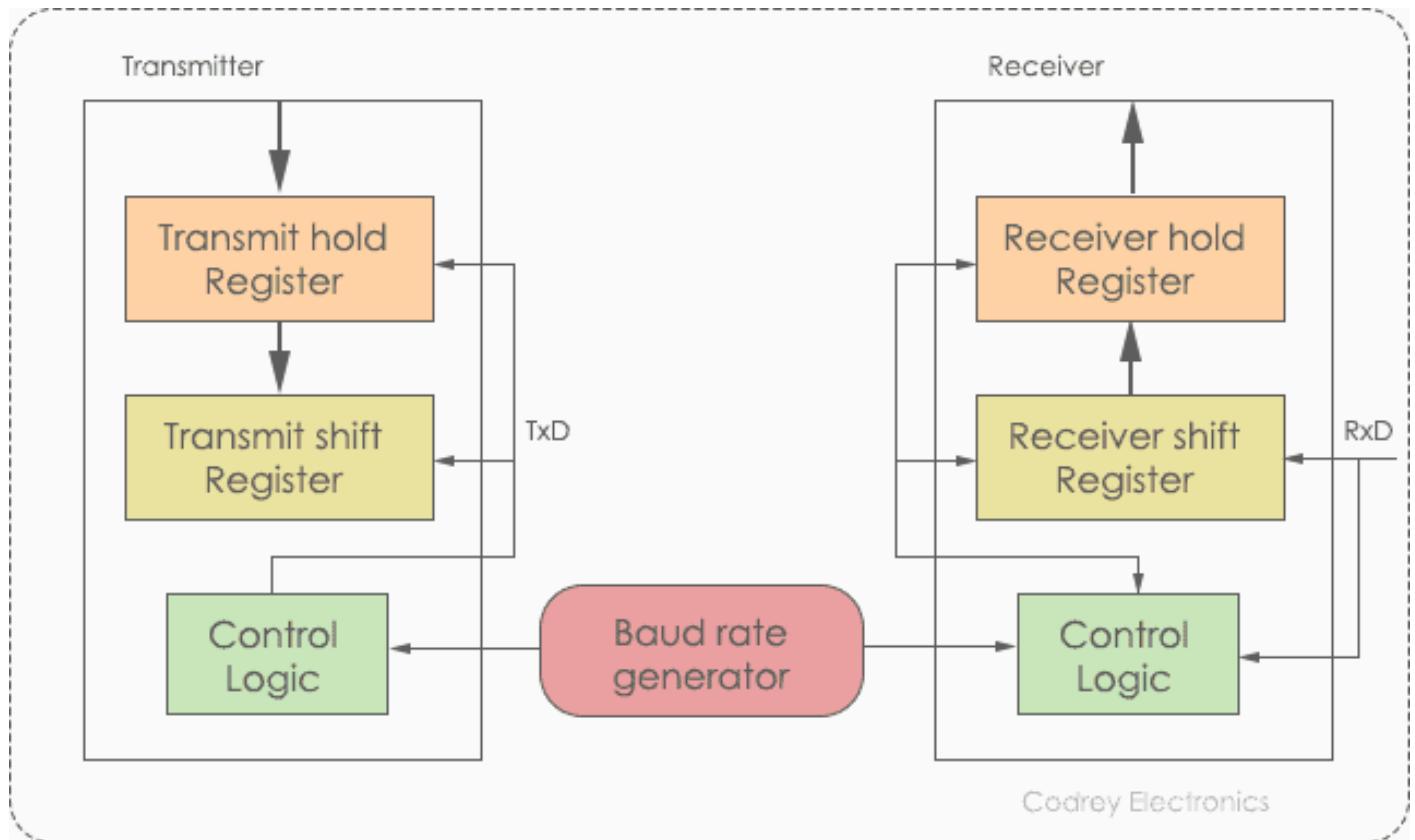
Imprimante 3D permettant d'imprimer les pièces du robot.

L'imprimante de la marque [Creality](#) a été choisie, car ayant une grande surface, je n'allais pas être importuné pour la taille de mes pièces.

## 4.5 Les protocoles

### 4.5.1 UART

L'UART (Universal Asynchronous Receiver-Transmitter) qui pour résumer est un protocole de communication composé d'un émetteur et d'un récepteur. L'émetteur se constitue d'un registre de maintien d'émission, registre à décalage d'émission et de la logique de contrôle. Pour le récepteur, c'est la même chose sauf qu'il n'y a pas de transmission, c'est de réception. En commun, l'émetteur et le récepteur sont dotés d'un générateur de vitesse de transmission.



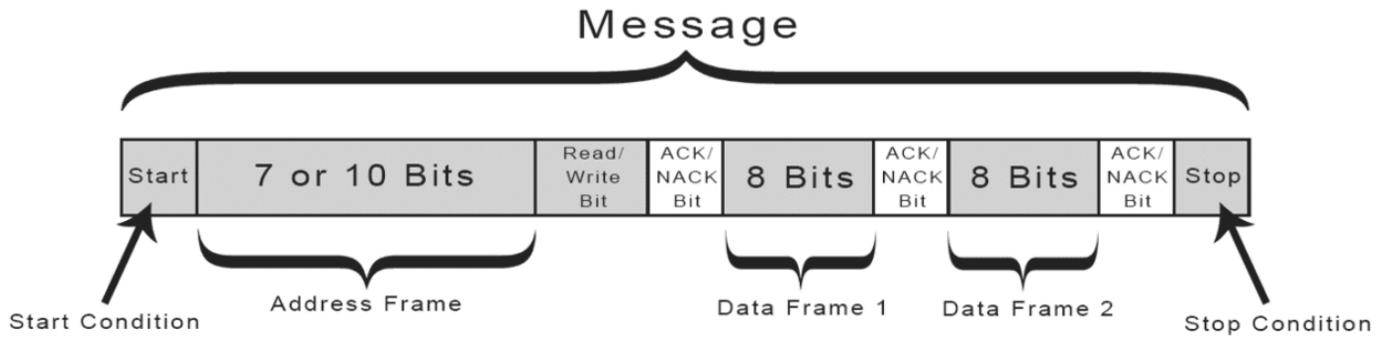
Le raspberry pi lui envoie une trame de données avec 7 bits d'information, qui est comprise par le lidar, et fait une action précise. La trame doit avoir un bit de début et ceux de fin de communication, car s'il en manque un des deux, soit l'information n'a jamais été écoute, soit le périphérique est en continu entrain d'écouter.



### 4.5.2 I2C

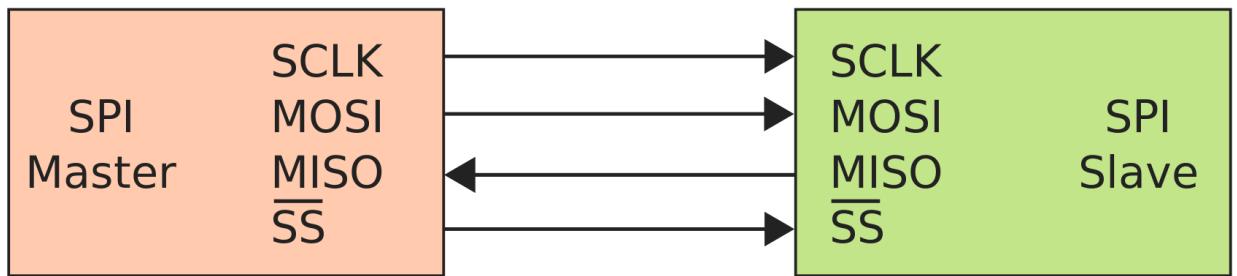
L'I2C (Inter-Integrated Circuit), c'est un protocole de communication qui sert beaucoup, si nous avons beaucoup de modules, car nous pouvons chaîner les modules. Ce protocole utilise des adresses afin de s'adresser aux modules, par exemple le gyroscope

est l'adresse 0x68, puis nous lui envoyons en data le registre qui dit au module ce qu'il doit faire. Puis, le module nous renvoie une réponse, par exemple avec le gyroscope il nous renvoie les angles.

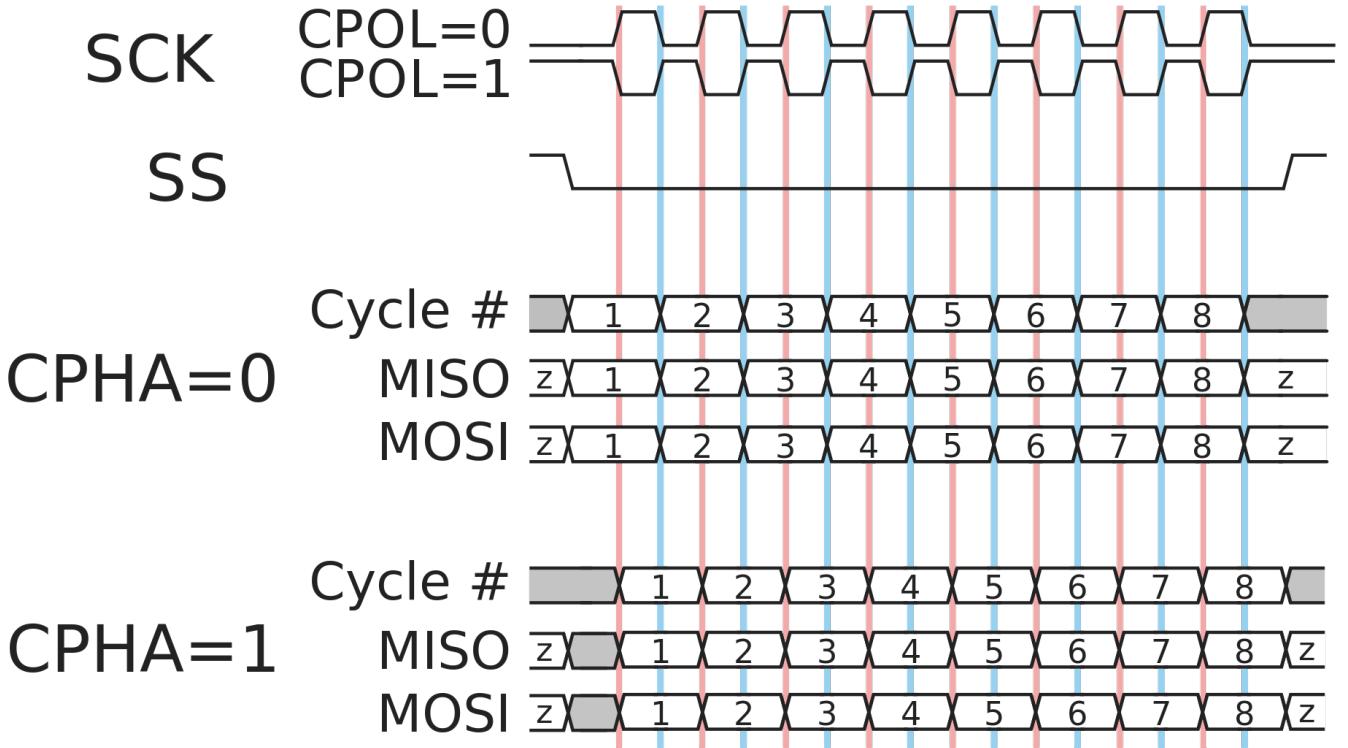


#### 4.5.3 SPI

La communication SPI (Serial Peripheral Interface) est un protocole de communication Master/Slave. Master étant le raspberry pi et le Slave étant le lidar.

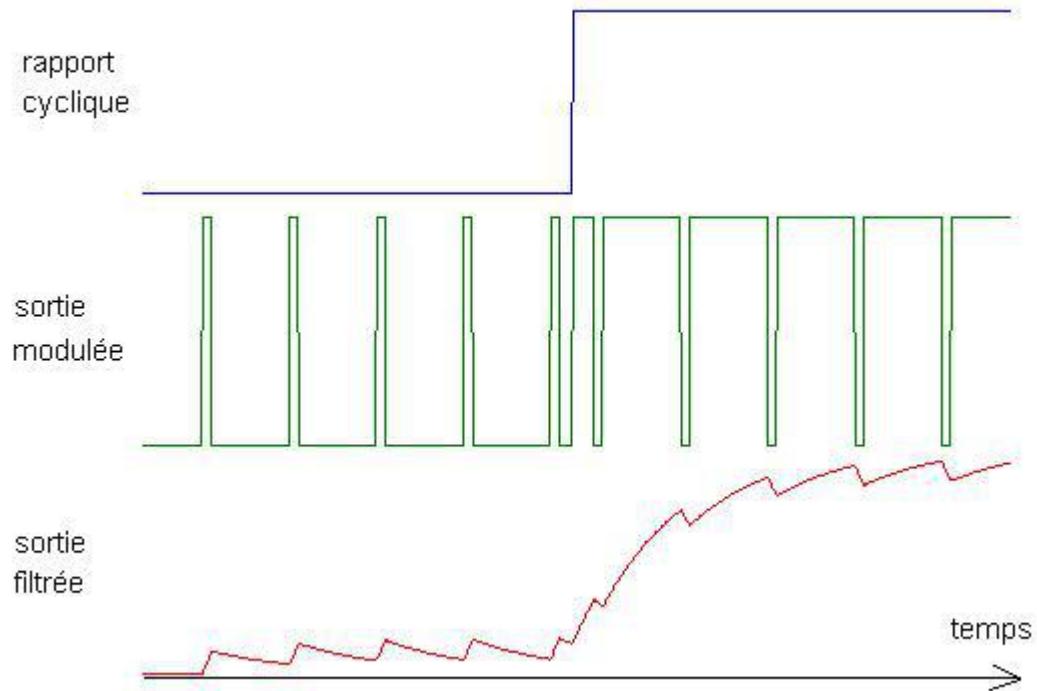


Le raspberry pi lui envoie une trame de données avec un registre, qui est comprise par le lidar, et fait une action précise. La trame doit respecter le clock(horloge) qu'envoie le raspberry pi, car sinon des bits vont se perdre et le message est incomplet.



#### 4.5.4 PWM

Le PWM (Pulse Width Modulation) est une technique couramment utilisée pour synthétiser des signaux pseudoanalogiques à l'aide de circuits numériques.



## 4.6 Librairies et outils externes

---

### 4.6.1 Python Flask

Flask est un framework web, qui permet de rédiger les applications web avec du Python. Flask a été créé par Armin Ronacher, membre de Pocoo, un groupe de développeurs Python formé en 2004 — le 1er avril 2010. Pour cette application j'ai utilisé Python 3.8.10.



#### Utilisation

Pour l'utilisation voici un lien vers mon [code source](#)

### 4.6.2 Turbo Flask

Turbo\_flask est une extension Flask qui intègre la bibliothèque JavaScript turbo.js à l'application Flask. Le programme utilise majoritairement la fonction push qui actualise juste la partie souhaitée de la page.

#### Utilisation

Pour l'utilisation voici un lien vers mon [code source](#)

### 4.6.3 OpenCV

OpenCV (Open Source Computer Vision Library : <http://opencv.org>) est une bibliothèque open source qui comprend plusieurs centaines d'algorithmes de vision par ordinateur.



#### Utilisation

Pour l'utilisation voici un lien vers deux scripts qui l'utilisent: [Camera](#) et [Plot](#)

### 4.6.4 Matplotlib



Matplotlib est une bibliothèque Python qui permet de dessiner des diagrammes. Elle est utilisée pour la visualisation de données et le traçage de graphiques.

#### Utilisation

Pour l'utilisation voici un lien vers mon [code source](#)

### 4.6.5 NumPy

NumPy est une bibliothèque pour langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux.



### 4.6.6 Simple PID

Simple PID est une librairie destinée à aider afin de faire facilement et rapidement des PID.

#### Utilisation

Pour l'utilisation voici un lien vers mon [code source](#)

## 4.6.7 Servokit adafruit

Servokit d'adafruit est une librairie qui permet de contrôler les servomoteurs du robot grâce au module PCA9685.

### Utilisation

Pour l'utilisation voici un lien vers mon [code source](#)

## 4.6.8 adafruit\_rplidar

Adafruit\_rplidar est une librairie faite par adafruit qui permet de communiquer avec le lidar, et est efficace pour l'acquisition des données, puis le contrôle du lidar.



### Utilisation

Pour l'usage voici un lien vers mon [code source](#)

## 4.6.9 Git

Git est un logiciel de gestion de version qui a été utilisé durant la réalisation du projet afin d'avoir un historique de celui-ci. Le code source est disponible [ici](#).



## 4.6.10 Trello

Trello est un outil de gestion de projet en ligne. Inspiré par la méthode Kanban de Toyota. Il repose sur une organisation des projets en planches, listant des cartes, chacune représentant des tâches.

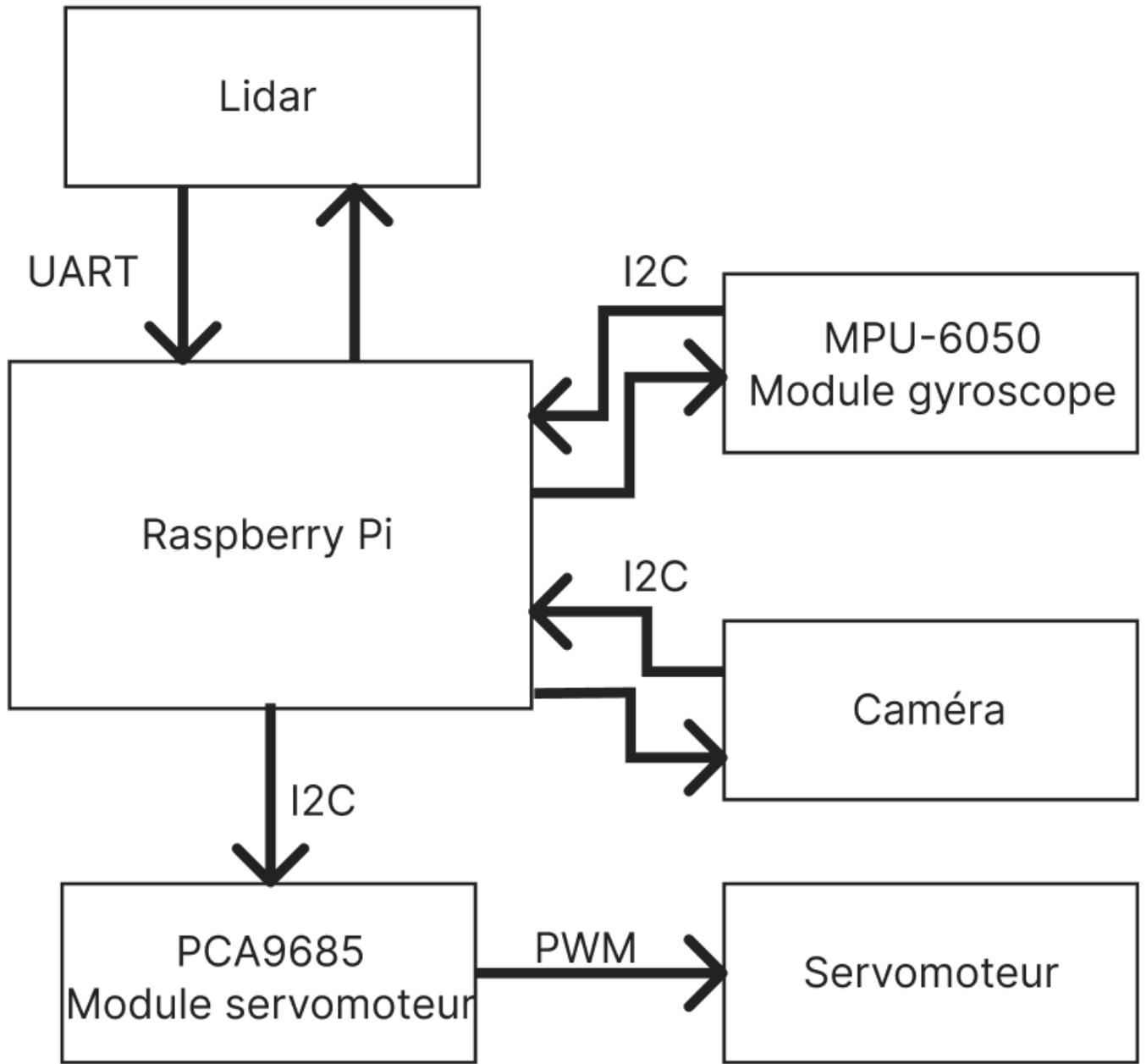
Vous trouvez [ici](#) le trello du projet.



| À faire                                  | Commencée                          | Finis   | Notes               |
|--|------------------------------------|---|---------------------|
| Module pour batterie                     | Remplir le jdb                     | Rendre le raspberry pi en accès point WiFi<br>🕒 00:49                               | HARDWARE            |
| Repenser la solidité de certaines pièces | Remplir la documentation technique | Insertion de l'information de l'obstacle le plus proche, sur la bannière<br>🕒 00:14 | SOFTWARE            |
| Détection d'une personne avec opencv     | Création des mouvements<br>✉ 0/6   | Creation du journal de bord   | DOCS                |
| Robot suit une personne                  | + Ajouter une carte                | Test des servos   | + Ajouter une carte |
| Mode autonome                            |                                    | Finir les impressions   |                     |
| + Ajouter une carte                      |                                    | Trouver la force minimal pour que les servos tiennent le robot quasi "Stable"       |                     |
|  |                                    | Voir comment fonctionnent les threads et le multi-threading                         |                     |
|  |                                    | montage du robot<br>🕒 03:00   |                     |
|  |                                    | Optimisation de l'interface Web<br>🕒 33:40 Running timer                            |                     |
|  |                                    | Faire la classe lidar   |                     |
|  |                                    | Création du poster  |                     |
|  |                                    | + Ajouter une carte   |                     |

## 4.7 Analyse des fonctionnalités majeures

Voici un schéma qui représente la communication entre les composants hardware et leurs protocoles de communication.



### 4.7.1 Classe Lidar

Pour le projet j'ai placé le [lidar](#) sur le haut du robot, car si je le mettais en dessous j'allais toujours détecter les pattes ce qui nous enlèverait beaucoup trop de vision. Le lidar tourne en continu, car, ainsi les informations prennent moins de temps à être prises.

Ensuite, pour la programmation de cette classe, à l'initialisation du script «app.py» je lance le moteur du lidar, puis, de manière asynchrone, j'envoie les informations reçues du lidar dans une liste, ce qui les actualise régulièrement. Pour l'acquisition des données, il faut prendre toujours 2 tours du lidar, car au premier tour le lidar nous envoie depuis sa position actuelle jusqu'à ~360°, donc si on veut avoir un tour complet sûr, nous avons qu'à enregistrer le tour complet d'après, c'est pour ça que je fais 2 tours. Puis dès qu'une classe en a besoin, elle appelle la fonction «Get\_Data()» qui lui envoie une liste des dernières valeurs recueillies.

| <b>Lidarasync</b>     |
|-----------------------|
| + ShorterScan : Float |
| + __init__()          |
| + __new__(object)     |
| +Get_Data() : List    |
| +DoScan()             |
| +StartLidar()         |
| +StopLidar(Bool)      |

Comme vous pouvez le voir ci-dessus, cette classe est faite d'une propriété (ShorterScan), et de 4 fonctions. Je vais détailler ci-dessous ce que font ces éléments.

#### **ShorterScan : Float**

ShorterScan est une propriété qui sert à ce qu'il y ait toujours la distance la plus proche entre le robot et un obstacle disponible. Elle se met à jour dans la fonction «DoScan».

#### **\_\_init\_\_()**

Ceci est le constructeur de ma classe Lidar. Il initialise les valeurs, puis instancie un lidar.

#### **\_\_new\_\_(**

Cette fonction est utile, pour rendre les classes en singleton, afin de ne pas avoir d'autres instances du lidar dans le programme.

#### **Get\_Data() : List**

Dans cette fonction, je vérifie que la liste avec les données ne soit pas vide et si l'il n'est pas vide je renvoie les valeurs, sinon un tableau vide.

#### **DoScan()**

Au début de la fonction, j'initialise les valeurs et reconnecte le lidar. Ensuite, je lance un scan grâce à la fonction du module RPLidar «iter\_scans», et j'entre les valeurs dans une liste vide, qui se remplit petit à petit. Puis dans ces valeurs là, je les trie et j'actualise la plus petite distance. Et finalement, quand il y a deux scans, j'arrête l'acquisition des données.

#### **StartLidar()**

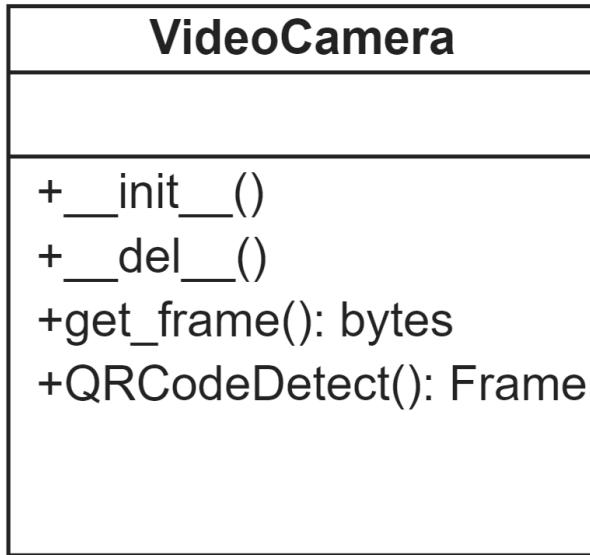
Cette fonction me sert à lancer le moteur au début du programme principal.

#### **StopLidar(Bool)**

Cette fonction me sert à arrêter le lidar, dès que les données sont récoltées, et avec le paramètre je peux aussi arrêter le moteur, afin de faire un arrêt complet du lidar.

#### 4.7.2 Classe VideoCamera

Pour cette classe, j'utilise la librairie [OpenCV](#). Cette classe me permet de récupérer les données de la [caméra](#).



##### `__init__()`

Ceci est le constructeur de la classe VideoCamera, il instancie la caméra grâce à openCV.

##### `__del__()`

Cette fonction sert à libérer la caméra.

##### `get_frame() : bytes`

Cette fonction capture le flux vidéo de caméra, la convertie en bytes et retourne celui-ci.

##### `QRCodeDetect() : Frame`

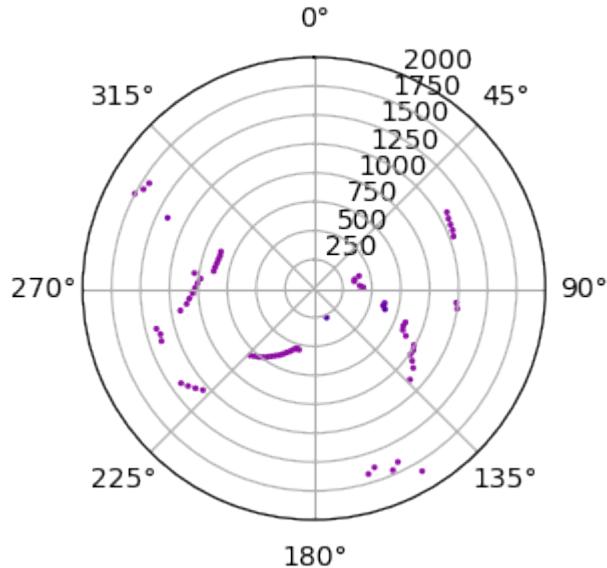
Cette fonction sert à analyser le retour de la caméra et y chercher un QRCode, puis retourne une image avec le QRCode encadré en bleu.

#### 4.7.3 Classe Radar

Cette classe sert à créer la vue radar. Elle m'est utile pour créer une image avec les points que la classe lidar détecte et je l'actualise afin d'avoir un retour un minimum fluide. À l'initialisation, la classe dessine l'arrière-plan de l'image, les lignes pour les distances ainsi que mettre les chiffres pour que l'utilisateur sache de qu'elle distance il s'agit.

Ensuite, j'actualise seulement les points dans le but d'essayer de ne pas prendre trop de temps inutilement en redessinant des éléments futiles, et/ou permanents.

Voici un exemple de la vue que l'utilisateur voit :



## Radar

```
+ __init__()
+CreatePlot(): Bytes
```

### \_\_init()

Cette fonction est le constructeur de cette classe. Elle me permet d'instancier matplotlib, puis initialiser les valeurs qui ne changent pas.

### CreatePlot(): Bytes

Cette classe me permet de remettre à jour la vue radar. Puis, j'envois la nouvelle figure à la classe principale qui l'affiche sur le site web.

## 4.7.4 Classe Gyroscope

Afin d'utiliser correctement ce module, j'ai utilisé la librairie faite par le fabricant. Ce qui gère pour nous la communication entre le gyroscope et le raspberry pi. Ce qui rend les fonctions plus courtes et simples.

## Gyroscope

- + \_\_init\_\_()
- +get\_angle()
- +get\_acceleration()

### \_\_init\_\_()

Cette fonction est le constructeur de la classe. Elle me permet d'instancier le capteur, ainsi qu'initialiser certaines valeurs.

### get\_angle(): dict[str, float]

Cette classe permet récupérer l'inclinaison du robot grâce au gyroscope.

### get\_acceleration(): dict[str, float]

Cette classe permet de récupérer les valeurs de l'accéléromètre.

## 4.7.5 Classe ServoMoteur

Afin d'avoir une meilleure compréhension dans mes mouvements, j'ai fait une classe qui contrôle les servomoteurs.

Voici le diagramme de classe :

## ServoMoteur

- + \_\_init\_\_(str, int)
- +SetAngleRel(int,int)
- +StayWithForce()
- +WithoutForce()

### \_\_init\_\_(string, int)

Ceci est le constructeur de la classe qui initialise le servomoteur, sur le bon module PCA et à la bonne Pin.

Channel = "Gauche"/"Droite"

Position = 0 à 15

**SetAngleRel(int, int)**

Cette fonction permet de définir l'angle du servomoteur avec une force donnée. La force est entrée en pour cent.

Angle = -180 à 180

Force = 1 à 100

**StayWithForce(string)**

Cette fonction permet de garder le servomoteur avec une légère force, grâce au paramètre nous pouvons décider dans quel sens.

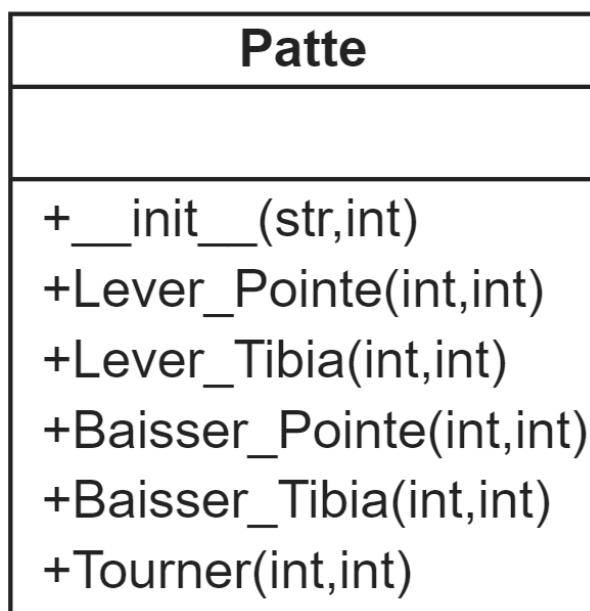
**WithoutForce()**

Cette fonction met le servomoteur sans force.

## 4.7.6 Classe Patte

Cette classe me sert à faire les mouvements des parties des pattes plus simplement. Les fonctions font des actions précises, et que le nécessaire afin de créer des mouvements.

Voici le diagramme de la classe :

**\_\_init\_\_(string, string)**

Ceci est le constructeur de cette classe. J'initialise les servomoteurs, de la patte selon le branchement hardware.

Le branchement a été fait en sorte que chaque groupe de 3 servomoteurs par patte soit facilement retrouvable.

Pour l'avant gauche:

- La pointe: Pin 2 - Le tibia: Pin 1 - La Hanche: Pin 0

Pour le milieu gauche:

- La pointe: Pin 4 - Le tibia: Pin 5 - La Hanche: Pin 6

Pour l'arrière gauche:

- La pointe: Pin 15 - Le tibia: Pin 13 - La Hanche: Pin 12

Pour l'avant droit:

- La pointe: Pin 15 - Le tibia: Pin 14 - La Hanche: Pin 13

Pour le milieu droit:

- La pointe: Pin 4 - Le tibia: Pin 5 - La Hanche: Pin 6

Pour l'arrière droit:

- La pointe: Pin 0 - Le tibia: Pin 1 - La Hanche: Pin 2

#### **Lever\_Pointe (int, int)**

Cette fonction permet de lever la pointe de la patte à un angle précis et à la force donnée.

#### **Lever\_Tibia (int, int)**

Cette fonction permet de lever le tibia de la patte à un angle précis et à la force donnée.

#### **Baisser\_Pointe (int, int)**

Cette fonction permet de baisser la pointe de la patte à un angle précis et à la force donnée.

#### **Baisser\_Tibia (int, int)**

Cette fonction permet de baisser le tibia de la patte à un angle précis et à la force donnée.

#### **Tourner (int, int)**

Cette fonction permet de tourner la patte à un angle précis et à la force donnée.

### 4.7.7 Classe Animations

---

Cette classe permet d'avoir toutes les animations du robot. Chaque animation est détaillée étape par étape afin que la compréhension soit plus simple.

## Animations

- + \_\_init\_\_(Gyroscope)
- + Init()
- + Avance()
- + Recule()
- + Gauche()
- + Droite()
- + Rotation\_Horaire()
- + Rotation\_AntiHoraire()
- + Maintiens()

### \_\_init\_\_(Gyroscope)

Cette fonction est le constructeur de cette classe. Elle permet d'initialiser tous les servomoteurs, la communication I2C avec les modules PCA9685 et le gyroscope.

### Init()

Cette fonction est l'animation au démarrage du robot. Elle permet de lever le robot.

### Avance()

Cette fonction est l'animation qui permet de faire avancer le robot.

### Recule()

Cette fonction est l'animation qui permet de faire reculer le robot.

### Gauche()

Cette fonction est l'animation qui permet de déplacer le robot vers la gauche.

### Droite()

Cette fonction est l'animation qui permet de déplacer le robot vers la droite.

### Rotation\_Horaire()

Cette fonction est l'animation qui permet de faire une rotation de sens horaire au robot.

### Rotation\_AntiHoraire()

Cette fonction est l'animation qui permet de faire une rotation de sens antihoraire au robot.

**Maintiens()**

Cette fonction permet de laisser le robot debout sans bouger, et le remettre droit, grâce aux PID pour les axes X et Y du gyroscope.

## 4.8 Tests

---

### 4.8.1 Raspap

| ID  | Description  | Statut | Commentaire                                  |
|-----|--|--------|--|
| 001 | Le smartphone détecte le réseau                          | OK     | Le smartphone détecte correctement le réseau |
| 002 | Le smartphone peut se connecter au réseau                | OK     | -  |
| 003 | Le smartphone peut accéder au site local du raspberry pi | OK     | -  |
| 004 | Le PC détecte le réseau                                  | OK     | Le PC détecte correctement le réseau         |
| 005 | Le PC peut se connecter au réseau                        | OK     | -  |
| 006 | Le PC peut accéder au site local du raspberry pi         | OK     | -  |

### 4.8.2 Flask

| ID  | Description                         | Statut | Commentaire   |
|-----|-------------------------------------|--------|---|
| 011 | Serveur Flask accessible            | OK     | -   |
| 012 | Permet l'accès à la bonne page      | OK     | -   |
| 013 | Rafraîchis correctement les données | OK     | Le rafraîchissement se fait correctement et toutes les secondes |

### 4.8.3 Mobile

| ID  | Description  | Statut | Commentaire                                   |
|-----|--|--------|---|
| 021 | La taille est adaptée au smartphone  | OK     | -   |
| 022 | Les boutons apparaissent aux bons endroits                                     | OK     | -   |
| 023 | Les boutons contiennent les bonnes informations                                | OK     | -   |
| 024 | La caméra est active?  | OK     | La caméra affiche correctement sur le support |
| 025 | La caméra est fluide?  | OK     | La caméra est suffisamment fluide             |
| 026 | Tous les boutons de directions renvoient la méthode post, avec la bonne valeur | OK     | -   |
| 027 | Le sélecteur de modes renvoie la méthode, avec la valeur du mode choisi        | OK     | -   |
| 028 | Si l'utilisateur clique sur la vue radar, elle s'agrandit                      | OK     | -   |
| 029 | Le bouton pour choisir la transparence est fonctionnel                         | OK     | -   |

#### 4.8.4 PC

| ID  | Description  | Statut | Commentaire                                   |
|-----|--|--------|---|
| 031 | La taille est adaptée au PC  | OK     | -   |
| 032 | Les boutons apparaissent aux bons endroits                                   | OK     | -   |
| 033 | Les boutons contiennent les bonnes informations                              | OK     | -   |
| 034 | La caméra est active?  | OK     | La caméra affiche correctement sur le support |
| 035 | La caméra est fluide?  | OK     | La caméra est suffisamment fluide             |
| 036 | Tous les boutons de directions renvois la methode post, avec la bonne valeur | OK     | -   |
| 037 | Le sélecteur de modes renvois la méthode, avec la valeur du mode choisis     | OK     | -   |
| 038 | Si l'utilisateur clique sur la vue radar, elle s'agrandit                    | OK     | -   |
| 039 | Le bouton pour choisir la transparence est fonctionnel                       | OK     | -   |

#### 4.8.5 Modules

| ID  | Description                               | Statut | Commentaire                                    |
|-----|---|--------|--|
| 041 | Le module Gyroscope agit correctement     | OK     | s'allume et intéragit avec le raspberry pi     |
| 042 | Les modules PCA9685 agissent correctement | OK     | s'allume et intéragit avec le raspberry pi     |
| 043 | La caméra agit correctement               | OK     | s'allume et intéragit avec le raspberry pi     |
| 044 | Les servomoteurs sont fonctionnels        | OK     | -  |
| 045 | Le lidar agit correctement                | OK     | l'acquisition des données est faite rapidement |

## 4.8.6 Classes

### Animations

| ID  | Description                                   | Statut | Commentaire   |
|-----|---|--------|---|
| 051 | Initialise correctement tout les servomoteurs | OK     | -   |
| 052 | Init2 est finis et fonctionnel                | OK     | -   |
| 053 | Avance est finis et fonctionnel               | OK     | -   |
| 054 | Maintiens est finis et fonctionnel            | OK     | -   |
| 055 | Recule est finis et fonctionnel               | KO     | Problème avec les servomoteurs expliqué <a href="#">ici</a> |
| 056 | Droite est finis et fonctionnel               | KO     | Problème avec les servomoteurs expliqué <a href="#">ici</a> |
| 057 | Gauche est finis et fonctionnel               | KO     | Problème avec les servomoteurs expliqué <a href="#">ici</a> |
| 058 | Rotation_Horaire est finis et fonctionnel     | KO     | Problème avec les servomoteurs expliqué <a href="#">ici</a> |
| 059 | Rotation_AntiHoraire est finis et fonctionnel | KO     | Problème avec les servomoteurs expliqué <a href="#">ici</a> |
| 060 | Off est finis et fonctionnel                  | OK     | -   |

### VideoCamera

| ID  | Description  | Statut | Commentaire   |
|-----|--|--------|---|
| 151 | Initialise correctement la caméra                      | OK     | -   |
| 152 | get_frames dessine correctement les carrés sur l'image | OK     | -   |
| 153 | les carrés sont réactif selon les obstacles            | OK     | -   |
| 154 | get_frames renvoie correctement l'image                | OK     | -   |
| 155 | QRCodeDetect est finis et fonctionnel                  | OK     | Déetecte le QRCode le décrite et l'encadre sur la vidéo |

### Gyroscope

| ID  | Description  | Statut | Commentaire |
|-----|--|--------|-------------|
| 251 | Le gyroscope s'initialise correctement                               | OK     | -           |
| 252 | get_angle renvoie correctement les données sur trois axes            | OK     | -           |
| 253 | get_acceleration renvoie correctement les données sur les trois axes | OK     | -           |

**Lidarasync**

| ID  | Description   | Statut | Commentaire |
|-----|---|--------|-------------|
| 351 | Le lidar s'initialise correctement  | OK     | -           |
| 352 | Le singleton renvoie l'instance créée du lidar                                    | OK     | -           |
| 353 | La propriété de la distance la plus courte est atteignable par les autres scripts | OK     | -           |
| 354 | Get_Data renvoie correctement la liste des données récupérées par le lidar        | OK     | -           |
| 355 | DoScan effectue correctement les scans et les exceptions sont gérées              | OK     | -           |
| 356 | StartLidar démarre correctement le lidar  | OK     | -           |
| 357 | StopLidar arrête correctement le lidar  | OK     | -           |

**Motor**

| ID  | Description  | Statut | Commentaire |
|-----|--|--------|-------------|
| 451 | La classe initialise correctement les servomoteurs                               | OK     | -           |
| 452 | SetAngleRel bouge correctement le servomoteur à un angle relatif                 | OK     | -           |
| 453 | StayWithForce garde une force minimale au servomoteur et dans la bonne direction | OK     | -           |
| 454 | WithoutForce enlève correctement la force du servomoteur                         | OK     | -           |

**Plot**

| ID  | Description   | Statut | Commentaire |
|-----|---|--------|-------------|
| 551 | La classe s'initialise correctement                                 | OK     | -           |
| 552 | CreatePlot récupère et interprète correctement les données du lidar | OK     | -           |
| 553 | Les couleurs sont en corrélation avec la ligne de carrés            | OK     | -           |
| 554 | CreatePlot renvoie correctement la vue radar                        | OK     | -           |

**RepeatTimer**

| ID  | Description   | Statut | Commentaire |
|-----|---|--------|-------------|
| 651 | La classe crée correctement un thread qui est répété en continu selon une intervalle donnée | OK     | -           |

## 4.9 Difficultés rencontrées

---

Durant la conception de ce projet, en n'ayant pas assez préparé et compris le fonctionnement des modules hardware. J'ai rencontré de nombreux problèmes que j'ai su résoudre, mais qui m'ont fait perdre du temps.

### 4.9.1 Vue Radar

Pour la vue radar, visible sur l'interface en haut à droite, j'ai eu très longtemps un problème avec l'insertion des données pendant la création du plot. Ce qui faisait en sorte que la réalisation des nouveaux plots prenait beaucoup trop de temps. Et de ce fait, la vue était plus souvent vide qu'avec une image avec les points qui représentent les obstacles. Pour régler ce problème, j'ai examiné ce qui ne changeait jamais sur l'image (par exemple le fond, les cercles, ainsi que la distance). Puis, j'ai optimisé le code en évitant de refaire ce qui ne change pas, et de juste mettre les nouveaux points dessus.

Ensuite, j'ai remarqué que le temps de téléchargement des images prenait aussi beaucoup trop de temps. J'ai cherché longuement sur internet et sur la documentation officielle de flask, le pourquoi. Malheureusement, je n'ai pas réussi à trouver une réponse concrète. J'ai essayé plusieurs techniques comme ne pas enregistrer en cache en y mettant un header, mais ça n'a pas résolu ce problème. J'ai alors pensé à faire comme avec la caméra et faire un stream de l'image (image qui est donc enregistrée et ensuite streamée).

### 4.9.2 Lidar

Le lidar m'a donné beaucoup de problème, ou plutôt d'incompréhension. Tout d'abord, en programmation en général, je n'ai jamais travaillé avec les exceptions et encore moins en python. En travaillant avec le lidar, j'avais beaucoup d'exceptions à prendre en compte. Car si je ne les gérerais pas, le programme s'arrêtait et ce n'était pas concevable que le programme s'arrête à cause d'une exception. Ce que j'ai fait pour régler ce problème fut un try except, qui dans son nom auto-explicatif, sert à gérer les exceptions, et au lieu d'arrêter le programme, je faisais un arrêt du lidar, et le relançais.

Ensuite, en corrélation avec les problèmes de la vue radar, j'ai remarqué que l'acquisition de données était un peu longue. Dans la documentation j'avais lu qu'en un tour, il enregistrait 400 données, mais dû à une mauvaise compréhension de ma part, je faisais donc une boucle for, qui enregistrait 400 tours, et pas 1 tour (qui a l'intérieur avait 400 valeurs). Et donc, pour régler ce problème, je n'ai que changé la valeur de la boucle for à 2 tours, chiffre expliqué dans [l'analyse fonctionnelle](#).

Puis, pour le projet, je devais faire du multithreading afin que le programme tourne tout le temps sans qu'il soit bloqué dans l'acquisition des données du lidar constamment. La compréhension du fonctionnement du multithreading m'a pris un certain temps. Mais ensuite, ce fut assez rapide et en pratiquant, j'ai remarqué que le timer de base de python, n'est pas une fonction qui lance un thread chaque x temps, mais qui lance une fois la fonction en thread, mais après x temps. J'ai donc du faire ma propre classe qui reprend la fonction de base, mais qui répète automatiquement la fonction en thread.

### 4.9.3 Mouvements

Pour les mouvements du robot, ce qui a été le plus compliqué, a été de synchroniser tous les servomoteurs, car ceux que j'utilise sont des servomoteurs à rotation continue expliquée dans la partie [Composants](#) et pour résumer je n'ai pas de feedback donc ils sont à l'aveugle. Pour régler ce problème, je peux détecter les pattes grâce au lidar, et ainsi je peux un savoir l'angle où ils sont afin de faire l'initialisation, et ainsi avoir un référentiel.

Ensuite, le plus gros problème du projet, à la suite d'une mauvaise estimation du poids total du robot, donc de la force nécessaire par rapport à la mécanique, les servomoteurs n'ont pas assez de force pour soulever le robot de manière à l'aise. Et les seuls moyens pour régler ce souci sont:

- Changer les servomoteurs

Je pourrais changer les servomoteurs et en prendre des plus puissants, sauf que cette solution est très chère, car il y a 18 servomoteurs à changer et les prix grimpent vite pour avoir de la puissance.

- Changer la mécanique

Je pourrais changer la mécanique du robot, et ainsi les moments de force seront moins importants, et aussi réduire le poids des pièces en enlevant les espaces inutiles.

#### 4.9.4 Raspap

---

Durant le développement du projet, il y a eu un problème suite à l'installation de raspap. Si nous laissons les valeurs par défaut, et qu'un de mes camarades faisait l'installation de raspap en laissant tout par défaut aussi, celui qui a émis en dernier est le dominant. Ce qui veut dire que tout les autres raspberry pi, ne pouvaient plus émettre, ce qui m'empêchait de communiquer avec mon raspberry pi.

## 4.10 Conclusion

---

### 4.10.1 À finir dans le projet

Malheureusement dans ce projet, je n'ai pas pu finir tout ce qui était dans mon [cahier des charges](#). Voici la liste de ce que je n'ai pas pu achever et le problème qui me bloque:

#### Idées non réalisées

- Les mouvements du robot:

Je n'ai pas pu finir les mouvements du robot, parce que ça prend beaucoup de temps à les faire, puis j'avais un problème sur toute la moitié du projet qui faisait freeze le raspberry pi. Il se bloquait, car j'utilisais la connexion SSH de Visual Studio Code, qui consomme beaucoup trop et qui faisait surchauffer le raspberry pi. Pour résoudre ce souci, j'ai arrêté d'utiliser Visual Studio Code et je me suis servi de puTTY qui son utilisation est quasi nulle.

- L'algorithme Suiveur:

Avec le temps qui me restait dans les derniers jours, avec la documentation à faire, je n'ai pas pu implémenter cette fonctionnalité. J'ai pu faire la détection de QRCode, mais il me manquait le fait de décider s'il faut se tourner à gauche ou à droite dépendamment d'où est détecté le QRCode sur l'image. Puis, avancer ou reculer grâce à la détection par le lidar.

- Mettre le robot sur batterie:

Je n'ai pas mis le robot sur batterie, car il consomme beaucoup trop quand les servomoteurs sont actifs. Il consomme entre 3 [A] et 5 [A], une batterie normale se déchargerait extrêmement vite. Et il y a le problème du serpent qui se mord la queue. Si je mets une batterie assez grande pour fournir les 5 [A], il me faut des servomoteurs plus puissants, car ça sera une batterie lourde. Et si je mets des servomoteurs plus puissants, ils consommeront plus donc il faut une plus grosse batterie.

- Monter/Descendre des escaliers:

Cette idée n'a pas été réalisée, car les mouvements ne sont pas finis, et il aurait fallu avoir le feedback des servomoteurs puis des boutons sur chaque patte pour savoir la hauteur de la marche d'escalier.

#### Fonctionnalités avec des bugs

La ligne de carré:

Dans le retour caméra, il y a une ligne de carré que sert à voir par rapport à la vue de la caméra et la distance que détecte le lidar, la distance entre le robot et les obstacles en face de lui. Cette fonctionnalité marche dans son ensemble, sauf qu'il a un bug où les valeurs ne se remettent pas à 0 directement.

Changement de mode:

Au changement de mode, je ne sais pas pourquoi, il y a toujours un bug avec la caméra qui surgit. Le bug est que pour créer les carrés j'ai besoin d'une image, et qu'au changement je n'ai plus de retour vidéo, et ça me renvoie une alerte. Mais si on recharge la page web, tout revient à la normale et nous sommes dans le bon mode.

### 4.10.2 Améliorations possibles

---

#### Servomoteurs plus puissants

L'un des gros problèmes de ce projet fut que j'ai mal dimensionné les servomoteurs, ils ne sont pas assez forts pour que le robot fasse des mouvements à l'aise.

#### Optimisation

Je pense que le code n'est pas très propre par moment, par conséquent que l'on pourrait optimiser certaine partie du code qu'il s'agisse de performance ou tout simplement de suppression de redondance dans certains cas. De plus, je pense que l'utilisation

du lidar branché sur les pins du GPIO du Raspberry Pi 3 serait moins coûteuse énergétiquement et la transmission des données plus rapide.

#### Ajout de modes

Grâce à tous les capteurs intégrés au robot, on pourrait ajouter plein de modes différents tels que:

- L'inverse du suiveur donc le fuyard

De la façon dont j'imagine ce mode est que dès qu'il détecte le QR Code, le robot hexapode fuirait dans n'importe quel sens sauf celui où il a détecté le QR Code.

- Création de chemin

Ce mode serait le fait que grâce à la caméra, l'utilisateur place des points et que le robot suive ce parcours.

Et un moyen pour que n'importe quel utilisateur puisse créer leurs propres modes.

#### 4.10.3 Bilan personnel

---

Durant ce travail de diplôme, il s'est passé un nombre incalculable d'évènements, des bons comme des mauvais. Ces deux derniers mois ont mis mes connaissances informatiques à rude épreuve. Malgré les divers problèmes que j'ai pu rencontrer durant l'élaboration du ce projet, je suis très fier de ce que j'ai réussi à produire avec les connaissances actuelles. Surtout en n'étant pas un professionnel dans le domaine de l'informatique. Durant ces deux mois, j'ai pu apprendre beaucoup de choses.

La première est qu'il est nécessaire d'avoir une approche méthodique et organisée face au travail qui nous est demandé de réaliser. Pour ma part, je devais réaliser une application capable de contrôler un robot hexapode à distance, tout en devant interagir avec les composants suivant:

- Caméra
- RPLidar
- Gyroscope
- Servomoteurs

Et de récupérer les données de certains de ces composants. Les éléments principaux que je n'ai pas pu implémenter, ça a été les mouvements en général du robot. J'arrive à le faire se lever, et il avance, mais je n'ai pas fait toutes les autres animations, car ça demande beaucoup de temps. Cependant, on peut avoir accès au radar 360° en temps réel. De plus, avec mon projet on peut contrôler un hexapode à distance et interagir en temps réel avec les capteurs attachés à lui à l'aide de son téléphone portable.

La seconde est qu'avoir une approche méthodique et rigoureuse est très important, car c'est en décomposant les diverses thématiques que l'on peut plus aisément prévoir les points bloquants afin de les résoudre. En plus de toutes les connaissances que j'avais, décomposer les problèmes auxquels j'ai fait face, m'aurais permis de les résoudre plus aisément.

La troisième est la vitesse de réflexion qu'il faut avoir. Pour un projet aussi conséquent que celui-là, il n'aurait pas fallu réfléchir des journées entières pour savoir où était le problème, savoir comment faire une certaine fonctionnalité, etc. J'aurais dû savoir bien à l'avance tout ce que j'avais à faire et avoir déjà une idée de comment le faire.

C'est pourquoi je suis fier de ce que j'ai réalisé durant ce travail de diplôme et que s'il m'était demandé de refaire un projet similaire, je sauterais sur l'occasion sans hésiter. Pouvoir créer quelque chose dans la robotique est très satisfaisant.

## 5. Cahier des charges

---

### 5.1 Sujet

Création d'un robot capable de se déplacer de manière autonome, ainsi que d'être contrôlé par quelqu'un.

### 5.2 But du projet

Ce projet a pour but de fabriquer un robot hexapode, qui aura comme fonctionnalités:

- Pouvoir se déplacer sur le sol (pente ou pas).
- Monter/Descendre des escaliers.
- Déetecter toute sorte d'obstacles autour de lui.
- Pouvoir être autonome.
- Pouvoir être contrôlé par un téléphone grâce à une application.
- (optionnel) Pouvoir être contrôlé par une télécommande.

### 5.3 Spécification

Ce robot sera fait de A à Z par mes soins, modéliser les pièces en 3D, les imprimer grâce à une imprimante 3D, monter le tout, et aussi programmer le robot ainsi que l'application pour le smartphone.

Ce robot comporte trois modes principaux: un mode autonome ainsi qu'un mode contrôlé par un utilisateur.

Dans le mode autonome, le robot pourra marcher, éviter les obstacles grâce au lidar, puis monter des marches tout seul, et si le robot par malchance se retourne, il y aura un gyroscope pour qu'il puisse détecter le fait qu'il soit à l'envers, et ainsi, il puisse se retourner.

Ensuite, il y a le mode utilisateur, qui lui, permettra à une personne de contrôler le robot grâce à un smartphone, et qui pourra faire plusieurs fonctionnalités, par exemple: faire un saut, désactiver les capteurs, contrôler la vitesse de marche, activer la caméra, etc. Et finalement, le mode suiveur, où le robot détectera une personne devant lui grâce à la caméra et ainsi la suivra partout où elle ira.

L'avantage de faire un robot hexapode, est que nous pouvons décider de la hauteur du robot, par exemple, nous pouvons décider qu'il soit à ras du sol, comme nous pourrons décider qu'il soit sur la pointe des pattes. Ensuite, autre avantage, est que vu qu'il a 6 pattes, il pourra être stable sur 4 pattes et ainsi avoir 2 pattes, pour faire certaines actions, comme pousser une porte, déplacer un objet, ou voire même pouvoir le tenir entre 2 pattes. Et enfin, le dernier avantage, c'est que vu que c'est des pattes et pas des roues, le robot pourra descendre et monter des escaliers ainsi que grimper certains obstacles.

Ensuite, afin d'avoir une certaine stabilité électriquement parlant, j'ai décidé de mettre 2 batteries sur le robot. Une sera pour alimenter le raspberry pi ainsi que tous les modules, et l'autre sera là pour alimenter que les servos, car c'est ce qui va tirer le plus de courant, ce système est fait pour qu'il n'y ai aucun problème avec un manque de courant pour le raspberry pi.

#### 5.3.1 Spécification des différents modules

- Lidar: composant qui fait une cartographie de l'environnement afin de détecter les obstacles autour du robot.
- Sonar: composant qui envoie un ultrason au sol avec un certain angle afin de détecter s'il y a un vide en face du robot.
- PCA9685: composant qui sert à gérer plusieurs servomoteurs en même temps.
- MPU6050: gyroscope et accéléromètre qui servira à détecter si le robot, c'est retourné, ainsi que son accélération.
- Camera: caméra qui sera utile pour que l'utilisateur puisse voir sur son smartphone une vue par rapport au robot.

#### 5.3.2 To do list

- Déterminer la consommation générale du robot afin de calculer une batterie adéquate pour minimum 2 h d'autonomie.
- En fonction des servomoteurs, déterminer la vitesse maximale à laquelle le robot pourra se déplacer.
- En fonction des servomoteurs, déterminer la charge totale que pourrait recevoir le robot.

- Réalisation des pièces en 3D, car s'il y a des problèmes avec l'imprimante 3D, toutes les pièces prendront beaucoup trop de temps à être fabriquées.
- Réalisation d'une carte pouvant charger les deux batteries du robot.

## 5.4 Restrictions

---

- Début le 4 Avril au 10 Juin
- Les pièces 3D doivent déjà être utilisables
- utilisation majeure du python (cause raspberry pi)
- Utilisation d'un oscilloscope
- (Utilisation du Lidar)

## 5.5 Parties prenantes

---

| Nom              | Fonction   |
|------------------|------------|
| Pascal Bonvin    | Professeur |
| Nicolas Oliveira | Élève      |

## 5.6 Analyse de l'existant

---

- lynxmotion [Phoenix](#)

Cet hexapode de chez lynxmotion est un robot intéressant et imposant. Malheureusement, il n'a aucun capteur, il lui est donc impossible d'être autonome, il lui faut donc être contrôlé par l'utilisateur.

- Adeept [RaspClaws](#)

Cet hexapode de chez Adeept est un robot ressemblant à celui de chez lynxmotion. La valeur ajoutée est qu'il y a une caméra d'installée dessus. Mais aucun autre capteur n'est installé dessus. Donc, comme celui au dessus il lui est impossible d'être autonome.

- EZ-Robot [Hexapod](#)

Cet hexapode de chez EZ-Robot, est le plus intéressant que j'ai pu trouver. Il a une interface pour mobile qui permet d'être contrôlé à distance, avec intégré le retour caméra. Mais comme pour les autres, c'est le seul capteur implémenté dessus.

Comme on peut le voir, est que mon robot à des capteurs qui lui permettent d'être autonome. Par exemple, le lidar qui permet à mon hexapode de détecter des obstacles même derrière lui. Ensuite, le gyroscope afin de pouvoir le stabiliser et détecter l'inclinaison du robot. Puis, comme les robots au-dessus, une caméra qui me sert de retour afin d'être facilement contrôlable à distance.

## 5.7 Environnement

---

### 5.7.1 Langage de programmation

- Python
- peut-être web (python Flask)

### 5.7.2 Système d'exploitation

- Raspbian

### 5.7.3 Matériel

---

- Ordinateur personnel
- Imprimante 3D
- Raspberry Pi

### 5.7.4 Software

---

- SolidWorks (CAO)
- Visual Studio Code
- KiCad (carte recharge des batteries)

## 5.8 Livrables

---

- Documentation
- Journal de bord
- Code source
- Robot

## 6. Annexes

### 6.1 Manuel utilisateur

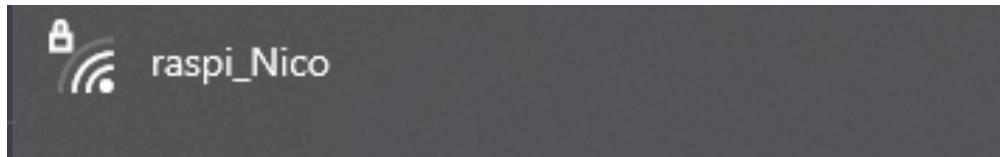
#### 6.1.1 Connection au raspberry pi

la première étape pour pouvoir utiliser est de vous connecter en ssh au raspberry pi.

Veuillez l'allumer

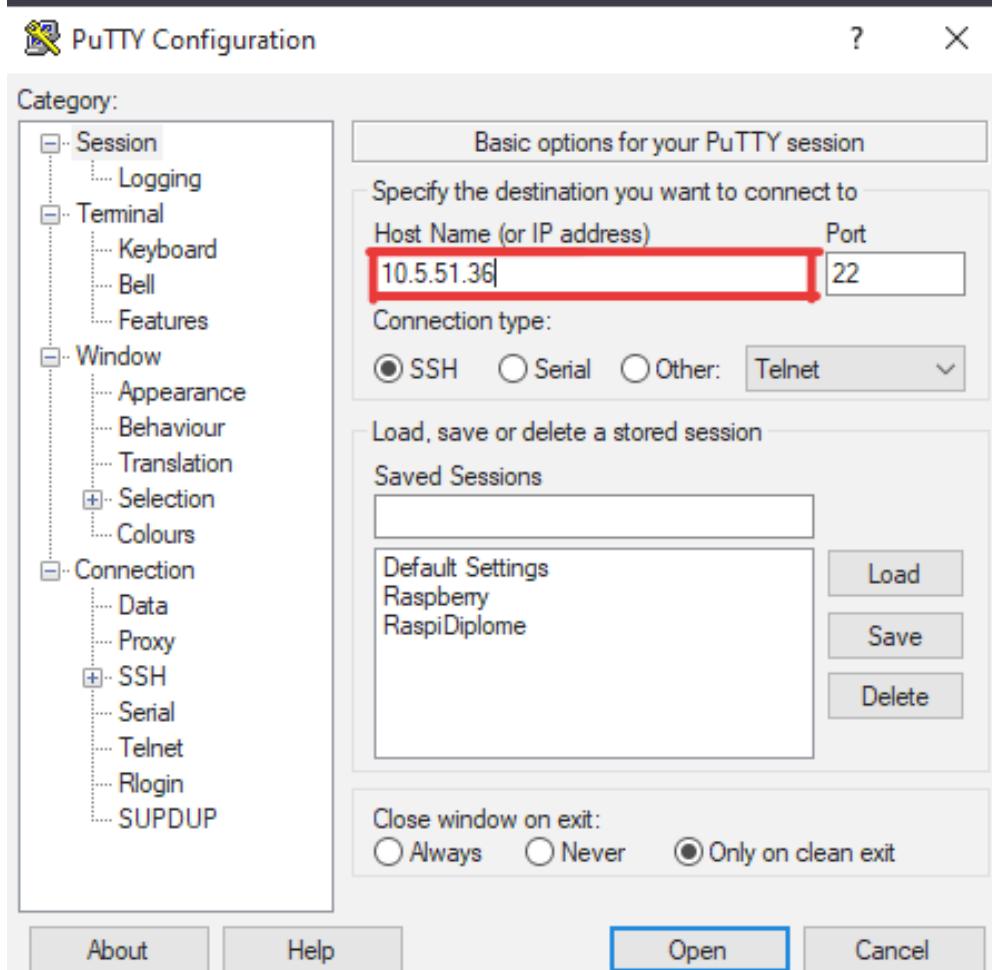
Pendant que le raspberry pi s'allume téléchargez et installez le logiciel [puTTY](#).

Ensuite, allez sur un ordinateur et cherchez votre raspberry pi dans les connections WiFi

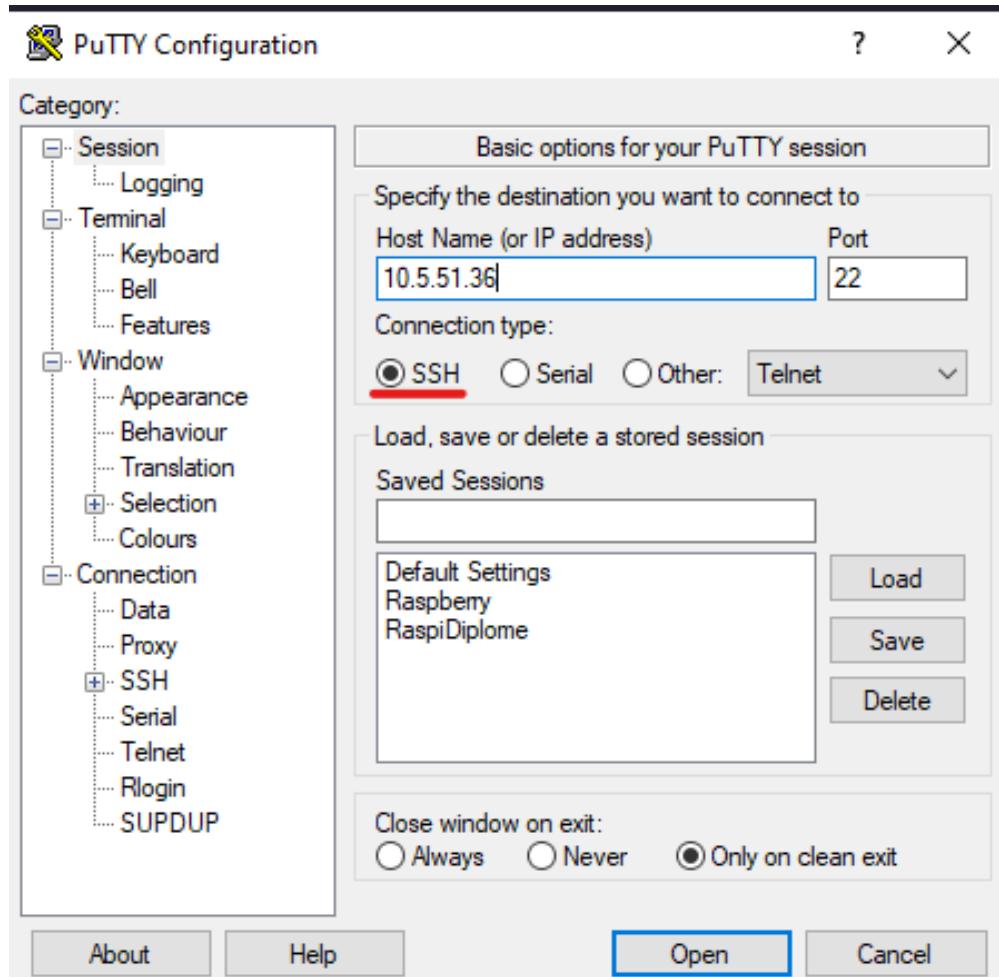


Puis, si vous avez bien suivis l'installation, vous pouvez vous connectez au raspberry pi.

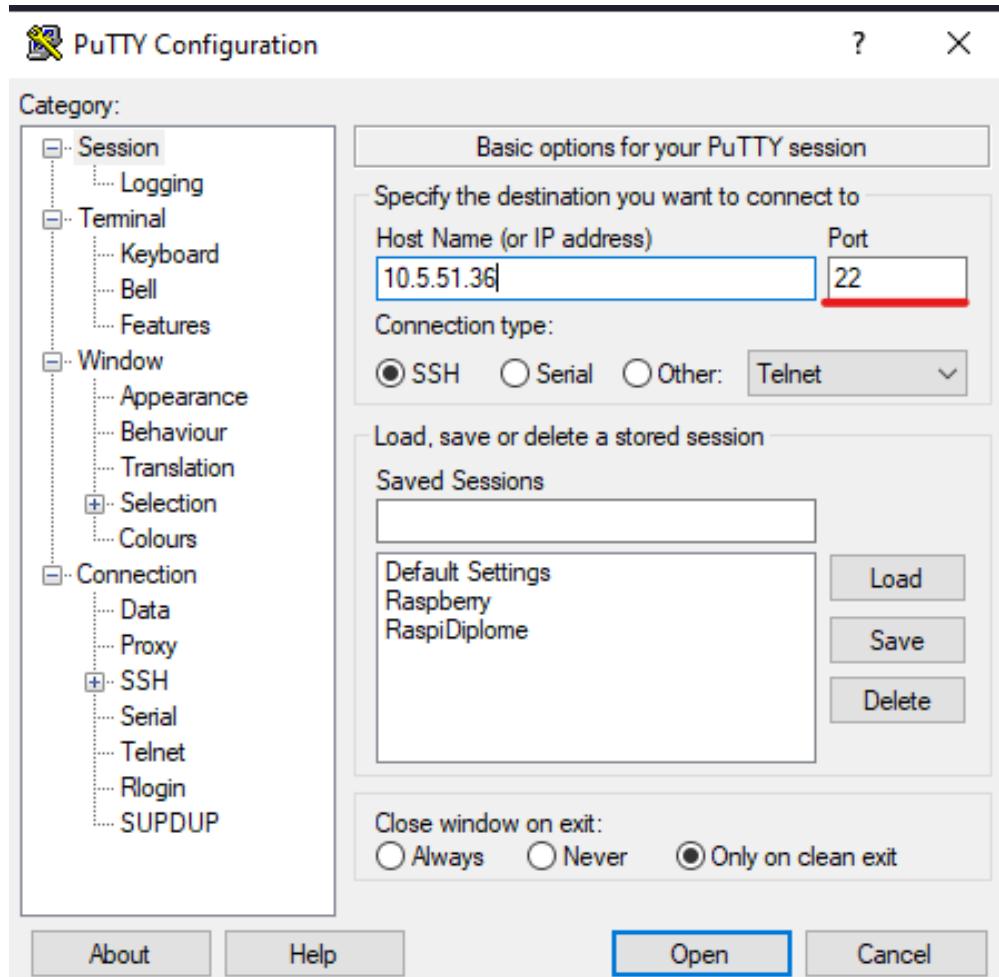
Dès que vous êtes connecté, lancez le programme puTTY et entrez-y l'ip de votre raspberry pi.



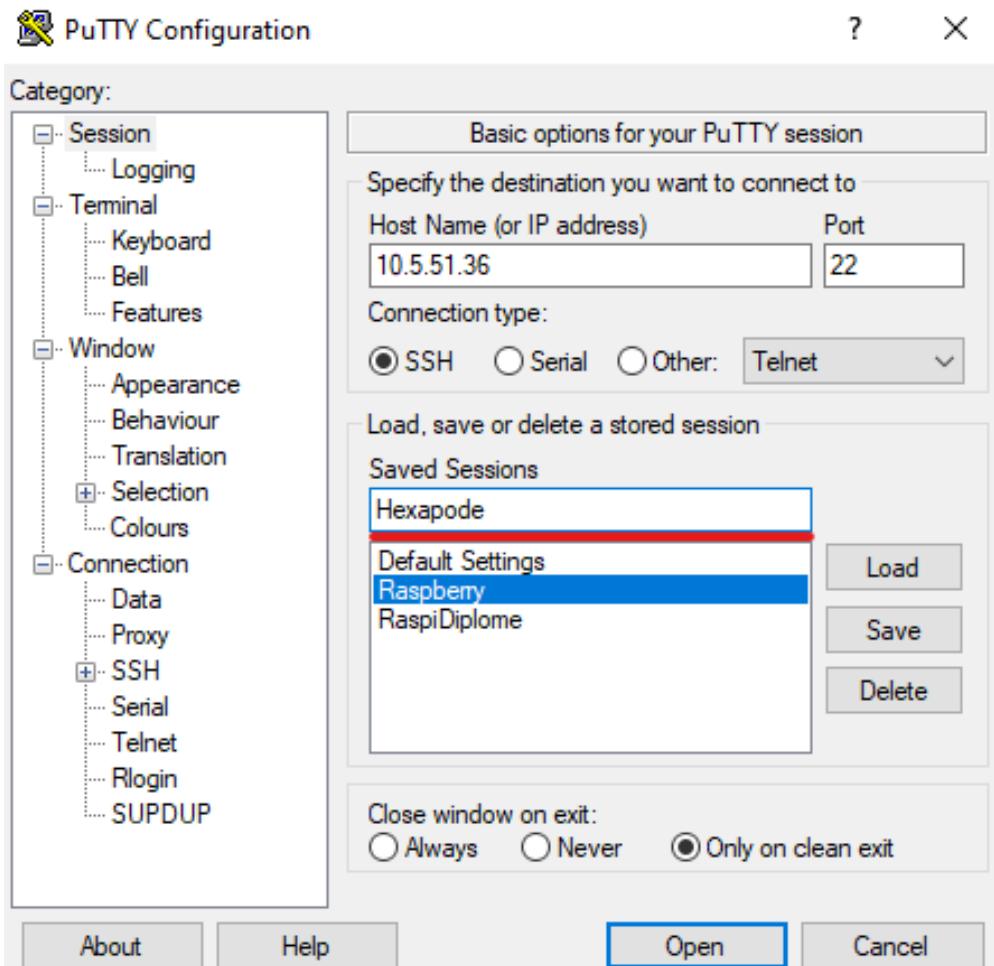
Vérifiez que vous soyez bien en SSH.



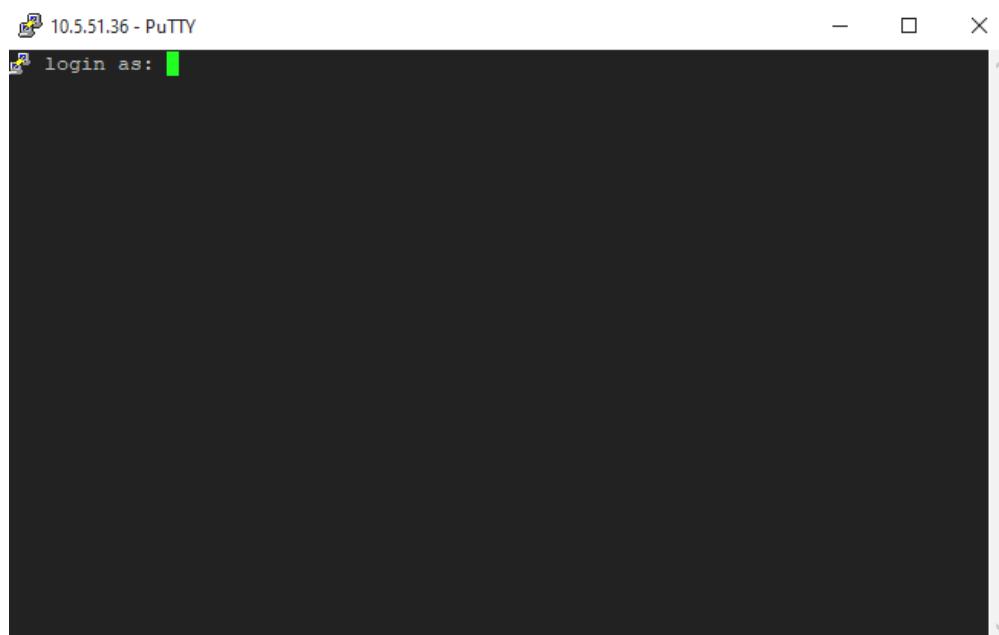
Puis, entrez le numéro de port 22.



Afin que ce soit plus simple pour les prochaines fois je vous conseille de l'enregistrer en bas, ainsi, les prochaines fois vous n'avez qu'à cliquer dessus pour vous connecter.



Maintenant que vous êtes connecté en SSH au raspberry pi, il vous demandera de vous identifier (comme sur l'image en dessous). Si vous avez tout laisser par défaut voici le login est pi.



Ensuite il vous demandera le mot de passe qui, si vous ne l'avez pas changer, est raspberry.

### 6.1.2 Lancement du programme

Dans cette étape, on va lancer le programme.

Dans le terminal pyTTY, tapez :

```
python3 Hexapode/app/app.py
```

Voilà le programme est lancé!

S'il ne s'est pas lancé vérifiez bien que vous avez cloner le répertoire git dans le home du raspberry pi.

Si ce n'est pas le cas, tapez juste

```
cd
```

Cette commande vous enverra à l'home du raspberry pi.

Et maintenant tapez:

```
git clone https://github.com/NickVanMarkes/Hexapode.git
```

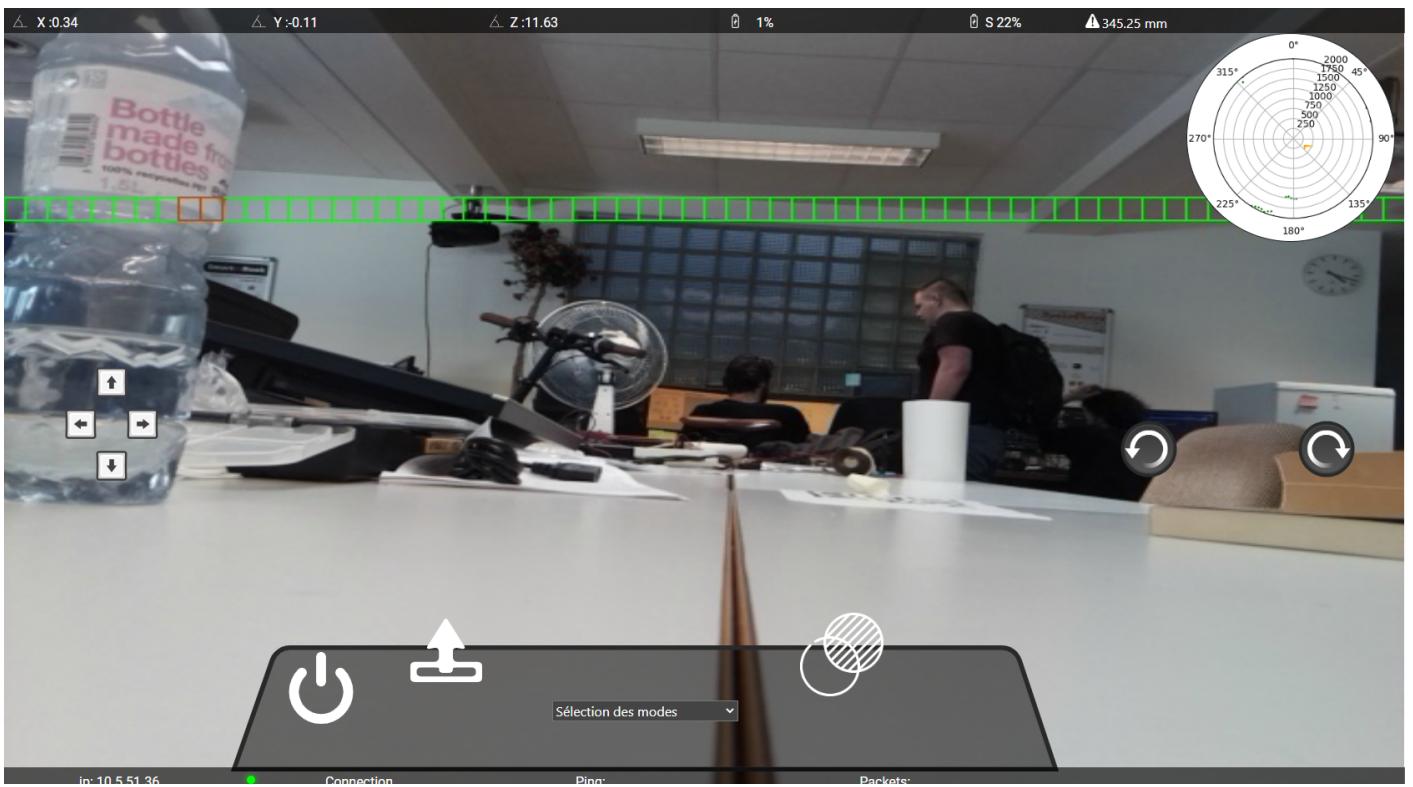
Puis refaites l'étape du lancement du programme.

### 6.1.3 Utilisation du programme

Dès que tout sera bon, le programme vous montrera plein de message, mais surtout l'adresse ip à vous connecter sur votre navigateur web.

```
* Serving Flask app 'app' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
Stoping RPLidarException.
* Running on all addresses.
  WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://10.5.51.36:5000/ (Press CTRL+C to quit)
```

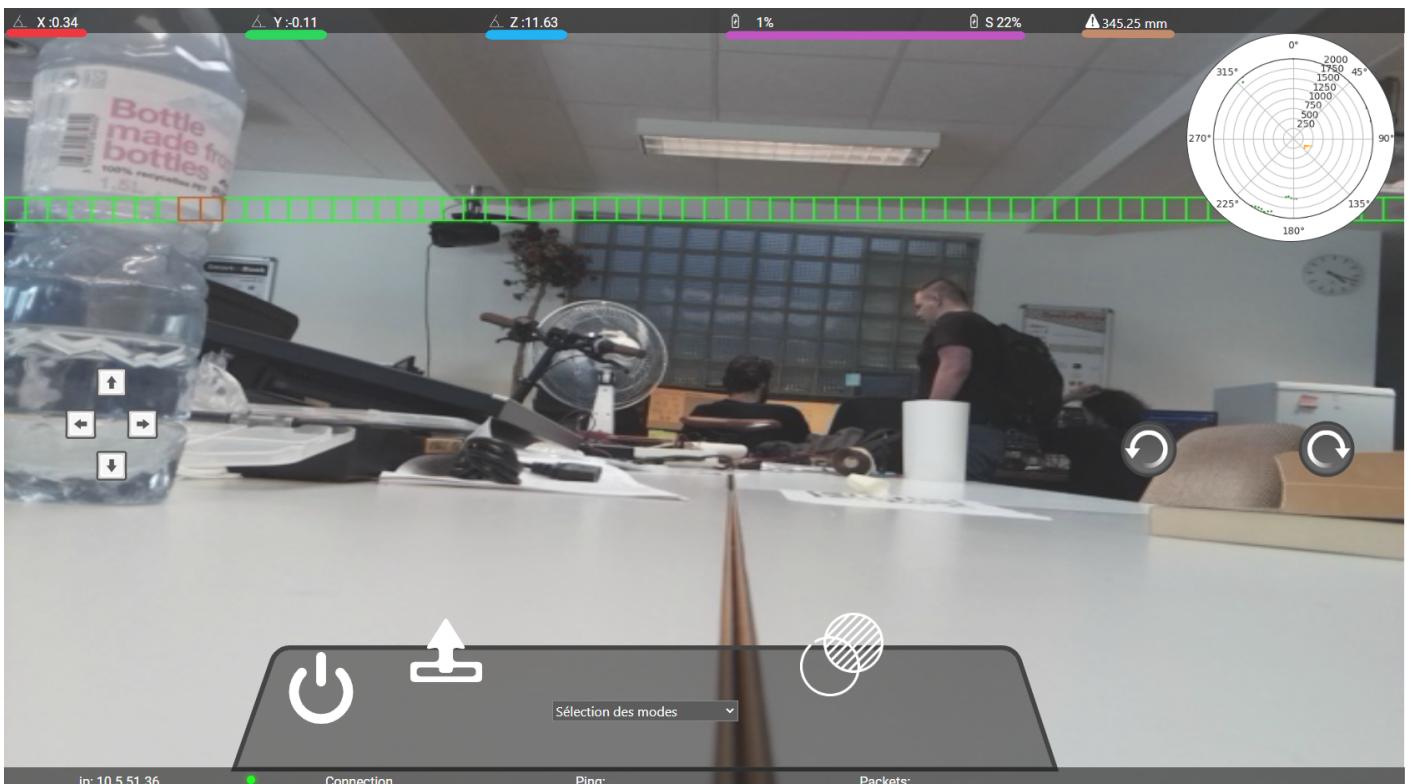
Tapez cette adresse sur votre navigateur puis vous verrez l'interface web:



## 6.1.4 Utilisation de l'interface

### Bannière

Tout d'abord je vous explique les informations que vous retrouvez sur la bannière du haut:



En rouge, nous avons l'orientation du robot sur l'axe X. L'axe X est l'orientation direction la caméra, savoir si le robot est incliné vers l'avant ou vers l'arrière.

En vert, nous avons l'orientation du robot sur l'axe Y. L'axe X est l'orientation direction des pattes, savoir si le robot est incliné vers la gauche ou vers la droite.

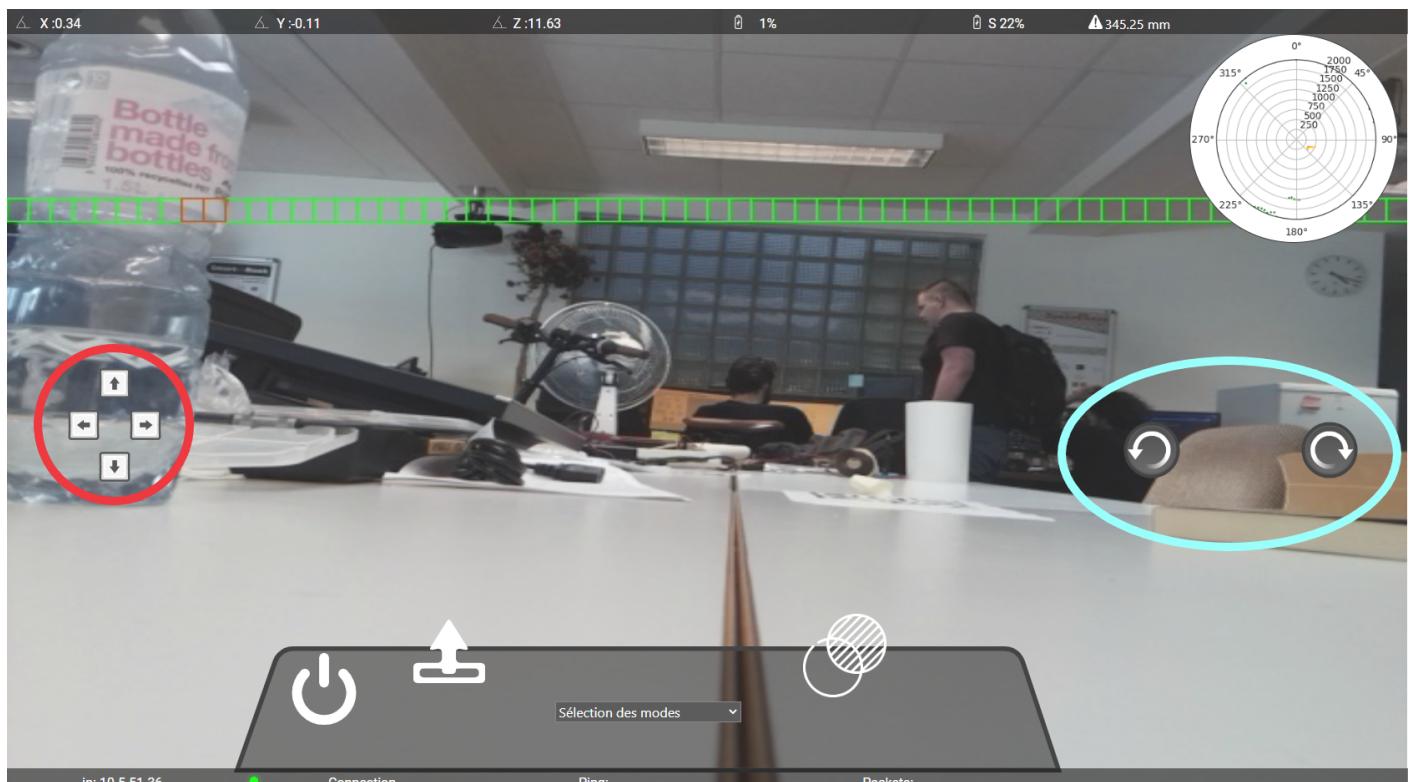
En bleu, nous avons l'orientation du robot sur l'axe Z. L'axe Z est l'orientation de rotation du robot, savoir si le robot est tourné vers le nord, sud, est ou ouest.

En violet, il y a le pourcentage de batterie restant sur les deux batteries. La batterie de gauche, est celle pour le raspberry pi (le cerveau du robot), et celle de droite pour les servomoteurs (les muscles du robot).

En brun, nous avons la distance de l'obstacle détecté le plus proche. La distance est toujours affichée en millimètres.

### Boutons de directions

Dans cette partie, je vous explique quels sont les boutons de directions et à quoi ils servent.



En rouge, nous avons quatres boutons. Ces quatres boutons servent à se déplacer comme sur un cadrillage.

- La flèche du haut:

Ce bouton sert à faire avancer le robot d'un pas. Cette touche peut être activée aussi avec la touche **W** du clavier.

- La flèche de gauche:

Ce bouton sert à faire un pas vers la gauche (comme un crabe). Cette touche peut être activée aussi avec la touche **A** du clavier.

- La flèche de droite:

Ce bouton sert à faire un pas vers la droite (comme un crabe). Cette touche peut être activée aussi avec la touche **D** du clavier.

- La flèche du bas:

Ce bouton sert à faire reculer le robot d'un pas. Cette touche peut être activée aussi avec la touche **S** du clavier.

En bleu, nous avons deux boutons qui servent à faire les rotations du robot.

- La flèche de gauche:

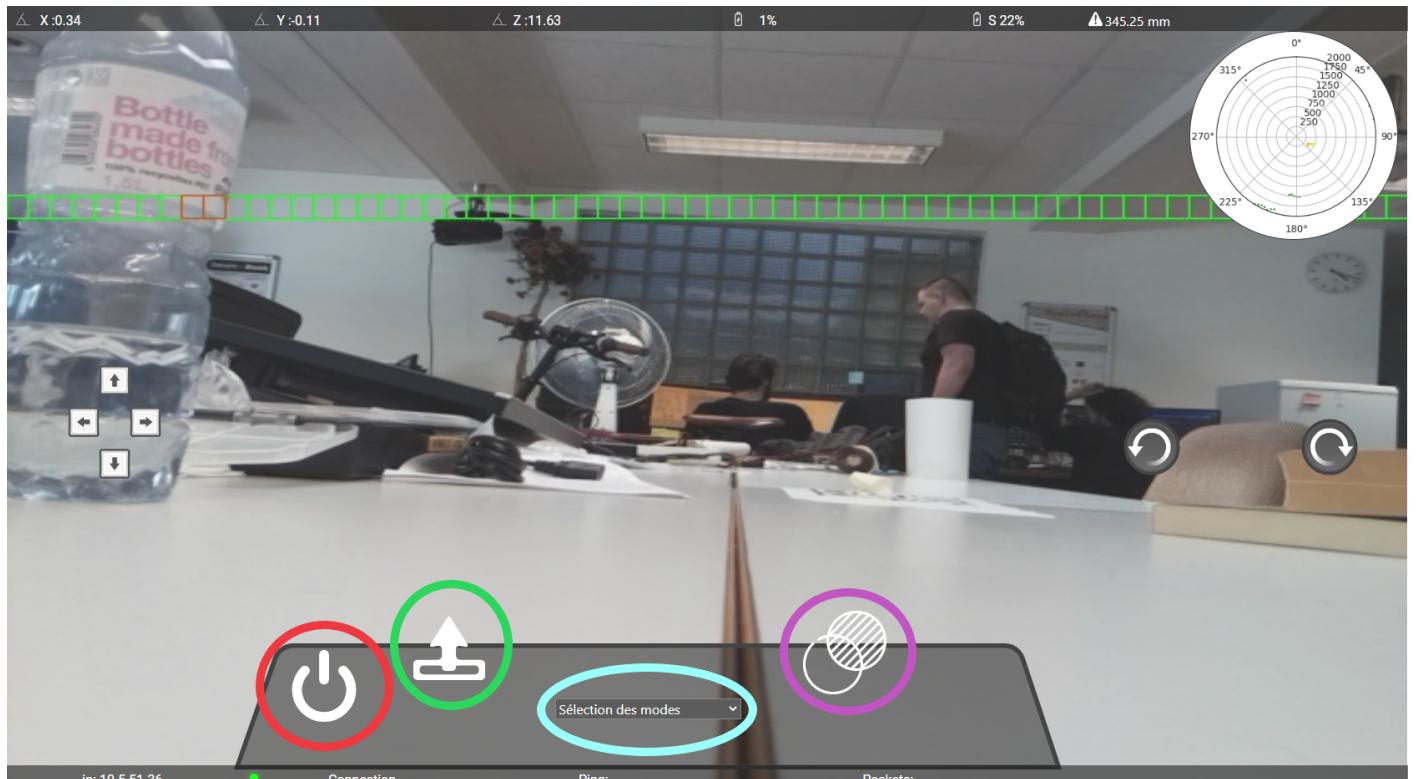
Ce bouton fait faire au robot une rotation anti-horaire. Cette touche peut être activée aussi avec la touche **Flèche de gauche** du clavier.

- La flèche de droite:

Ce bouton fait faire au robot une rotation horaire. Cette touche peut être activée aussi avec la touche **Flèche de droite** du clavier.

### Bandeau

Dans le bandeau, il y a quelques boutons ainsi que le selecteur de modes.



En rouge, c'est le bouton qui sert à éteindre le raspberry pi(ce qui éteindra le robot). Cette touche peut être activée aussi avec la touche **X** du clavier.

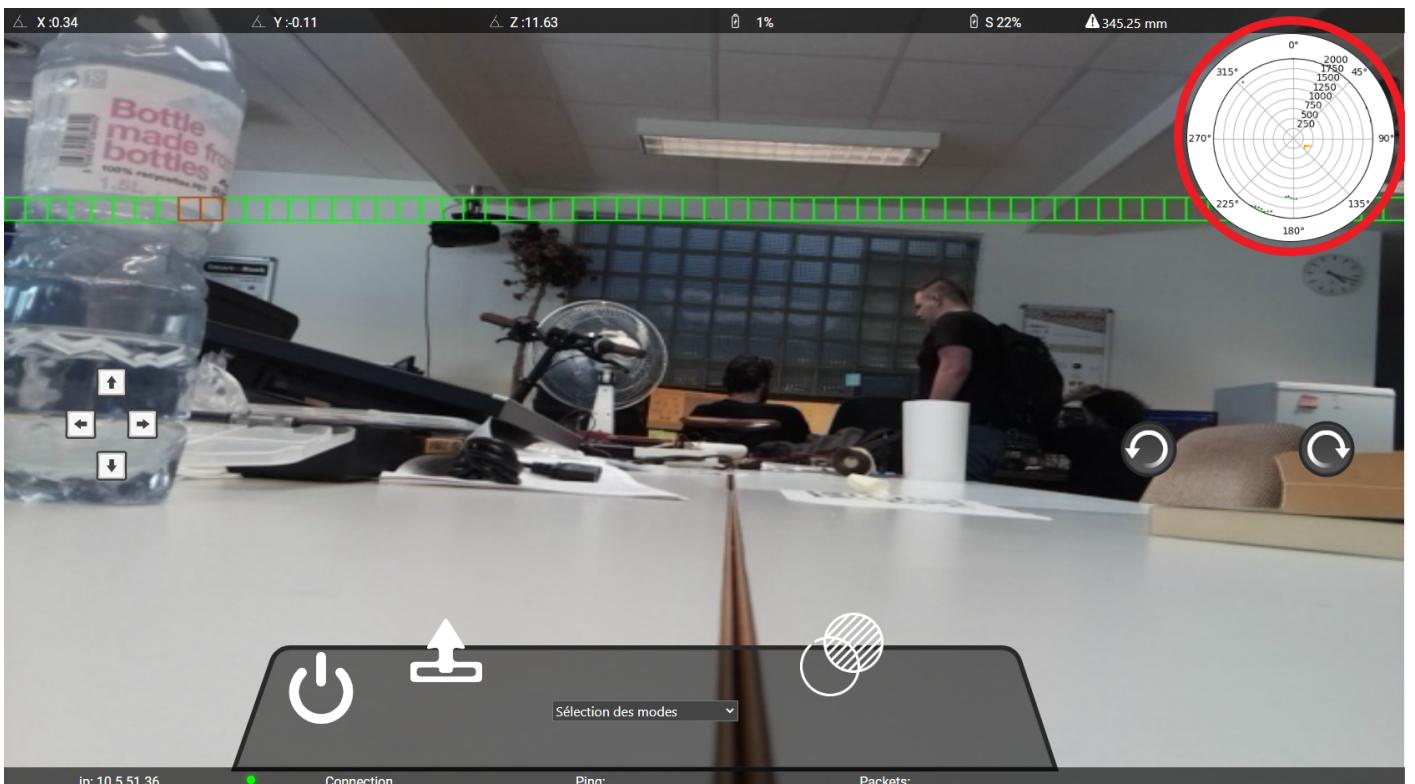
En vert, c'est le bouton d'initialisation du robot. Il sert au démarrage de faire lever le robot. Cette touche peut être activée aussi avec la touche **I** du clavier.

En bleu, c'est le selecteur de modes. Il faut cliquer dessus et plusieurs choix de modes s'afficheront, et vous n'avez qu'à cliquer sur le mode que vous voulez et ça le changera.

En violet, c'est le bouton pour changer la transparence de la vue radar. Elle change de blanc à transparent ou vice-versa la vue radar. Cette touche peut être activée aussi avec la touche **T** du clavier.

### Vue Radar

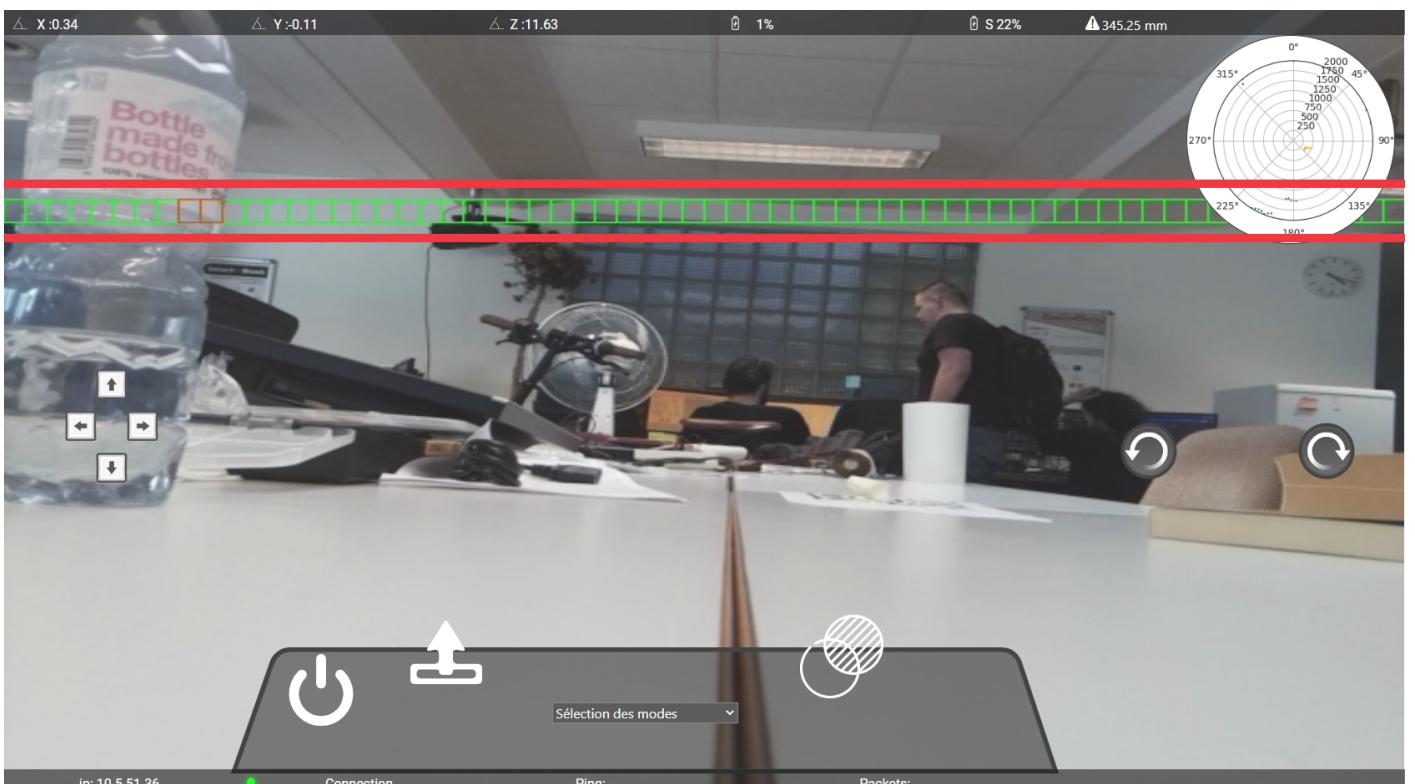
Sur l'interface, en haut à droite, il y a une vue radar, qui montre où sont les obstacles autour du robot.



Dans le cercle rouge, nous avons une vue radar des obstacles autour du robot. Si vous voulez voir plus précisément les obstacles, cliquez dessus et la vue s'agrandira.

#### Détecteur d'obstacles Caméra

Sur le retour de la caméra, il y a une ligne de carré qui représente les obstacles les plus proches du robot.



Plus un obstacle sera proche, plus le carré deviendra rouge. Plus il est loin, plus il sera vert.

## 6.2 Manuel d'installation

### 6.2.1 Prérequis

Afin que les versions coïncident ainsi que le matériel, vous devez avoir:

- Un poste fonctionnel
- Raspberry pi 3
- Carte SD(16GB minimum) + adaptateur
- Accès à internet

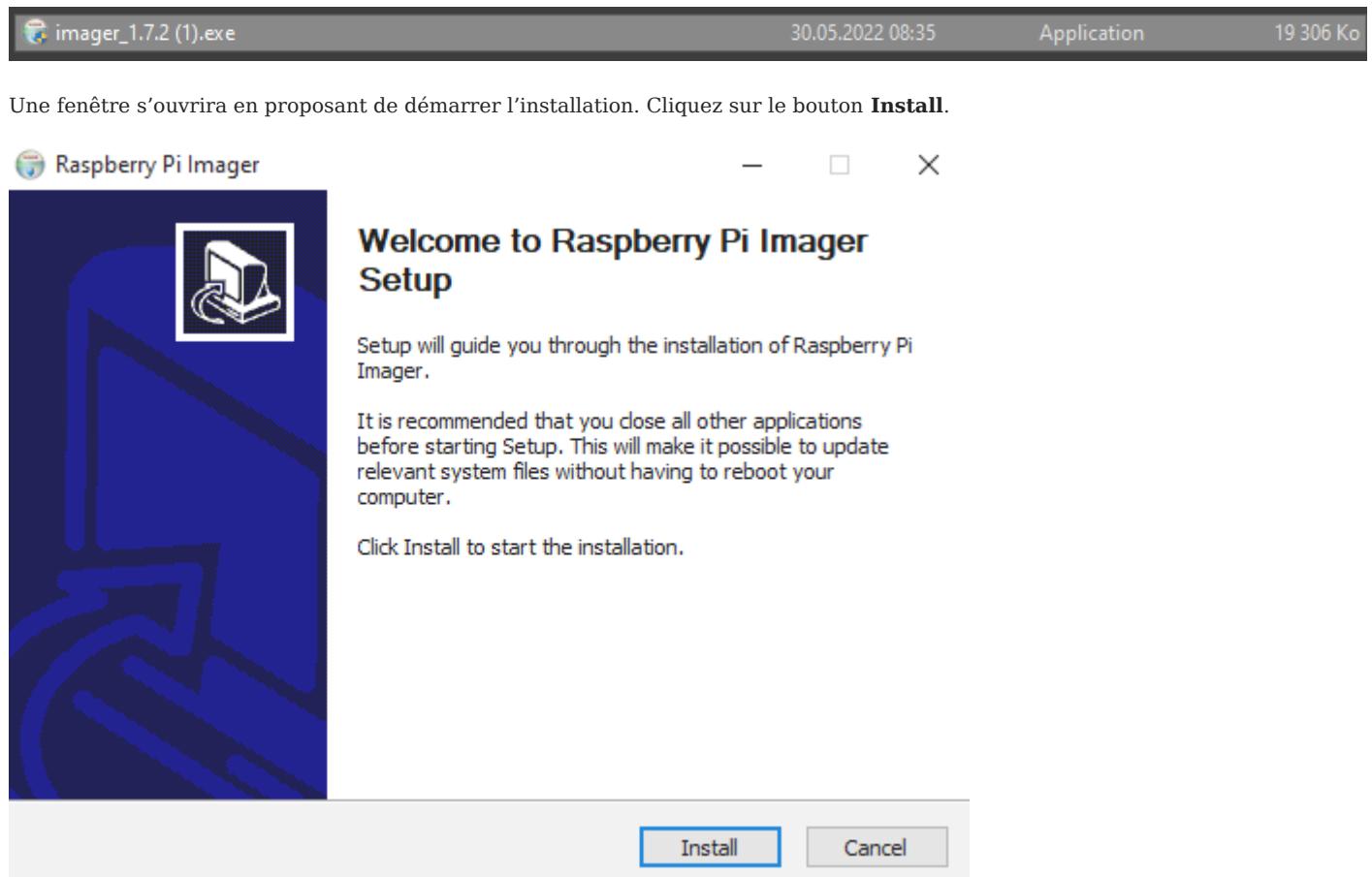
### 6.2.2 Téléchargement de l'OS Raspbian 32 bits

#### Installation de Raspberry pi Imager

Pour commencer, téléchargez l'application [Raspberry pi Imager](#), cela permettra de mettre l'OS sur la carte SD.

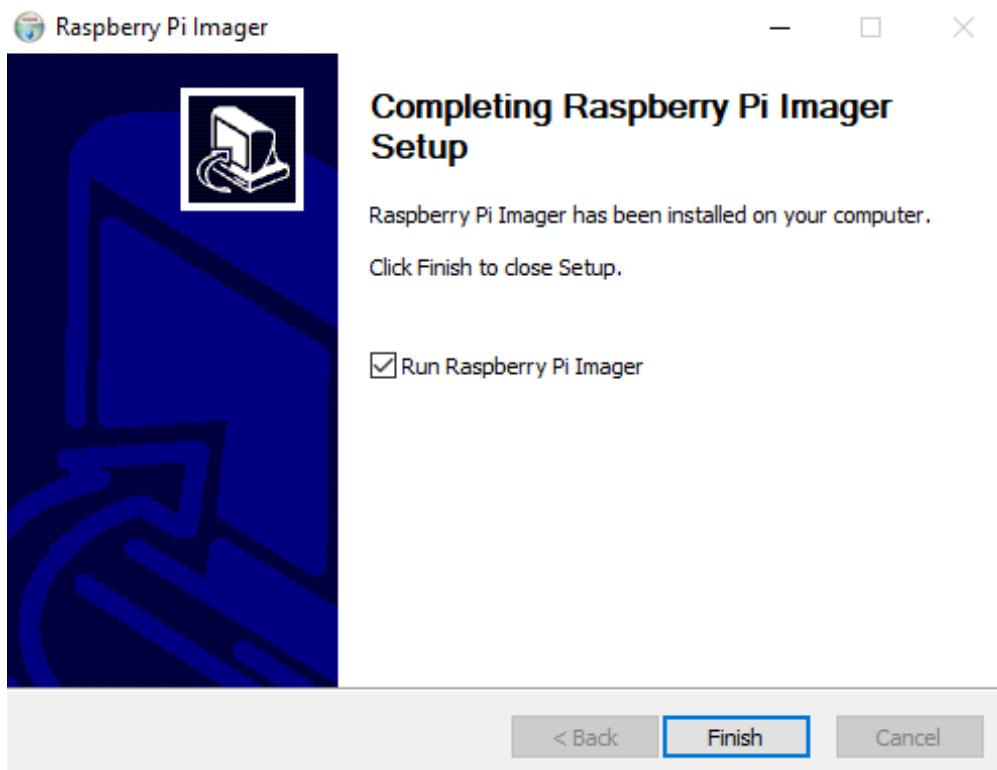
Pour cette étape, je l'ai fait sur un poste Windows, de ce fait j'ai juste téléchargé leur installateur. Une fois l'application téléchargée, il faut l'installer.

Faites un double-clic sur l'exécutable téléchargé.



L'application s'installera. Une fois l'installation terminée, le programme vous proposera de démarrer l'application.

Cliquez sur le bouton **Finish**.



#### Utilisation de Raspberry Pi Imager

Une fois l'application démarrée, vous devez cliquer sur le bouton **CHOISISSEZ L'OS**



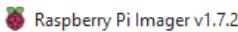
Cela ouvrira une fenêtre vous permettant de choisir l'OS que vous voulez installer sur le raspberry pi.

Pour ma part, j'utilise celui proposé en premier, **Raspberry Pi OS (32-bit)**.

**Système d'exploitation**

|   |  |   |
|---|--|---|
|  | <b>Raspberry Pi OS (32-bit)</b><br>A port of Debian Bullseye with the Raspberry Pi Desktop (Recommended)<br>Sorti le : 2022-04-04<br>Mis en cache sur votre ordinateur | X |
|  | <b>Raspberry Pi OS (other)</b><br>Other Raspberry Pi OS based images   | > |
|  | <b>Other general-purpose OS</b><br>Other general-purpose operating systems   | > |
|  | <b>Media player OS</b><br>Media player operating systems   | > |
|  | <b>Emulation and game OS</b>   | < |

Ensuite, cela vous ramènera à nouveau sur la page principale, et cliquez sur le bouton du milieu **CHOISISSEZ LE STOCKAGE**....

 Raspberry Pi Imager v1.7.2

  
**Raspberry Pi**

**Système d'exploitation**      **Stockage**

|                          |                     |        |
|--------------------------|---------------------|--------|
| RASPBERRY PI OS (32-BIT) | CHOISISSEZ LE ST... | ÉCRIRE |
|--------------------------|---------------------|--------|



Veuillez sélectionner le lecteur sur lequel vous voulez installer l'OS. Pour ma part, je ferais l'installation sur **Kingston UHSII uSD USB Device - 15.9 GB**.

**Stockage**

|   |   |   |
|---|---|---|
|  | KINGSTON SV300S37A240G - 240.1 GB<br>Mounté à D:\                   | X |
|  | Kingston UHSII uSD Reader USB Device - 15.9 GB<br>Mounté à E:\, F:\ |   |

Et enfin de finir avec cette partie, cliquez sur **ÉCRIRE** afin de démarrer l'installation.



**⚠️ Attention** ⚠️ Vérifiez qu'il n'y ait aucune donnée dans le support où vous allez installer l'OS. Toutes les données seront écrasées!

Si vous n'avez rien d'important dedans, alors vous pouvez cliquer sur le bouton **OUI**, sinon cliquez sur le bouton **NON** et prenez le temps de sauvegarder vos données.



Maintenant, vous n'avez plus qu'à attendre la fin de l'installation.



S'il n'y a eu aucune erreur durant l'installation, vous aurez ce message vous indiquant que l'opération est terminée et avec succès.

Vous pouvez à présent cliquer sur **CONTINUER** et fermer le programme.



## 6.2.3 VNC Server

Sur le raspberry pi, téléchargez [REALVNC](#).

Pour cela, il vous suffit de sélectionner **Raspberry Pi** et d'ensuite cliquer sur le bouton **Download VNC Server**.

Téléchargez VNC® Connect sur l'ordinateur à contrôler.

Ensuite, téléchargez [VNC Viewer](#) sur l'appareil depuis lequel vous souhaitez exercer le contrôle à distance.



Si vous utilisez Raspbian Jessie ou version ultérieure, VNC® Connect est préinstallé. Vous ne devez donc le télécharger que si votre Raspberry Pi est équipé d'une autre distribution Linux.

Sur Raspberry Pi uniquement, VNC® Connect est en outre déjà installé avec une [licence](#) pour offrir la connectivité cloud et la connectivité directe aux abonnés Home.

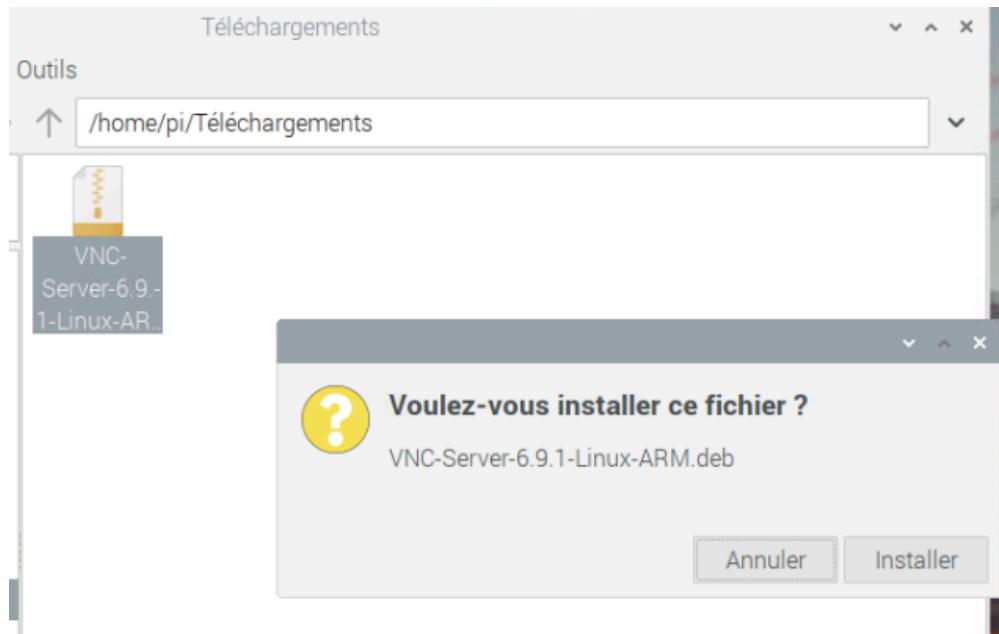
[Download VNC Server](#)

SHA256: 140a4f452b3326e9e257c878f9dd0fc9ab6a6ef3ad489fd9132821354e99deb3

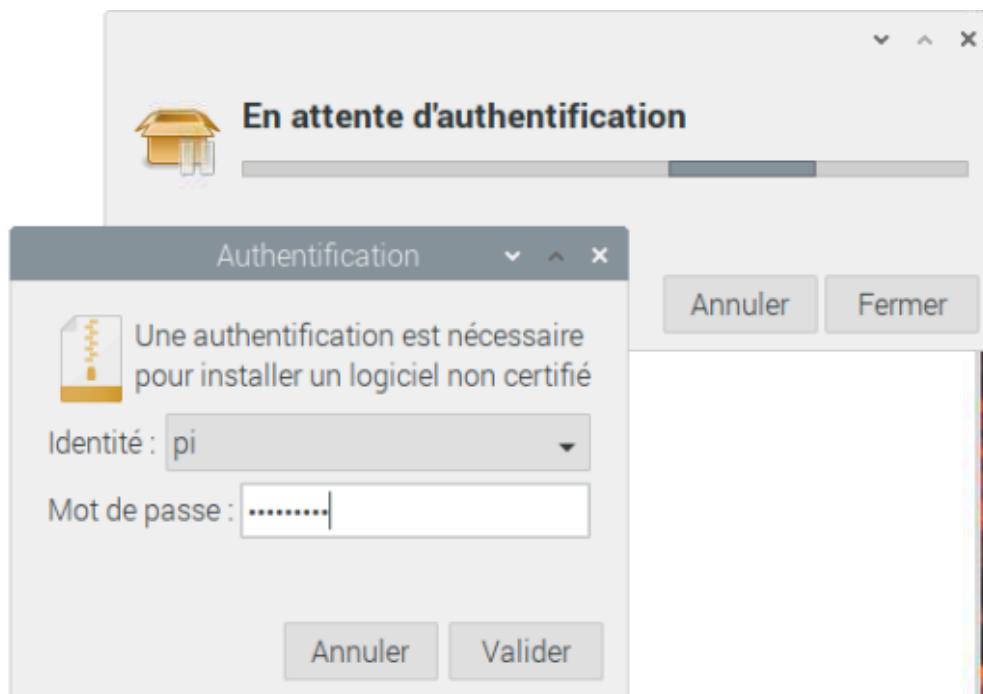
armhf ▾

Une fois le téléchargement terminé, allez dans votre dossier téléchargement et exécutez le fichier d'installation.

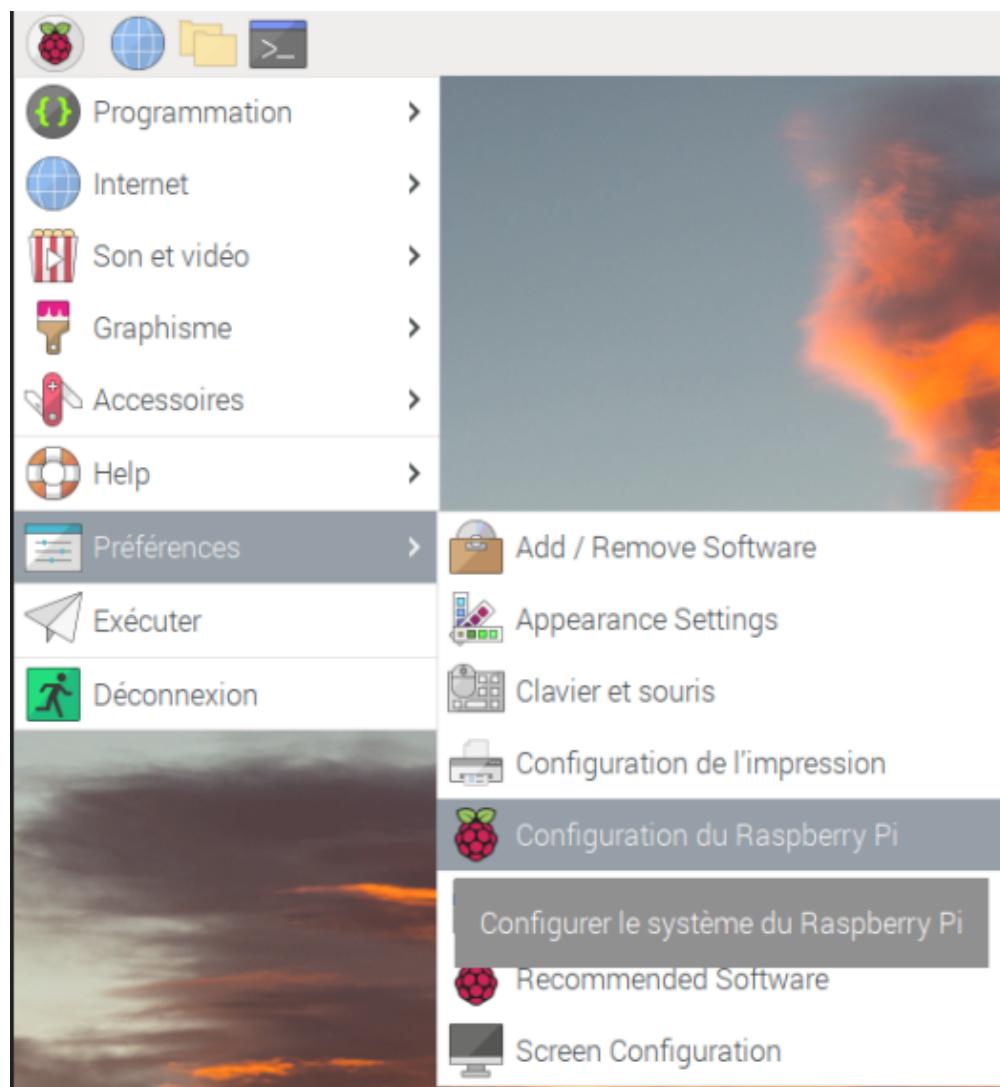
Un message est censé apparaître, appuyez sur le bouton **Installer**.



Après une petite attente, une nouvelle fenêtre s'ouvrira, **Authentification**. Insérez le mot de passe du raspberry pi, par défaut le mot de passe est: raspberry



Une fois l'installation terminée, allez sur le menu du raspberry, Préférences -> Configuration du Raspberry Pi.



Ensuite, cliquez sur le bouton en face de **VNC**. Cela activera VNC Server et permettra ainsi la connexion au Raspberry via VNC.

Dès que c'est fait, cliquez sur **Valider**.



Une fenêtre vous indiquant qu'il faut redémarrer va apparaître. Cliquez sur le bouton **Yes** afin de redémarrer.

Les changements que vous avez faits sur votre Raspberry Pi requièrent un redémarrage pour être opérationnels.

Voulez-vous redémarrer maintenant ?

No

Yes

Et voilà votre VNC Server est configuré et accessible!

Pour vous connecter sur votre raspberry pi, ouvrez un terminal de commande et tapez:

```
ifconfig
```

Appuyez sur enter.

Au début de la réponse, il y a l'adresse IP de votre Raspberry. La mienne est 10.5.51.36.

```
pi@raspberrypi:~ $ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 10.5.51.36 netmask 255.255.255.0 broadcast 10.5.51.255
              inet6 fe80::4d:e411:5b5b:6ce8 prefixlen 64 scopeid 0x20<link>
                ether b8:27:eb:b5:23:c4 txqueuelen 1000 (Ethernet)
                  RX packets 1244612 bytes 847969473 (808.6 MiB)
                  RX errors 0 dropped 8 overruns 0 frame 0
                  TX packets 333322 bytes 56435852 (53.8 MiB)
                  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Connaître l'adresse IP est nécessaire afin de pouvoir se connecter sur la machine qui contrôlera le raspberry pi.

## 6.2.4 VNC Viewer

### Installation

Maintenant que le VNC Server est installé et configuré, nous pouvons nous lancer sur l'installation du VNC Viewer.

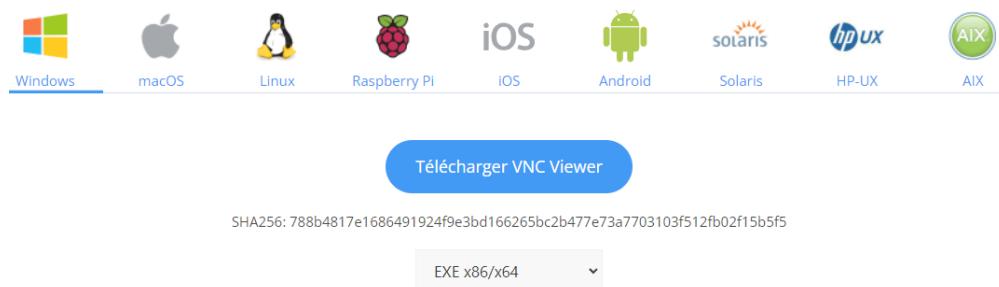
Pour commencer, allez sur [REALVNC](#) et lancer le téléchargement.

Choisissez votre système d'exploitation. (Windows personnellement)

Cliquez ensuite sur **Télécharger VNC Viewer**.

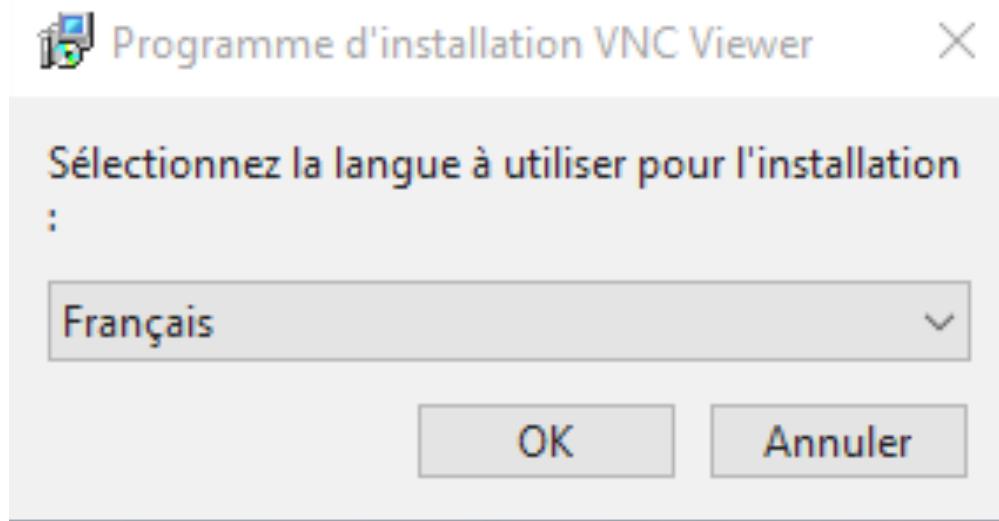
Téléchargez VNC Viewer sur l'appareil depuis lequel vous souhaitez exercer le contrôle à distance.

VNC® Connect doit être installé sur l'ordinateur à contrôler.

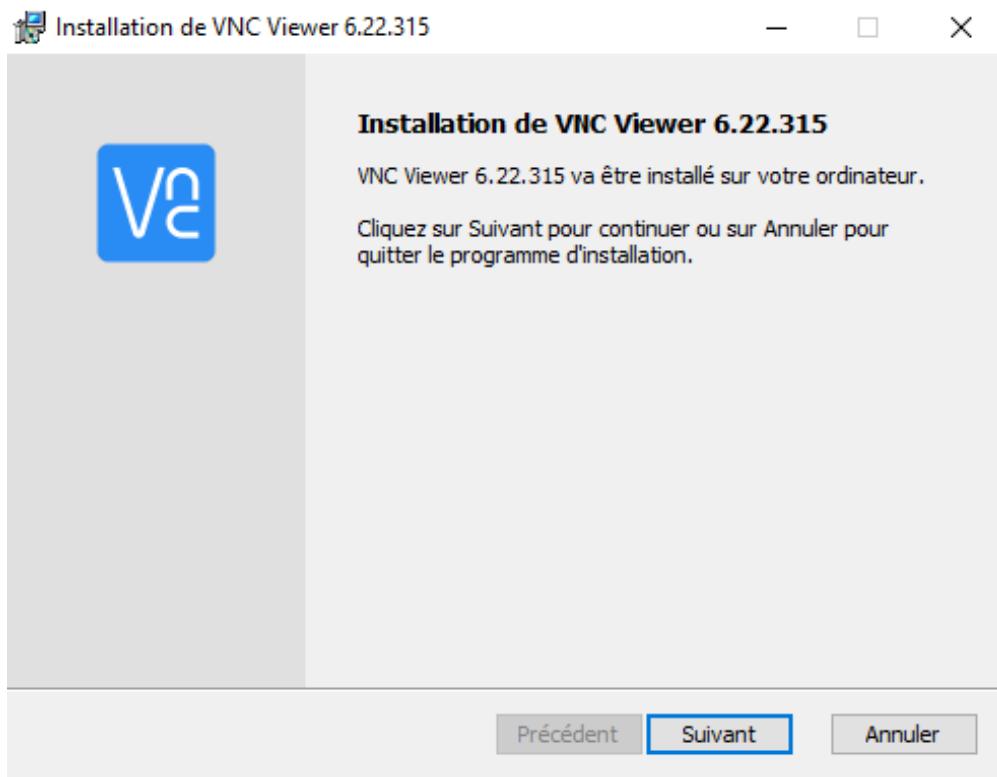


Allez sur l'installateur que vous avez téléchargé et exécutez-le.

Une fenêtre d'installation va apparaître, choisissez le langage que vous voulez utiliser. Une fois le choix effectué, appuyer sur **OK**.

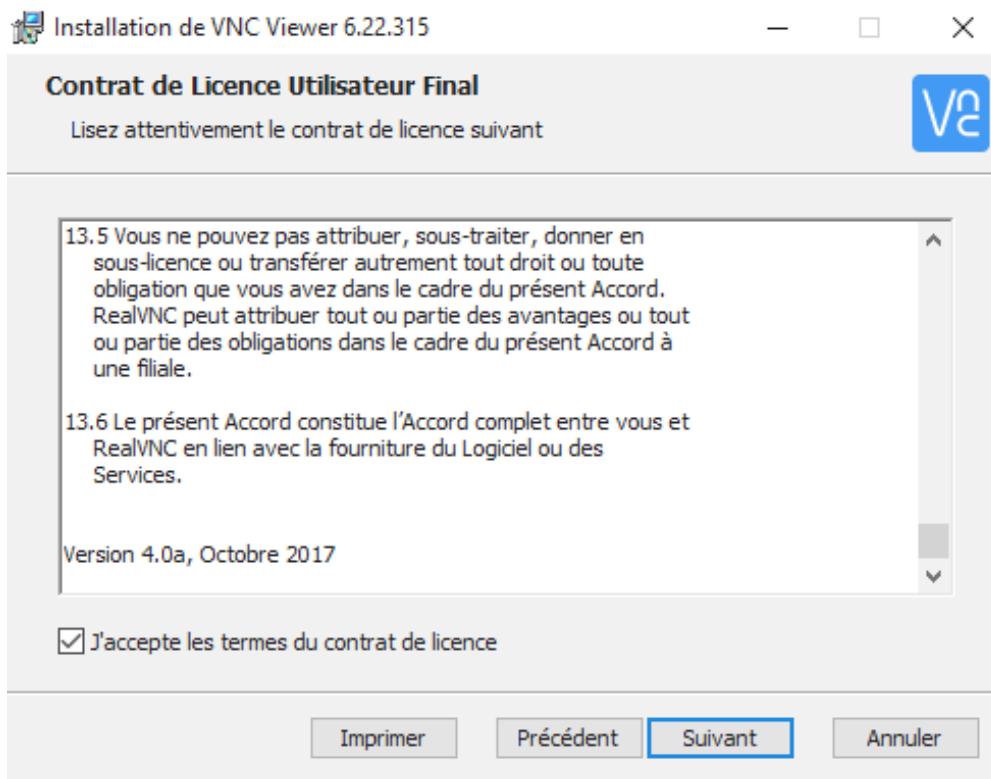


Appuyez sur le bouton **Suivant** pour commencer l'installation.



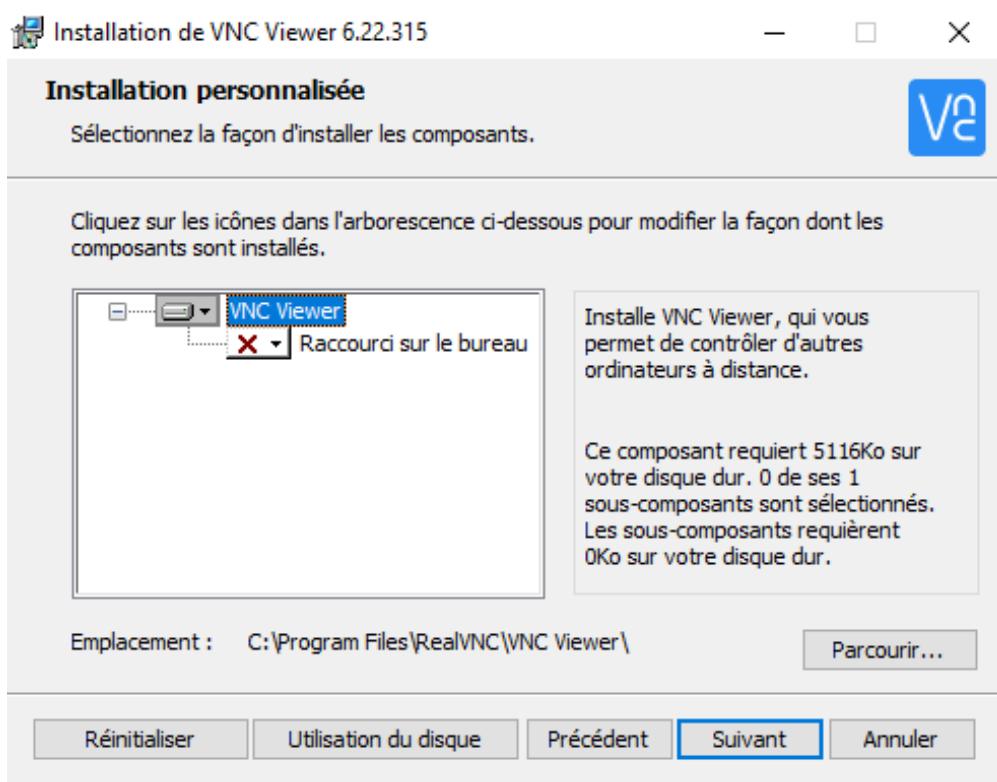
Lisez et acceptez les termes du contrat de licence.

Ensuite, appuyez sur le bouton **Suivant**.



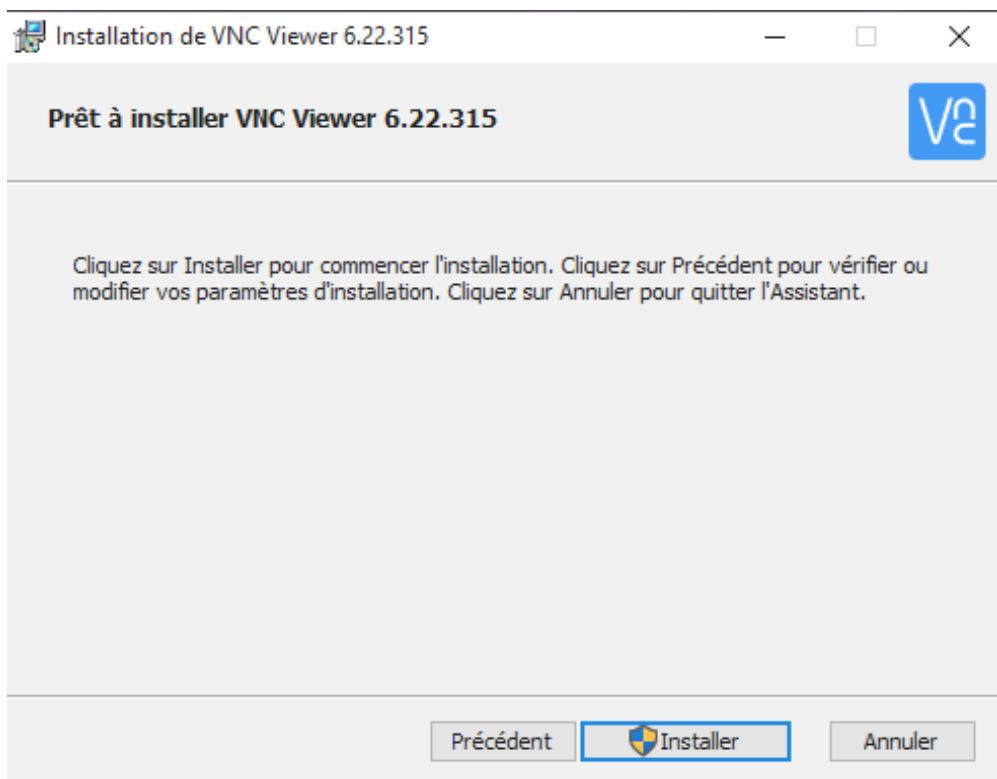
Si vous ne voulez pas modifier l'endroit d'installation, appuyez directement sur **Suivant**.

Sinon modifier l'endroit ou l'installer et appuyer ensuite sur **Suivant**.

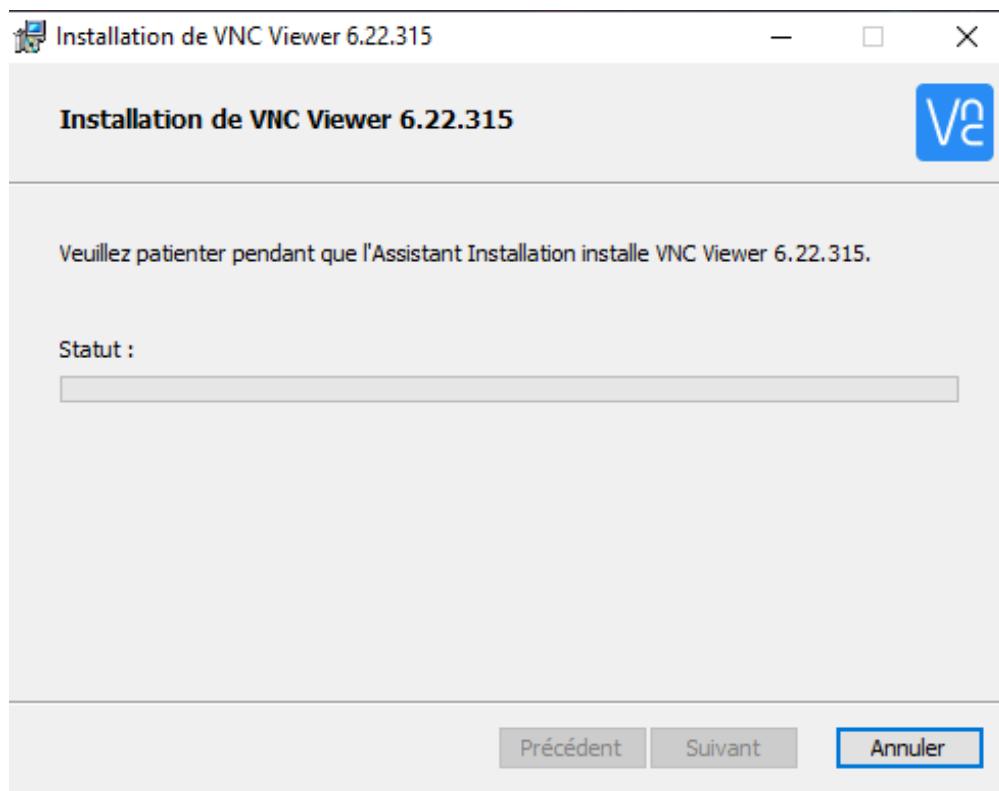


Une fois que vous avez terminé la configuration, cliquez sur **Installer**.

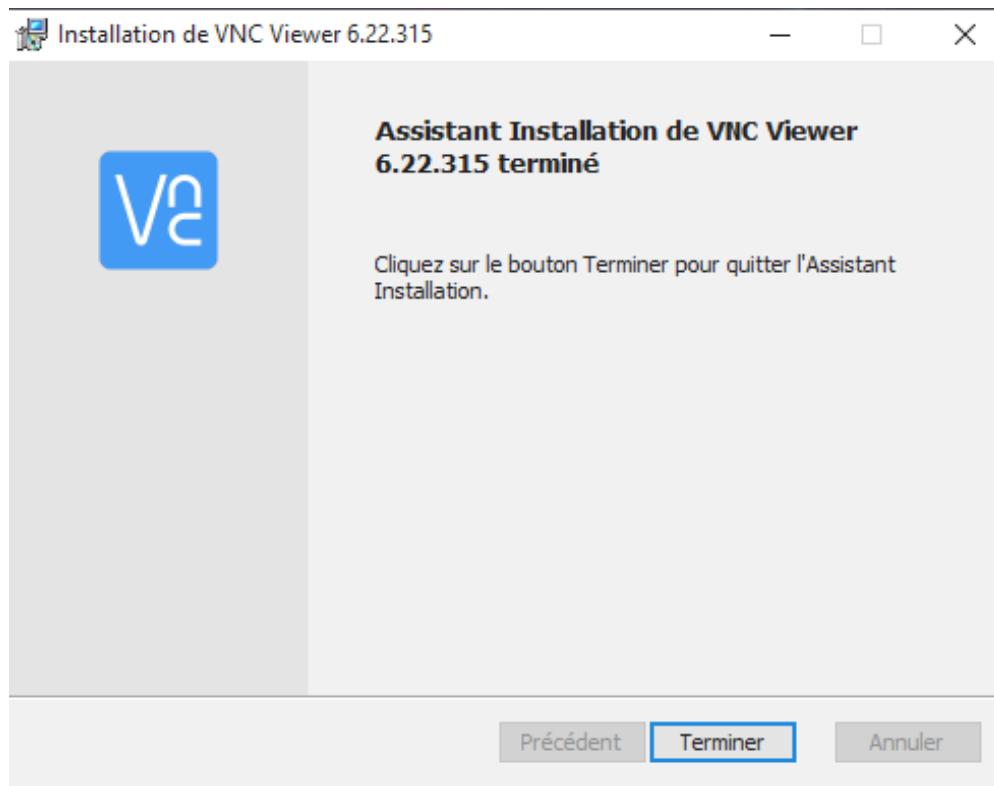
Une fenêtre peut apparaître sur votre écran, appuyer sur le bouton **Oui**.



La fenêtre d'installation devrait apparaître, il ne reste plus qu'à attendre.

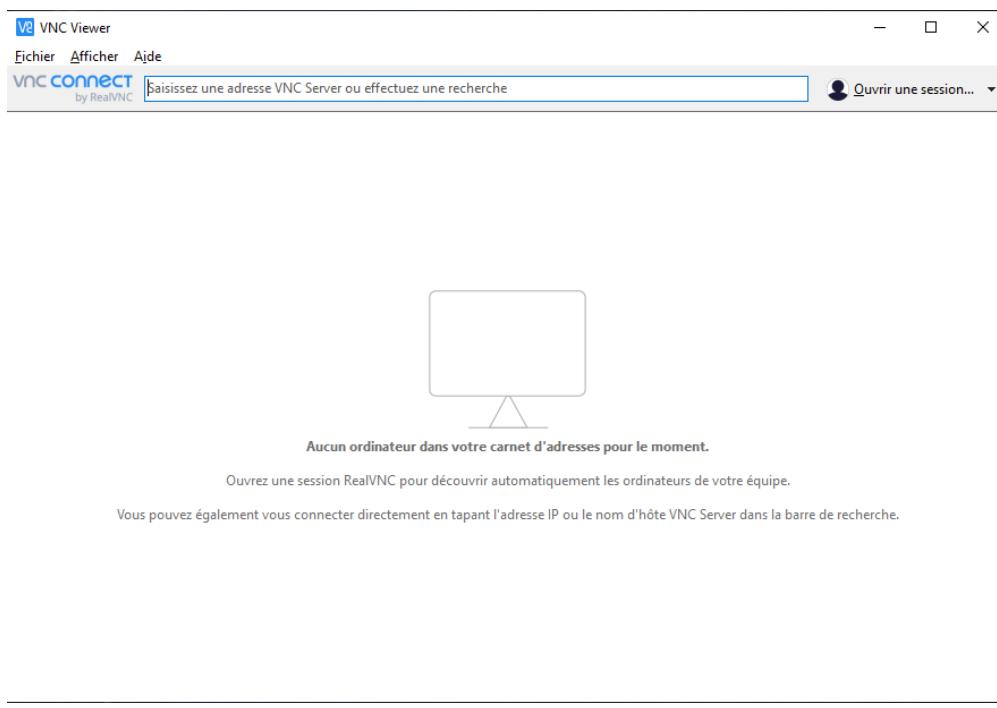


Une fois l'installation terminée, si tout s'est bien passé, il ne vous reste plus qu'à appuyer sur **Terminer**.



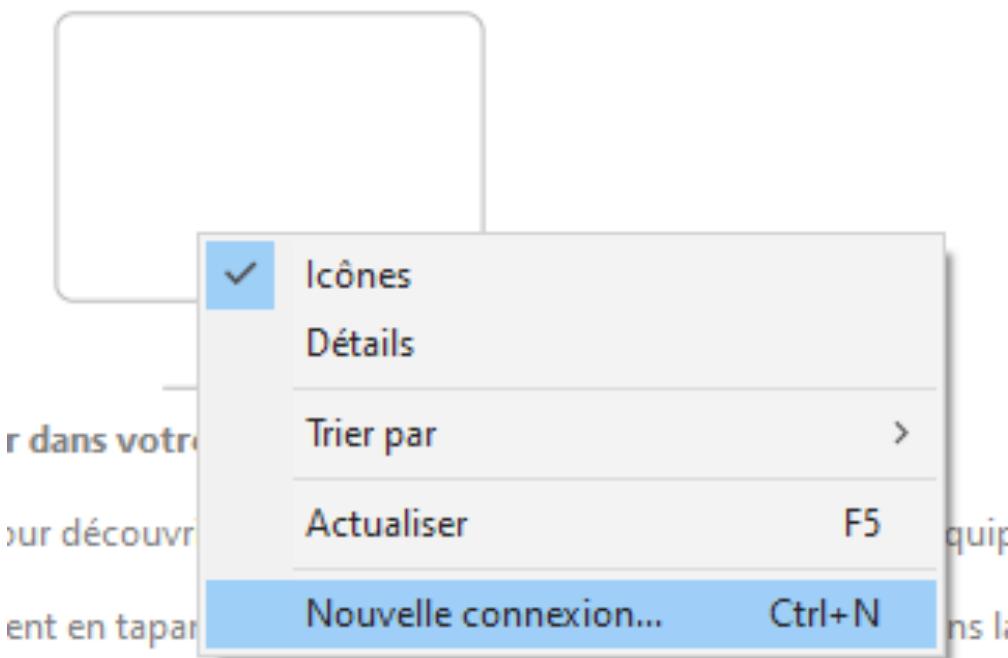
## Utilisation

Ouvrez VNC Viewer, l'application devrait ressembler à celle que j'ai.



Faites un clic droit sur le fond. Le menu contextuel devrait apparaître.

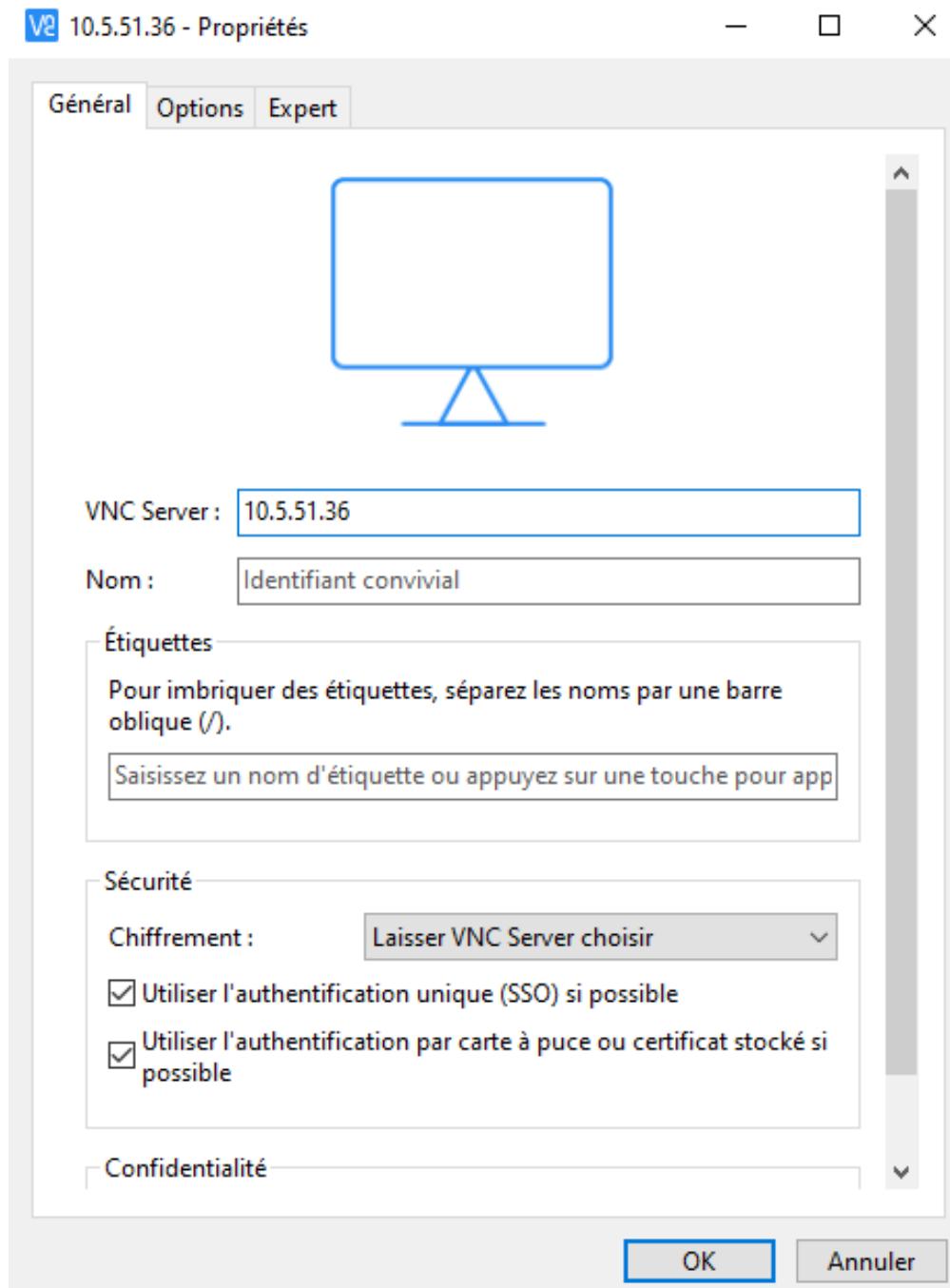
Cliquez ensuite sur **Nouvelle connexion....**



La fenêtre Propriétés devrait s'ouvrir.

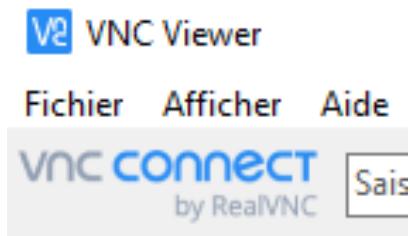
Remplissez le premier champ (**VNC Server**) avec l'adresse IP du Raspberry vu ci-dessus.

Une fois l'adresse entrée, appuyez sur le bouton **OK** en bas à droite.



Une connexion avec l'adresse IP que vous avez rentrée est censée apparaître sur le menu de VNC.

Faites un double-clic dessus.



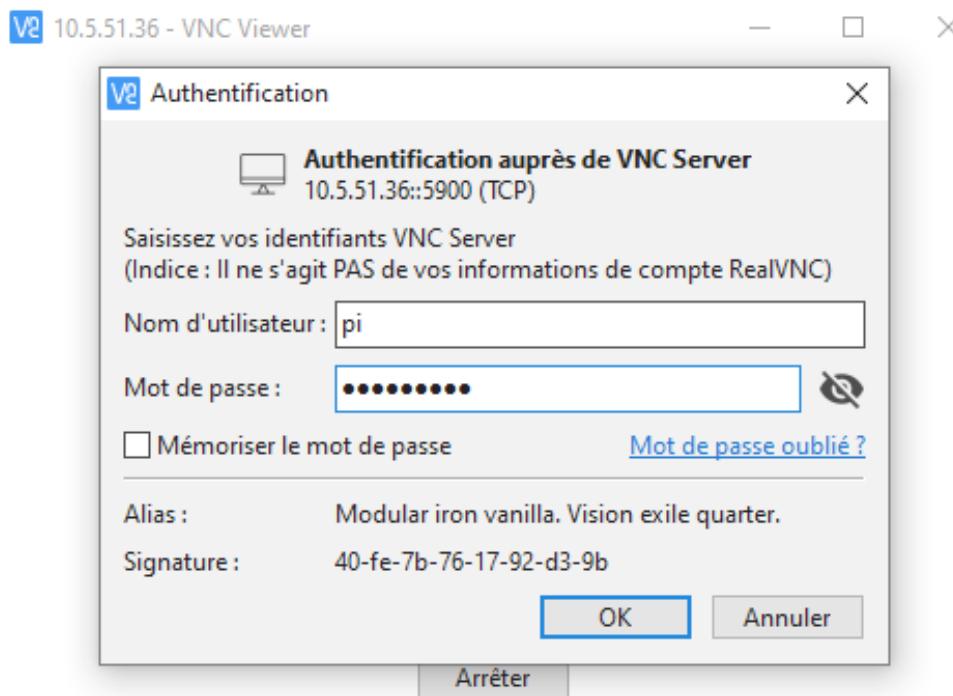
**10.5.51.36**

Il faut désormais remplir le nom d'utilisateur ainsi que le mot de passe.

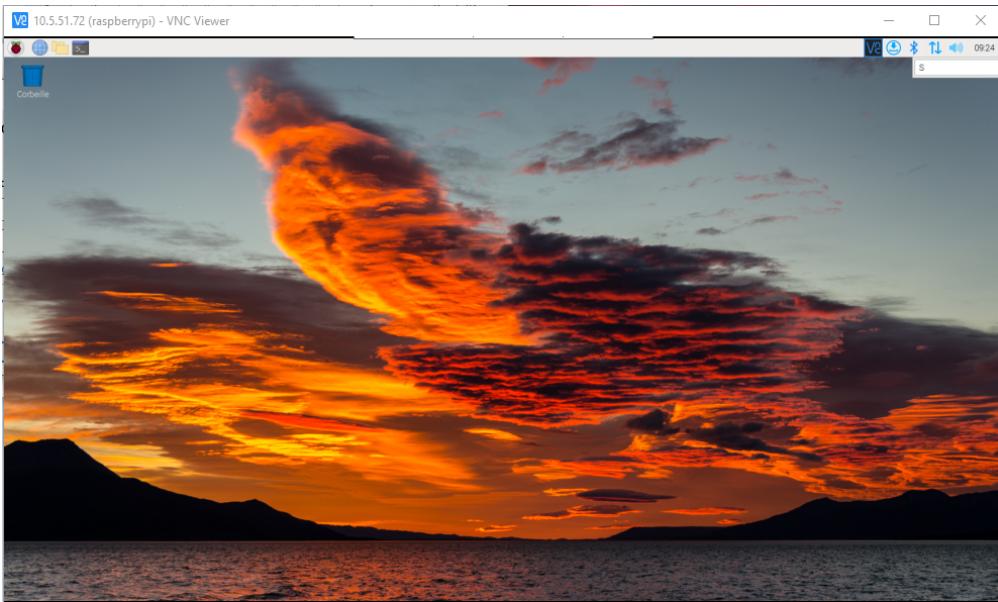
Par défaut, le nom d'utilisateur est pi, et le mot de passe raspberry.

Si vous allez vous connecter régulièrement au raspberry je conseille de cocher la case **Mémoriser le mot de passe**.

Une fois les champs remplis appuyez sur **OK**.



Vous voilà connecté au raspberry pi!



## 6.2.5 Raspap

Afin d'installer Raspap sur le raspberry, commencez par être sûr que votre raspberry est à jour.

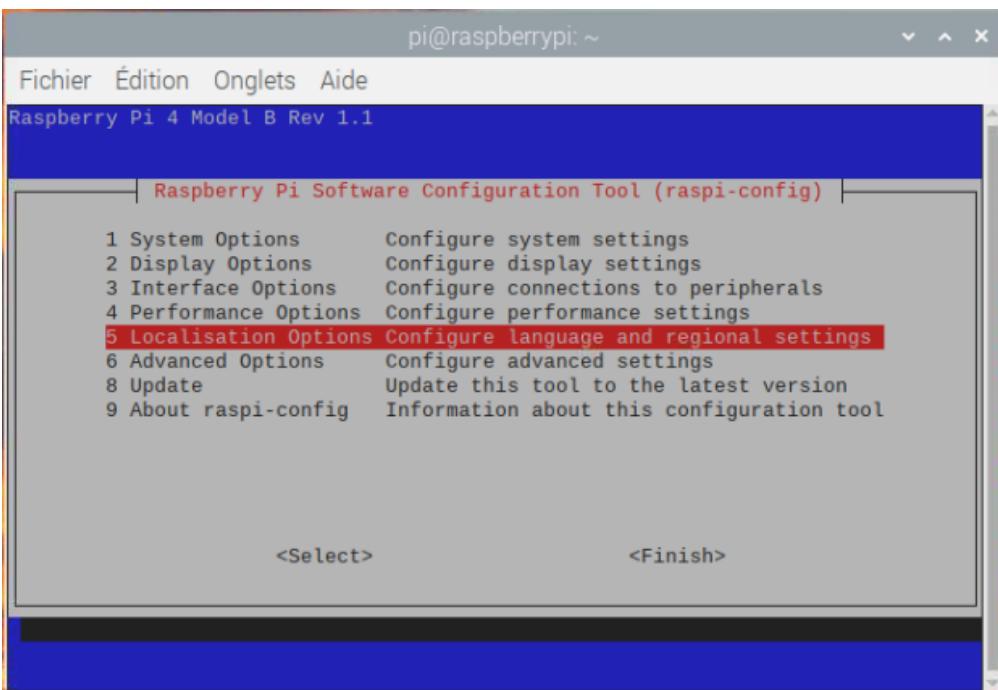
Si votre raspberry 's'arrête' pendant l'installation des packets, lisez la dernière ligne et appuyez sur enter (utilise la valeur par défaut).

```
sudo apt-get update
sudo apt-get full-upgrade
# !Redémarre le raspberry!
sudo reboot
```

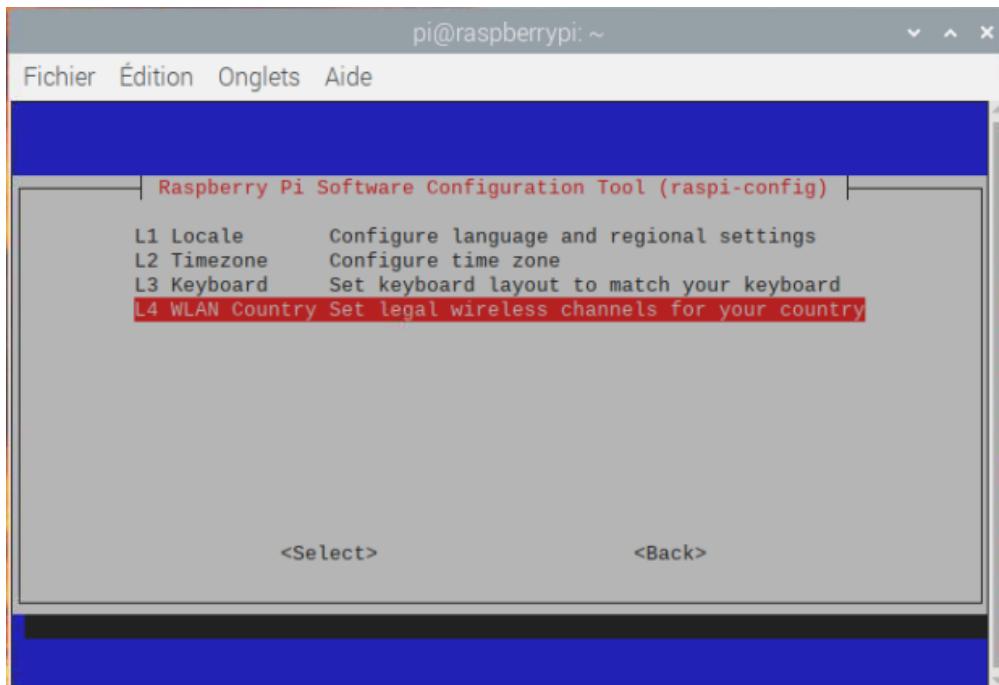
Une fois le raspberry redémarré, utilisez cette commande pour mettre la bonne localisation de votre raspberry.

```
sudo raspi-config
```

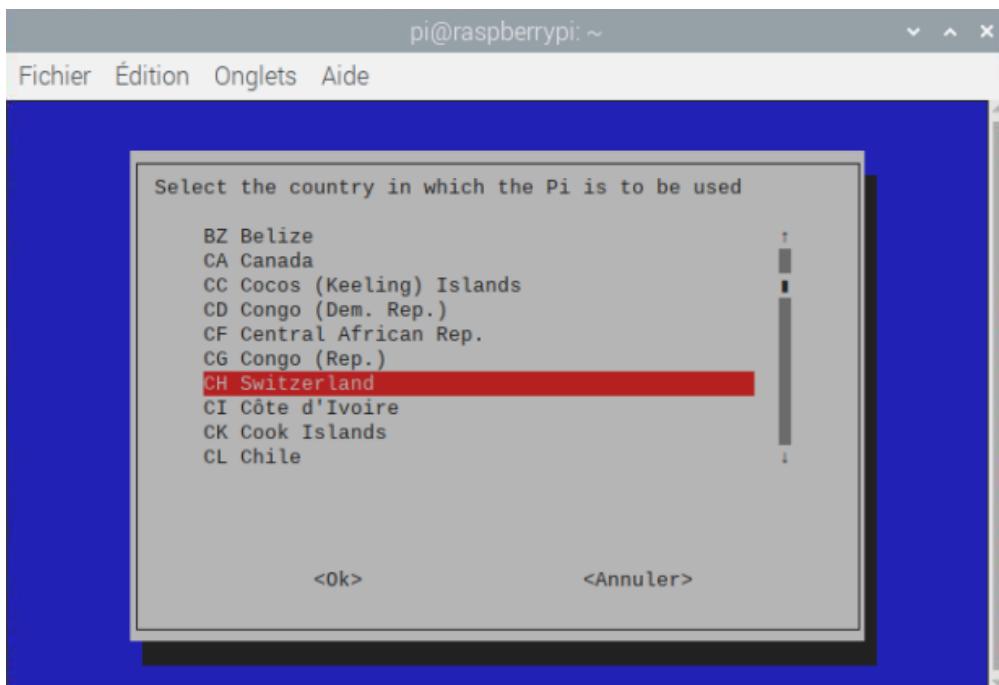
Une fenêtre s'ouvrira, avec les flèches directionnelles sélectionnez le 5e point Localisation Options.



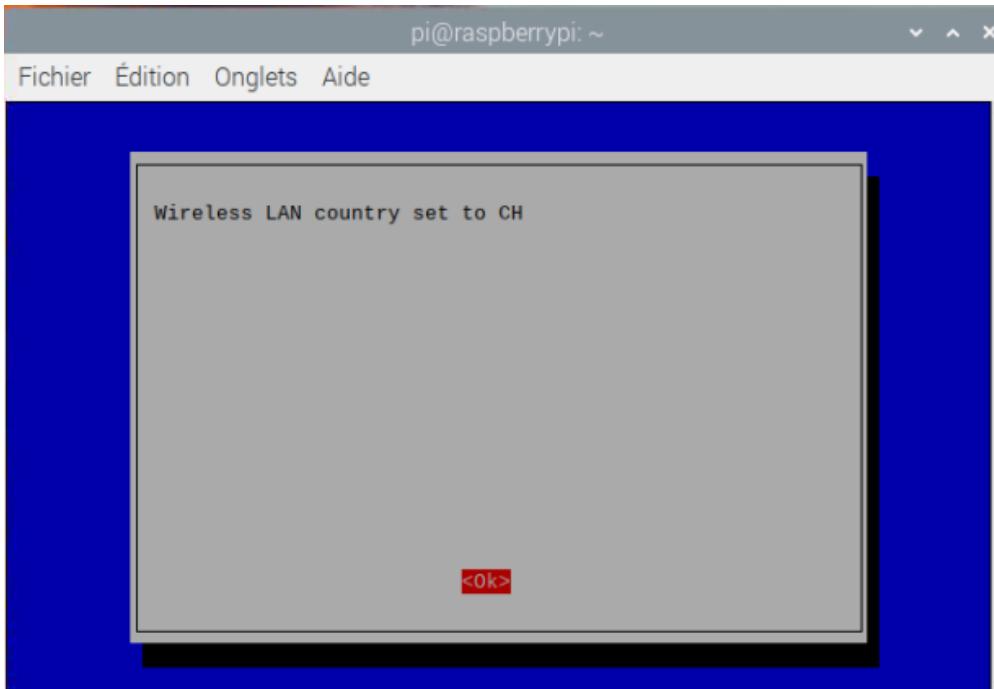
Ensuite, sélectionnez la dernière option WLAN Country.



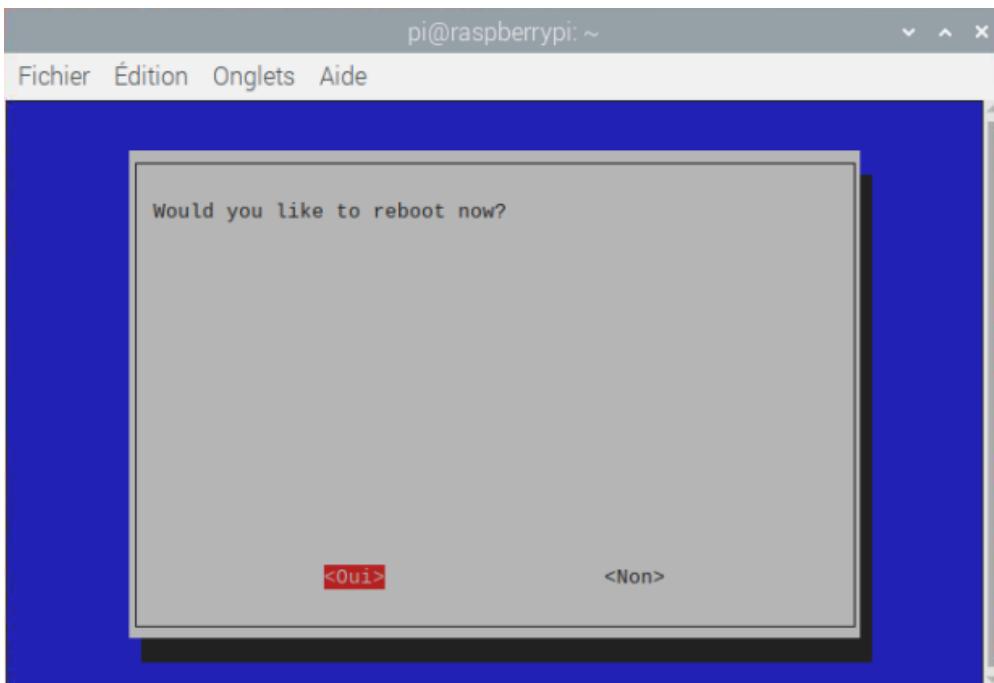
Veuillez sélectionner dans le pays auquel vous vous trouvez. (En Suisse pour ma part)



Maintenant, cette fenêtre n'est qu'une validation du pays, appuyez sur **Ok**.



Ensuite, acceptez le redémarrage, ainsi les modifications seront comptabilisées. Appuyez sur **Oui**.



Dès que le raspberry pi a redémarré, ouvrez un terminal de commande et lancez cette commande:

```
curl -sL https://install.raspap.com | bash
```

De temps en temps, la console vous demandera si vous voulez activer certaines fonctionnalités de raspap.

```
Installing lighttpd directory: /var/www/html? [Y/n]
```

Par défaut, la valeur est **Y**, et je conseille de la laisser ainsi.

sur toutes les questions, appuyez sur **Y**, jusqu'à ce que vous arriviez à ce message:

```
The system needs to be rebooted as a final step, reboot now? [Y/n]
```

## ⚠️ Attention ⚠️

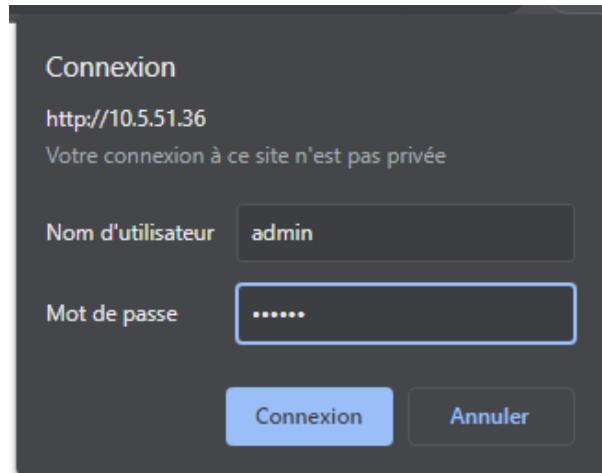
Si vous appuyez sur **Y**, votre raspberry pi redémarrera. Pensez à bien avoir sauvegardé ce que vous faisiez avant de redémarrer.

Si tout est sauvegardé, appuyez sur **Y**.

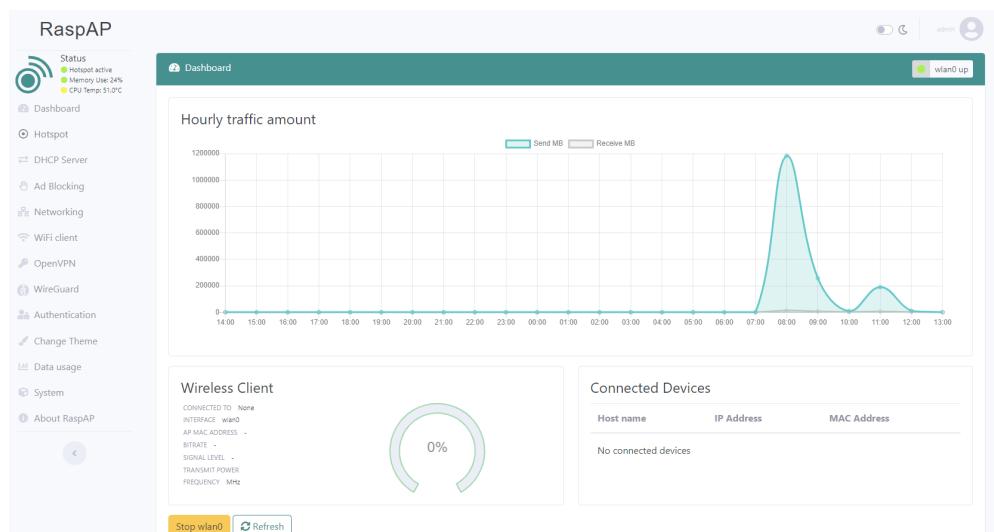
Une fois le raspberry redémarrer, vous pouvez vous connecter avec votre téléphone sur l'access-point que vous venez de créer.

Vous pouvez aussi accéder à la configuration du réseau en allant sur votre navigateur et en entrant l'adresse IP du raspberry (ifconfig). Par défaut, les identifiants de connexion sont:

| Nom               | Valeur |
|-------------------|--------|
| Nom d'utilisateur | admin  |
| Mot de passe      | secret |



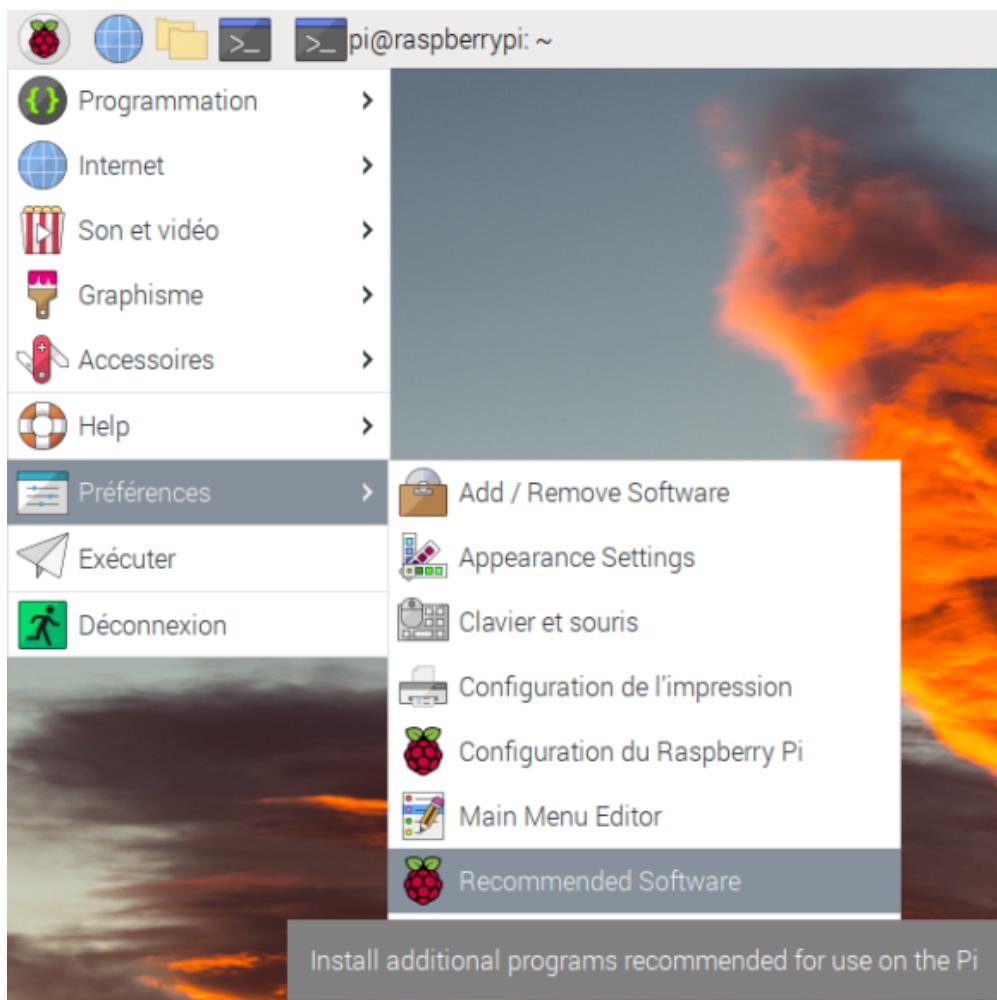
Vous accéderez à cette page. Je vous laisserais ici gérer votre réseau.



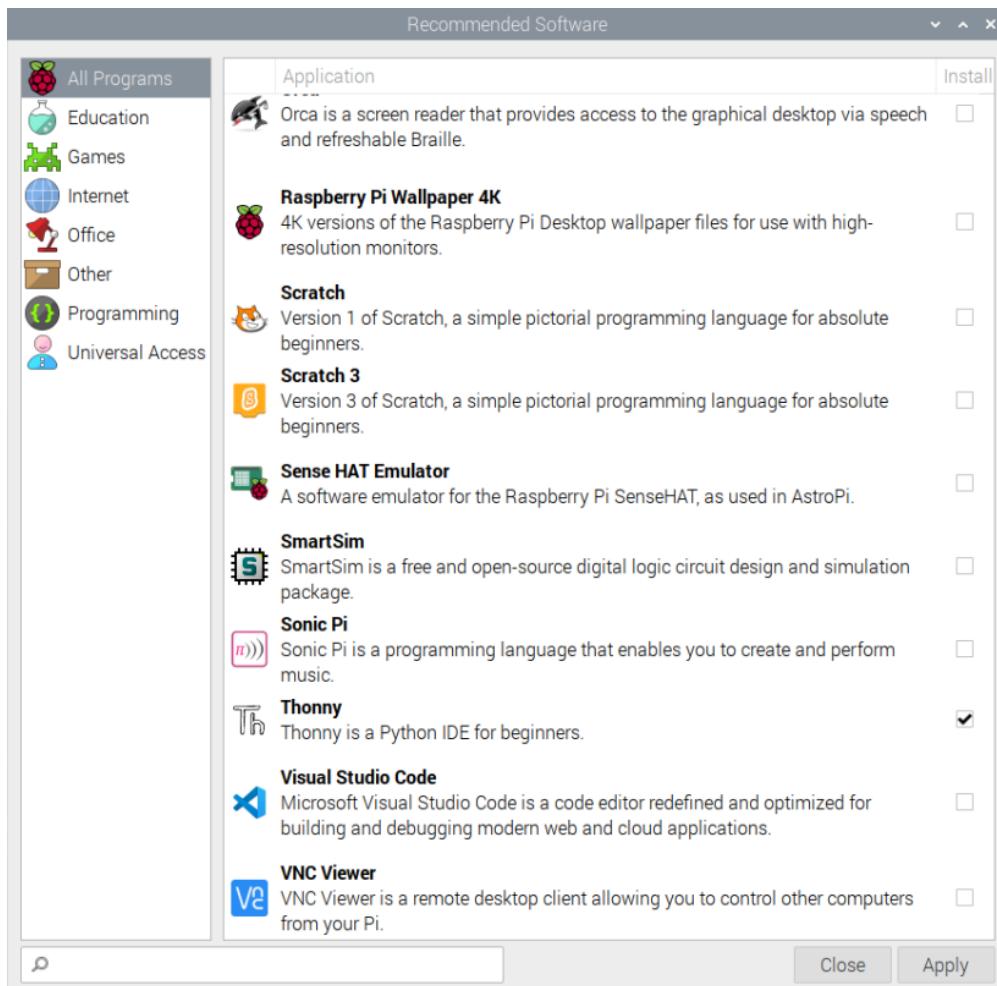
## 6.2.6 VS Code

Voici comment faire pour avoir l'éditeur VS Code pour raspberry pi.

Allez en premier dans les Préférences -> Recommended Software.

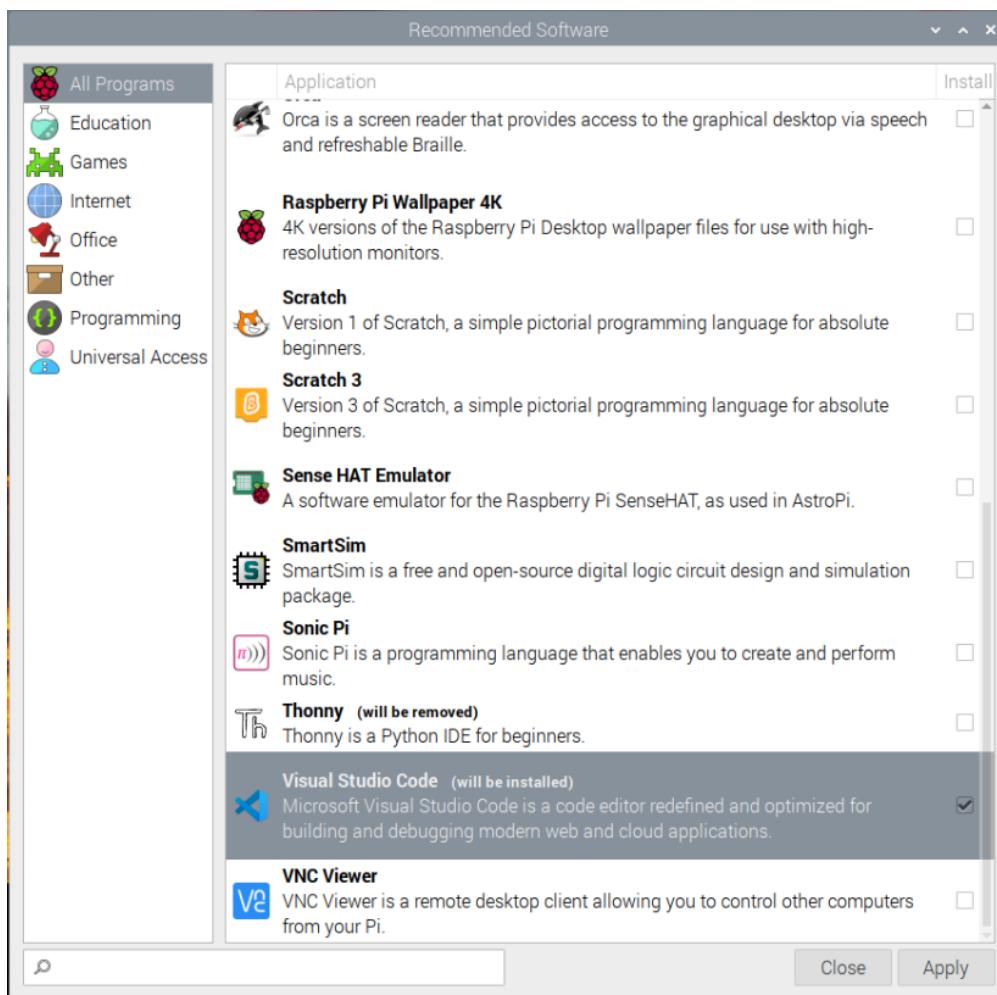


La fenêtre **Recommended Software** va s'ouvrir, allez tout en bas de l'ascenseur.

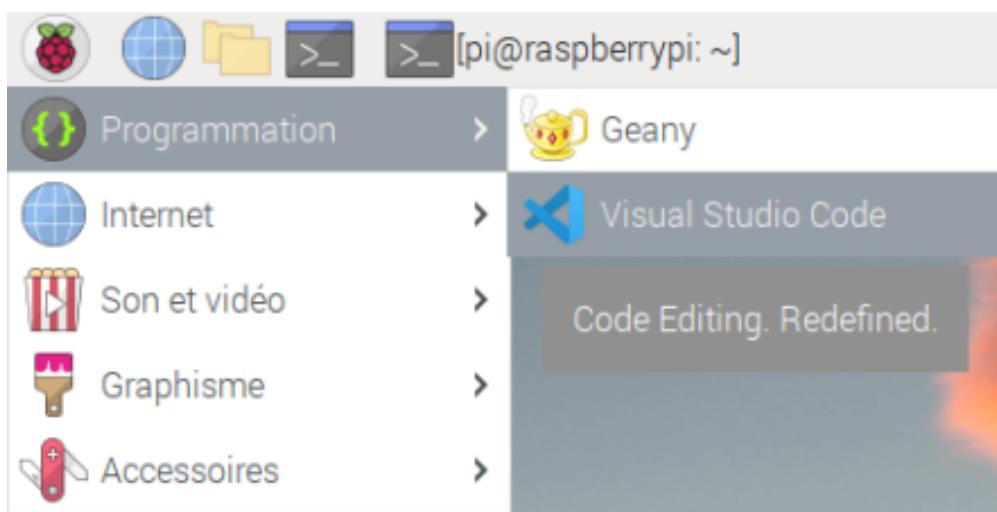


Cochez **Visual Studio Code** pour l'activer.

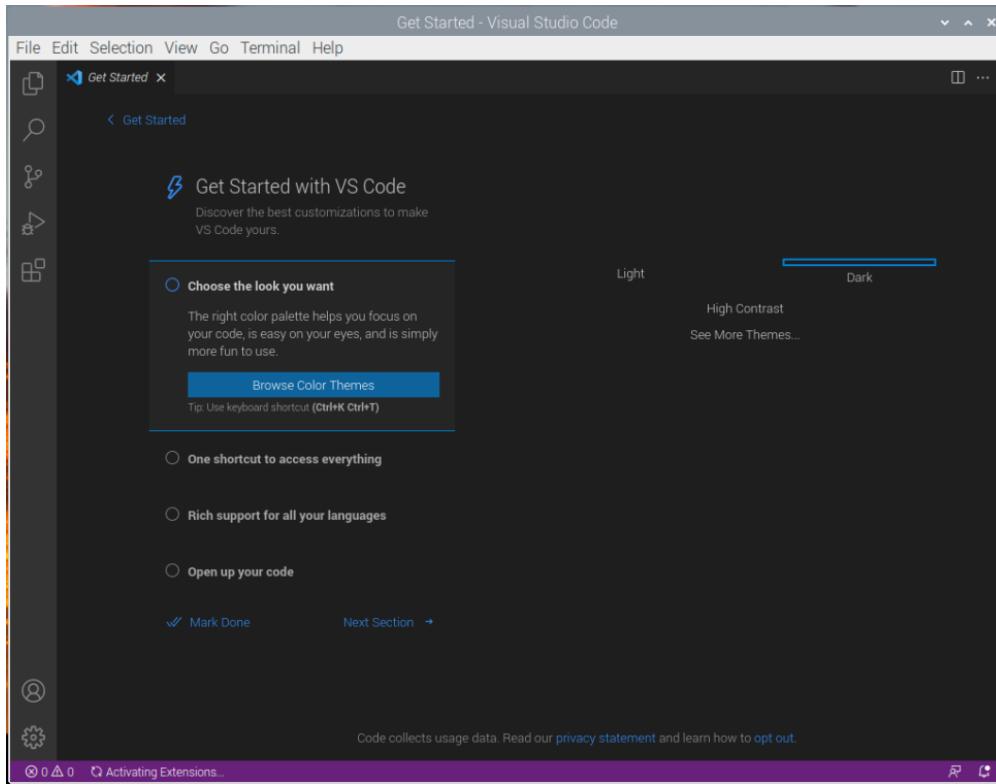
Une fois que c'est fait, appuyez sur **Apply**.



Normalement, une fois l'installation terminée, l'application devrait apparaître dans le menu Programmation -> Visual Studio Code



Au premier démarrage, l'application va vous proposer plusieurs réglages, à vous de les choisir selon vos préférences.



## 6.2.7 Librairies nécessaires

Il vous faut encore télécharger et installer diverses librairies nécessaires au bon fonctionnement de l'application.

```
# OpenCV, cette installation peut prendre jusqu'à plusieurs heures
pip3 install --upgrade pip
pip3 install -U numpy
pip3 install opencv-contrib-python
# Si il y a des erreurs, installer aussi ces librairies (ça peut corriger la plus grande partie des problèmes)
sudo apt-get install python3-opencv
sudo apt-get install libhdf5-dev
sudo apt-get install libhdf5-serial-dev
sudo apt-get install libatlas-base-dev
sudo apt-get install libjasper-dev
sudo apt-get install libqtgui4
sudo apt-get install libqt4-test

# Flask
pip3 install Flask
# Turbo-Flask
pip3 install Turbo-Flask

# Lidar
pip3 install adafruit-rplidar

# Matplotlib
python -m pip install -U matplotlib

# PID
pip3 install simple-pid

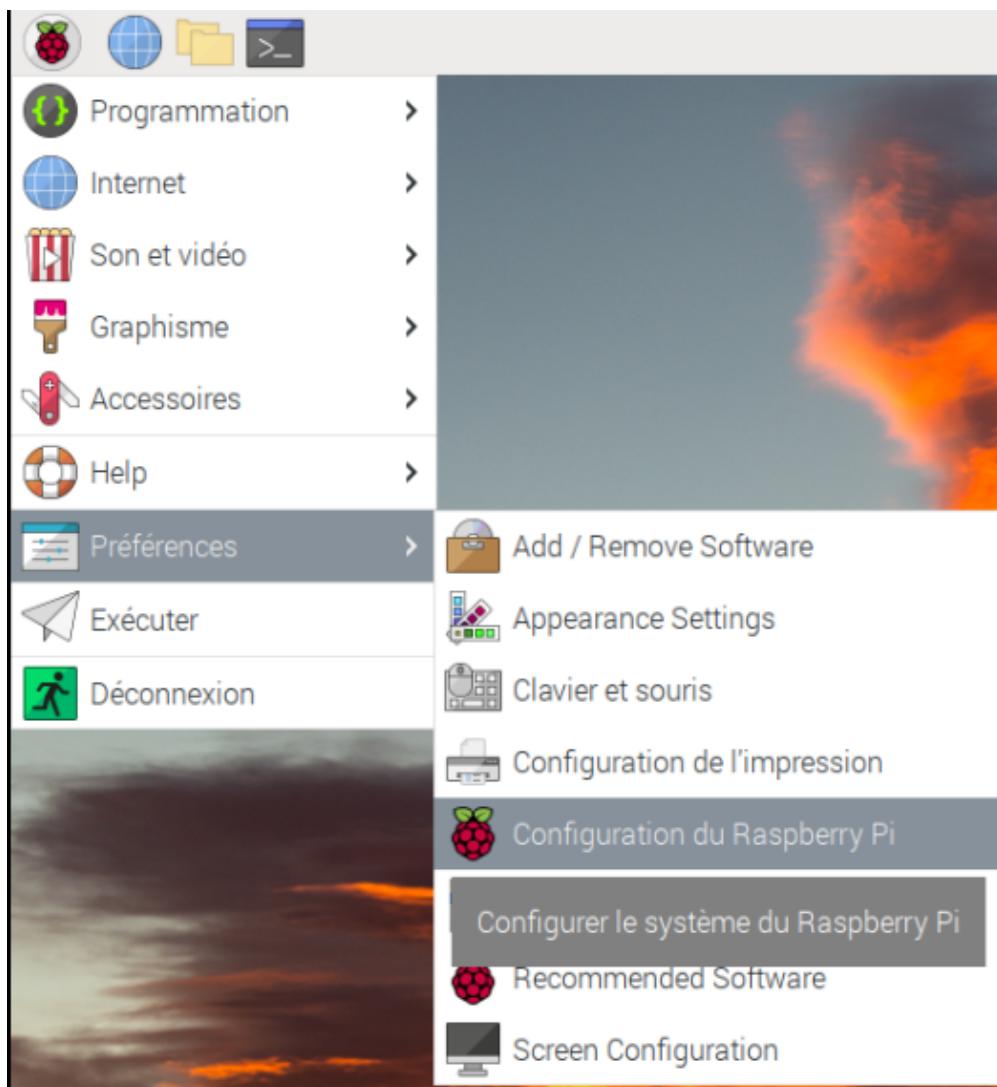
# PCA9685
pip3 install adafruit-circuitpython-pca9685

# MPU6050
pip3 install mpu6050-raspberrypi

# Servomoteur
pip3 install adafruit-circuitpython-motor
```

## 6.2.8 Activation des interfaces nécessaires

Pour commencer, accéder à Préférences -> Configuration du Raspberry Pi.



| Nom            | État     |
|----------------|----------|
| SSH            | Enabled  |
| VNC            | Enabled  |
| SPI            | Enabled  |
| I2C            | Enabled  |
| Serial Port    | Enabled  |
| Serial Console | Disabled |
| 1-Wire         | Disabled |
| remote GPIO    | Enabled  |

Puis ouvrez un terminal de commande et tapez:

```
sudo raspi-config
```

Allez dans Interface options.

Puis, activer la caméra.

Ensuite, accepter le redémarrage du raspberry pi.

Et finalement, dès que le raspberry pi aura redémarré. Relancez un terminal de commande et tapez:

```
git clone https://github.com/NickVanMarkes/Hexapode.git
```

## 6.3 Journal de bord

---

### 6.3.1 Avril

#### 04.04.2022

Aujourd'hui, nous avons eu une petite introduction faite par M. Bonvin et M. Garcia sur le déroulement général du travail de diplôme. Ensuite, j'ai commencé à préparer mon environnement de travail, faire un planning prévisionnel (chose dont je ne suis pas très bon, car je n'arrive pas souvent à préparer ce que j'ai à faire en avance.), un Trello avec mes tâches puis finalement faire le git afin que mon professeur, qui me suit durant le travail de diplôme (M. Bonvin), puisse suivre mon avancée. Durant cette première journée, on peut donc dire que j'ai juste tout bien préparé afin de commencer sérieusement le TD. À la fin de la journée, nous avons eu une visite à l'HEPIA.

#### 05.04.2022

Ce matin, j'ai commencé par essayer de régler le souci que j'avais déjà sur l'interface Web, qui était que je prenais trop de temps pour afficher la vue radar. Ce qui m'a pris toute la matinée en essayant moult méthodes différentes, et finalement, je n'ai pas réussi à finir l'optimisation, car M. Bonvin est venu me parler de la mécanique du robot. Puis, j'ai donc commencé à visser les parties que je pouvais du robot. Enfin, à la fin de la journée, M. Bonvin est venu pour me parler d'une technique afin de savoir où se situent les pattes, grâce à l'accéléromètre et grâce au gyroscope. La technique consiste à mettre le robot sur une surface plate, lever les pattes, et de prendre ce point comme le point 0, et ainsi tester les moteurs les uns après les autres, et ainsi, on peut savoir quand ils touchent le sol et déterminer leurs positions à tout moment. Durant son explication, M. Bonvin a remarqué que les hanches du robot, n'étaient pas solidement attachées au reste du robot. On a commencé à penser pour résoudre ce souci, et l'idée la plus prometteuse est de prendre un servomoteur en métal afin qu'il y ait le moins de jeu possible.

#### 06.04.2022

Aujourd'hui, j'ai passé ma journée à essayer de résoudre le problème avec l'interface Web. Malheureusement, je ne comprends pas pourquoi le plot, pour faire la vue radar, ne veut pas fonctionner. J'ai regardé dans l'inspecteur du navigateur et je voyais que les images prenaient beaucoup trop de temps à être téléchargées (environ 2 à 5 secondes) et je ne sais pas s'il faut compresser les images pour que ça s'envoie plus vite, car au niveau du raspberry pi il n'est pas surchargé.

#### 07.04.2022

Ce matin, je continue à essayer de résoudre le problème de la vue radar, sauf que rien ne marche, les images prennent encore trop de temps à arriver au navigateur. J'ai donc décidé de laisser ça de côté et attendre que M. Bonvin puisse m'aider sur ce problème, et j'ai commencé à afficher l'information de la distance de l'obstacle le plus proche sur la bannière sur le haut de l'interface. À midi, M. Bonvin est passé dans la classe et m'a ramené des caisses remplies de composants de chez lui, ainsi que les connecteurs afin de faire les rallonges pour les servos. Ensuite, il m'a expliqué en très rapide ce qu'il fallait faire pour résoudre mon souci: - Faire une classe externe qui gère la création des plots - On s'en fiche du Web et on fait en sorte que ça créer les plots hors Web. - Dès que tout fonctionne, intégrer cette classe dans le Web - Faire une documentation de cette classe et expliquer tout ce que je fais au mieux

En regardant mon trello, je trouve que je passe beaucoup trop de temps sur ce problème avec les plots. Il faut donc que je le règle au plus vite. Je pense refaire comme pour le travail de semestre et travailler le soir chez moi pour faire en sorte que ça marche. Après les cours, j'ai fait le programme que m'avait dit M. Bonvin, et j'ai remarqué que le message «Start» du lidar prenait trop de temps à venir, et j'ai vérifié en mettant plein de timers et matplotlib prends 5.51 secondes à être importé alors que les autres 5 mS. Il faudrait donc que je m'informe envers M. Bonvin sur ce qu'il prend autant de temps et pourquoi.

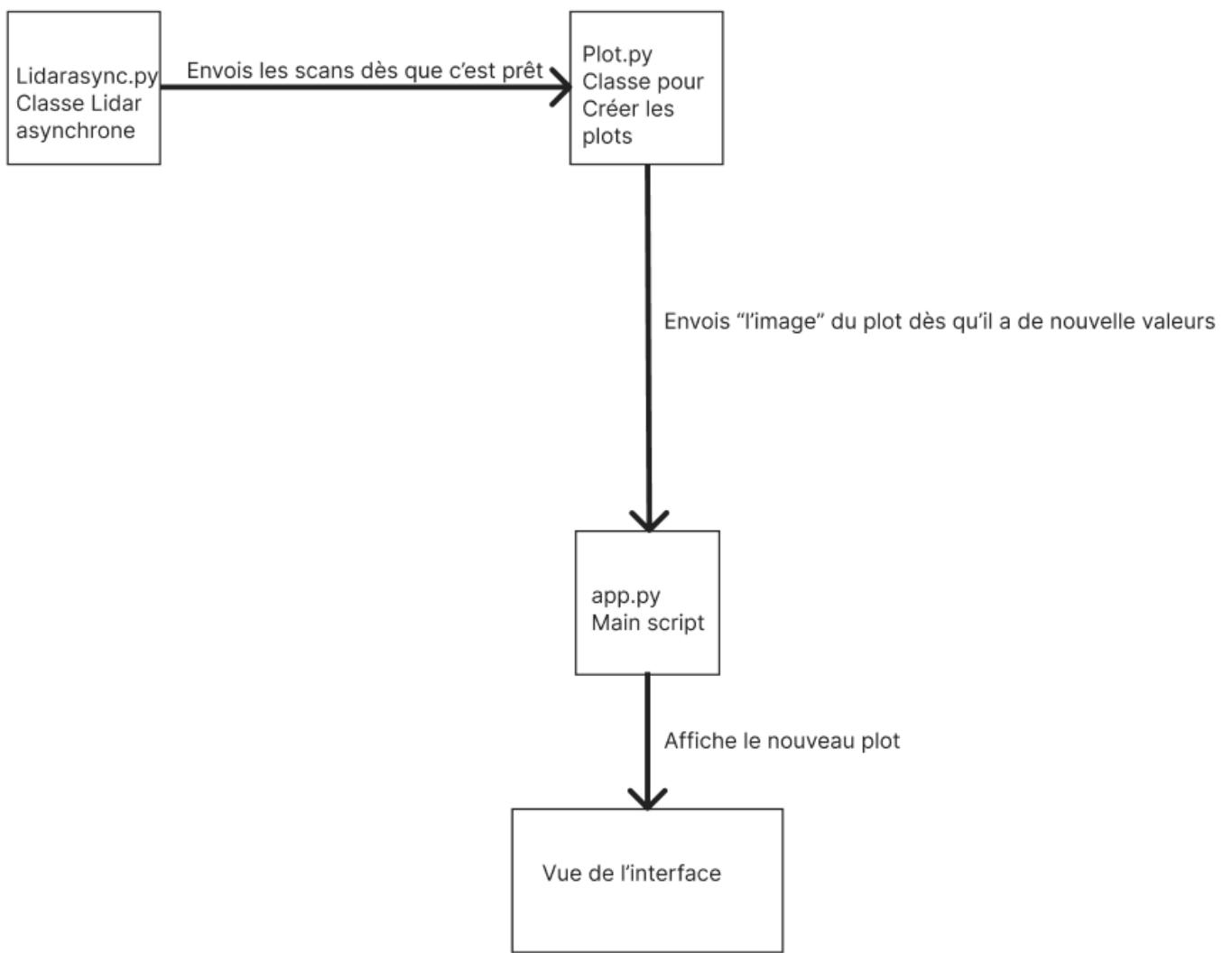
#### 08.04.2022

Ce matin, j'ai continué à travailler sur le débogage de la vue radar, et M. Bonvin m'a appelé en me disant de faire un autre script, et m'a dit de prendre le code d'exemple. Puis, ne voyant toujours pas le bout de la solution à ce problème, j'ai commencé à faire le câblage pour les rallonges des servos moteurs, car aujourd'hui, j'ai reçu la commande avec les gaines thermorétractables. Ensuite, je les ai essayées, en regardant dans les pires positions voir si tout était assez long, et tout est comme il faut. Puis, vers

la fin de la journée, je me suis réattaqué sur la solution de la vue radar. En testant, j'ai vu que même avec l'exemple cela prenait 30 secondes pour avec 400 scans. Mais en regardant ce qu'il y avait dans les 400 scans, il y avait  $400 \times 360^\circ$ , ce qui est problématique. J'ai essayé alors avec 1 seul scan, il y manque beaucoup d'information, car il prend que de là où il est en  $^\circ$  jusqu'à  $\sim 360^\circ$ . De ce fait, pour être sûr d'avoir toutes les infos, 2 scans sont nécessaires, ce qui prend 0.46 seconde.

#### 11.04.2022

Aujourd'hui, j'ai continué à travailler sur le programme pour le lidar. Ensuite, pendant la matinée, M. Bonvin m'envoie un mail, en me disant d'utiliser une autre librairie pour communiquer avec le lidar, et de le faire marcher parfaitement. Au début, j'avais plein de problèmes, car le baudrate n'était pas le bon (il avait spécifié dans son mail, d'essayer de prendre le baudrate le plus rapide.), alors je suis allé voir dans la documentation qu'il m'avait envoyée aussi, et elle n'explique pas grand-chose, elle ne fait qu'une brève présentation des fonctions, sans vraiment dire comment elle fait. Mais dans l'après-midi, j'ai réussi à régler le souci. Ensuite, j'ai essayé d'en faire une classe et de faire une version asynchrone, car je pense que ça serait très utile que le lidar soit en asynchrone, afin que dès qu'il a de nouvelles données, il envoie à la classe matplotlib, et celle-ci créer le plot et l'envoi sur l'interface. Comme représenté dans le schéma ci-dessous.



Et enfin, à la fin de la journée, j'ai décidé d'avancer le câblage du robot. Avec du veroboard, j'ai soudé tous les câbles utiles pour l'I2C (SDA et SCL). Et demain, je vais faire la même chose, mais pour les alimentations des modules, ainsi, je pourrais tester les modules des servos moteurs et commencer à faire bouger le robot.

**12.04.2022**

Aujourd'hui, j'ai commencé la journée et continuant de transformer ma classe Lidar en asynchrone. Puis avec certains problèmes, j'ai décidé d'avancer le câblage et de le finir, car ainsi, je pourrais commencer à travailler sur les servos. Dès le début de l'après-midi, j'avais fini avec le câblage, et j'ai commencé à tester les servomoteurs. Ils marchent tous correctement. Mais, habitué à n'avoir que des servos non continus, j'ai un peu peiné au début, pour les faire bouger. Ils se tapaient sur tout, car ils allaient toujours de la min au max, ou du max au min, mais jamais le degré que je voulais. Ensuite, je suis allé voir la documentation de la librairie adafruit, et ils expliquent implicitement comment ils fonctionnent. J'ai calculé, ensuite, le temps que prennent les servos pour faire un degré:

En 1 min le servo fait 50 tours, il faut donc savoir combien de temps pour 1 tour. Une simple règle de 3 pour le trouver, ce qui nous donne 1.2 seconde pour un tour. Ensuite, il faut juste diviser cette 1.2 seconde par 360 pour savoir le temps pour faire 1 degré, ce qui nous donne 0.003 seconde. Puis, j'ai préparé un code conséquent et faire certains tests avec les servos. Et à la fin de la journée, j'ai demandé conseil à M. Bonvin, et il me conseilla sur ce que je dois faire.

Je pense qu'en Off chez moi, j'imprimerais certaines pièces de la façon dont il m'a expliqué, afin de réduire le poids des pièces.

**13.04.2022**

Aujourd'hui, j'ai commencé ma journée en faisant des tests sur le module gyroscope/accéléromètre. J'ai eu quelques soucis pour trouver la bonne adresse, j'ai dû chercher dans la documentation technique, mais il n'y avait pas pour le module. Ce que j'ai fait, c'est que je suis allé chercher dans tous mes projets d'électronique, et j'ai réussi trouvé dans un dossier caché au milieu de tous mes dossiers, un ancien programme qui utilisait ce module et I2C et j'ai réussi à trouver l'adresse. J'ai testé voir s'il marchait correctement, et tout m'as l'air assez juste. Ensuite, durant l'après-midi, j'ai continué à travailler sur les servomoteurs. En branchant 2 pattes, je voyais qu'il arrivait déjà à soulever, mais légèrement, et les servos du milieu chauffent pas mal, alors j'ai testé avec une troisième patte, et là, tout le corps du robot, c'est soulevé avec aisance. Cela m'a fait très plaisir, car je me dis que finalement (sans le poids des batteries et du lidar) il arrive à soulever le robot! Je pense donc essayer durant toutes les vacances à faire des pseudoanimations, ou du moins réussir à le soulever complètement.

**18.04.2022**

Aujourd'hui, j'ai avancé la documentation technique avec ce que j'avais déjà, ainsi qu'expliqué les différents protocoles de communication que j'utilise.

**25.04.2022**

Aujourd'hui, j'ai continué à essayer de tester les servos, surtout voir quelle force minimale j'ai besoin pour qu'elles lèvent la patte. Et ensuite, dès que j'ai trouvé cette information (car ça m'a pris un bon bout de temps de faire sur chaque patte), j'ai pris une feuille, mon programme de test des servos où je peux mettre leurs angles, et faire les animations étape par étape. Mais malheureusement, vu que je n'ai pas encore reçu le reste de mes pattes, je ne peux pas finir correctement les animations, je suis donc bloqué sur cette étape, et heureusement que ce problème est arrivé en fin de journée sinon j'aurais vite perdu une journée de travail.

**26.04.2022**

Ce matin, M. Bonvin est venu nous donner les feuilles afin qu'on se fasse une auto-évaluation par rapport à notre façon de travailler sur le projet et autres. Puis, M. Garcia est venu aussi pour ces élèves et il m'a dit de relancer M. Muccioli pour les pièces 3D, chose que j'ai faite, et heureusement qu'on l'a fait, car il attendait une confirmation sur son e-mail edu.ge.ch, j'ai donc rédigé un mail de confirmation, avec le peu d'information dont je me souviens. Ensuite, il m'a très vite répondu, en me disant que demain elles seraient prêtes (28 heures d'impression), de ce fait demain à la grande pause, j'irais les chercher. Mais ça me ralentit pour mes tests, je ne vais donc pas rester en train de ne rien faire, je pense que le reste de la matinée, je vais faire en sorte que les câbles sur le robot, soient bien rangés, enfin avoir un bon câble management, afin de pouvoir s'y retrouver. Au début de l'après-midi, j'ai branché et testé les servos du côté droit du robot. Ensuite, j'ai avancé la documentation, afin d'expliquer les servomoteurs, comment fonctionnent-ils, et ce que j'en fais. Puis M.Bonvin est venu pour me rendre son bilan sur les deux premières semaines, ce qui n'a pas été concluant pour lui, et dans la discussion nous avons décidé que le vendredi, je dois rendre la classe Lidar asynchrone fonctionnelle.

**27.04.2022**

Ce matin, j'ai commencé à travailler sur la classe du lidar en asynchrone. J'ai plein de problèmes et d'incompréhension par rapport à l'asynchrone, j'essaye plein de façons différentes, mais rien ne marche, il y a toujours une erreur, ou une mauvaise valeur. Ensuite, je suis allé chercher à la grande pause, les nouvelles pièces imprimées à butin. Puis jusqu'à la fin de la matinée, je les ai montées, puis refais le câble management. Au début d'après-midi, j'ai recommencé à travailler sur l'asynchronisation de la classe lidar, et je n'y comprends toujours rien, j'essaye encore des façons différentes, et que des erreurs différentes. Puis, M. Aigroz, vers la fin de la journée il est passé dans la classe voir comment ça allait, et il m'a beaucoup aidé avec les mouvements des servos afin de faire bouger le robot. Ensuite, il m'a aussi très bien expliqué ce que je dois faire pour la classe lidar, et de ce que j'ai compris, il faut que je fasse en multithreading et non en asynchrone, car en asynchrone il bloque quand même le code jusqu'à recevoir les données, alors qu'en multithreading, il fait une copie et lance un autre process sur un autre thread pendant que notre programme principal tourne toujours.

**28.04.2022**

Ce Matin, j'ai vu le mail que M. Bonvin m'avait envoyé hier qui m'aidera à mieux comprendre l'asynchrone ou le multithreading. J'ai donc passé les deux premières heures à lire et essayer de comprendre ce cours tout en essayant avec mon programme [lien du cours] (<https://leimao.github.io/blog/Python-Concurrency-High-Level/>). Ensuite, à la grande pause, M. Bonvin est venu afin de nous aider, et m'as expliqué aussi ce dont j'ai besoin de comprendre dans tout ce cours, et que du ce fait, je ne dois qu'utiliser des threads. Puis, un petit remontage de bretelles, car il me demandait des informations basiques que je n'avais pas de tête et je comprends que pour la présentation il faut que je sois vraiment à jour sur cela. Durant l'après-midi, j'ai continué à essayer de faire le multithreadings, sauf qu'il y a encore des problèmes. Je lance un timer qui doit lancer toutes les secondes, la fonction du lidar pour scanner, sauf qu'il ne le fait pas, je me suis dit qu'il devait se bloquer dans une boucle, je regarde en mettant des print un peu partout, mais non, aucun blocage, juste le thread «timer» se lance une fois, et pas toutes les secondes. Je vais essayer de trouver la solution demain sinon j'aurais un problème avec M. Bonvin.

**29.04.2022**

Ce matin, j'ai continué à travailler sur la classe Lidar, et j'ai testé en faisant une fonction qui incrémente un chiffre et pas directement avec le lidar. Ce qui m'a donc permis de voir que le Timer ne passait qu'une fois et pas plus. J'ai alors cherché pourquoi il faisait ça, et de ce que j'ai cru comprendre, le Timer en python ne sert que pour lancer une fonction, mais avec un délai, et pas une répétition avec un intervalle. J'ai cherché comment le faire répéter, et tester et ça marchait bien. Ensuite, dès que le Timer a marché, j'ai eu un souci avec le lidar, plus particulièrement le port. Une erreur me disait que le port n'était pas ouvert, mais dans mon code je disais au lidar qu'après chaque scan qu'il devait se déconnecter, mais je ne le reconnectais pas. En réglant, ce souci tout allait pour le mieux, et je faisais en sorte que tout marche comme il faut. Durant l'après-midi, j'ai fait l'assemblage de l'hexapode sur solidworks, car je vais essayer de faire ce que m'as dit M. Aigroz, qui est de modéliser, faire les mouvements, et grâce aux animations reprendre les valeurs et les retranscrire sur le code, et ainsi si je veux rajouter des animations, je n'ai qu'à les faire sur solidworks et retranscrire les valeurs denouvelles.

**30.04.2022**

Aujourd'hui, j'ai passé ma journée à essayer de faire les animations avec solidworks sauf qu'il y a un problème et je n'arrive pas à le faire bien fonctionner, il y a un composant dans les pièces du servomoteur sont fixés, et afin de faire des animations correctes, il faut que je puise les libérées, mais l'option n'apparaît pas, j'ai cherché sur Internet, mais je ne trouve pas de solution. La solution que j'ai trouvée un peu plus tard, c'est de prendre un autre fichier 3D des servos, et rassembler le robot. Ensuite, il y avait un problème, car quand je fais bouger le Tibia dans l'animation, la pointe ne suit pas comme en vrai.

**6.3.2 Mai****02.05.2022**

Ce matin, j'ai continué à essayer de faire l'animation sur solidworks. J'ai vu qu'en mettant une contrainte en plus je pouvais faire bouger la pointe comme je le voulais, mais après dans la partie de l'animation, ça me faisait plein de problèmes, la contrainte faisait en sorte que dans l'animation plus rien ne bouge, chose que je ne comprends pas vraiment pourquoi, j'ai donc essayé d'activer ou désactiver grâce à l'animation, mais ça ne marchait pas. Ensuite, j'ai essayé de construire le tout sur Unity, mais ce n'était pas possible, car je n'ai pas une connaissance assez grande de Unity pour changer le point 0 d'une pièce. J'ai décidé alors d'arrêter de vouloir faire des animations préparées et de les faire en code directement. Durant l'après-midi, j'ai continué les deux

premières heures à essayer de faire lever le robot en code, sauf que les pattes du robot glissent trop, et je n'arrive pas à les faire lever correctement. Puis, le reste de la journée, j'ai avancé la documentation technique.

#### 03.05.2022

Aujourd'hui, j'ai passé la journée à essayer de faire monter le robot, depuis une position initiale. J'ai dû faire vraiment petit à petit, c'est assez lent pour faire les animations comme ça, mais il faut le faire. Durant l'après-midi, j'avais une version presque réussie de ce mouvement, sauf que vers la fin de l'animation, les servos faisaient des spasmes. J'ai décidé alors de prendre une autre alimentation, et de tester. Je suis assez surpris, car les servos du côté gauche sont très nerveux maintenant, et ceux du côté droit plus lent j'ai l'impression.

#### 04.05.2022

Aujourd'hui, j'ai commencé la journée par tester la technique de M. Bonvin pour lever le robot. Forcer sur les deux pattes avant, ensuite sur les deux pattes arrière, et dès que c'est levé, ajouter les pattes du milieu pour la stabilité. À ma surprise, il a de la peine à se lever. Je pense donc continuer à utiliser la technique des 6 pattes en même temps. Et en parlant avec mes camarades, ils m'ont rappelé qu'il fallait faire le poster, pour que demain M. Bonvin puisse les examiner, comme ça on peut les changer jusqu'à lundi. Faire le poster, m'a pris toute la journée, et demain, j'essayerai de faire l'abstract, comme ça tout ce dont j'ai à rendre sera fait, et je pourrai me concentrer sur le robot.

#### 05.05.2022

Aujourd'hui, au début de la matinée, j'ai continué à travailler sur l'animation d'initialisation du robot. Puis, après la grande pause, M. Bonvin, est venu, pour qu'on lui montre nos posters, et de ce fait, je lui ai montré le mien, et étonnement, il a été assez gentil, et il m'a juste dit que je devais changer quelques détails et rajouter le dessin du lidar sur le poster, ce que j'ai fait jusqu'au début de l'après-midi. Ensuite, j'ai recommencé à travailler sur le robot, et je me suis dit qu'il faudrait utiliser le gyroscope afin de savoir quand le robot est droit. En testant, j'ai eu un très joli résultat:





Et ce résultat sans le gyroscope. J'ai implémenté le gyroscope après ce résultat-là.

**06.05.2022**

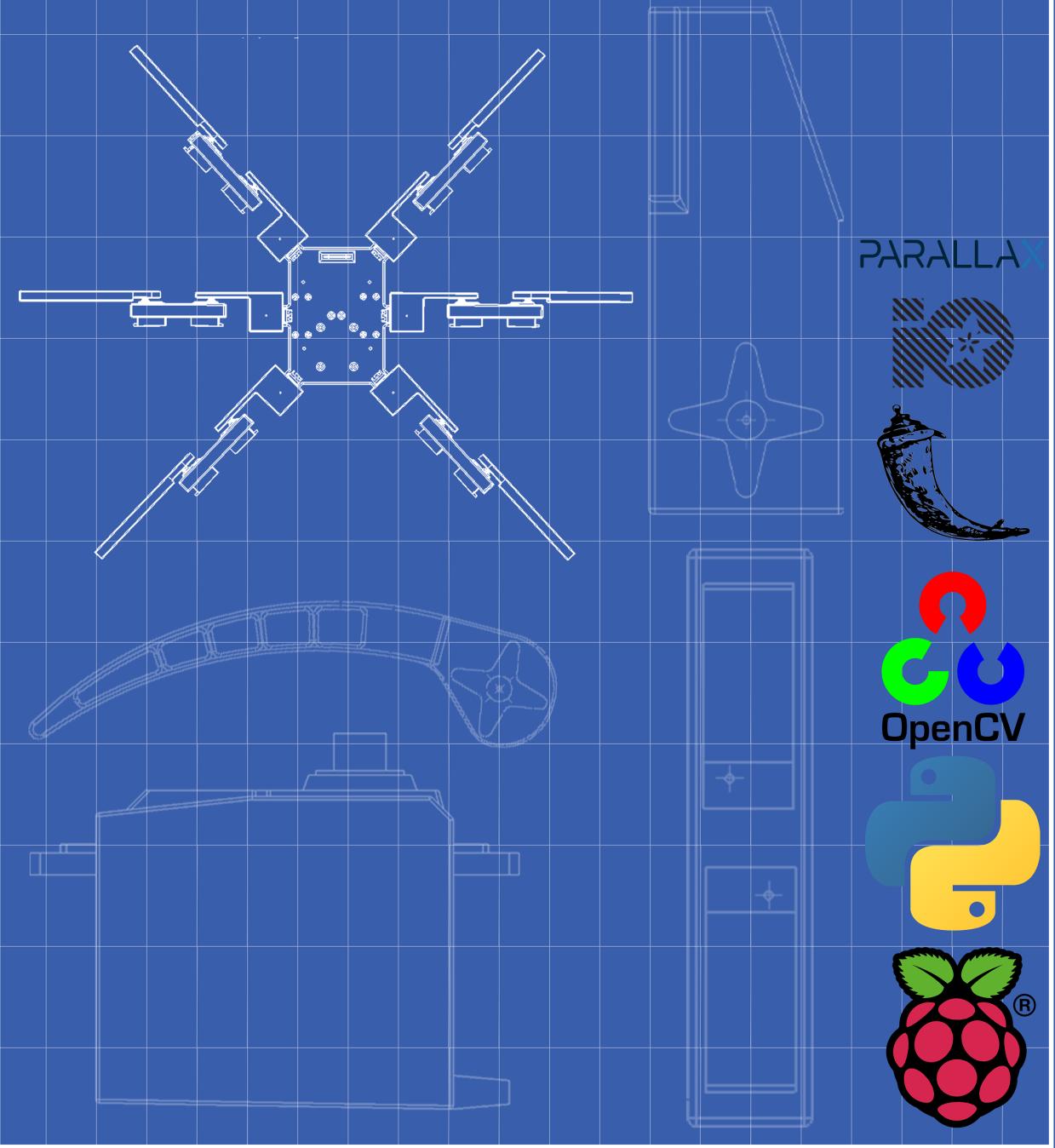
Ce matin, j'ai continué à travailler sur l'implémentation du gyroscope sur le mouvement d'initialisation, sauf que le gyroscope me rend des valeurs, on dirait de l'aléatoire. Je peux passer de -3° à 20°. Puis, j'ai fait quand même quelques essais, et ils étaient assez concluants, le robot était droit. Mais plus j'essayais, moins ça marchait. Je pense que lundi, il faut que j'implémente une façon pour ne pas forcer sur tous les servomoteurs en même temps, et les faire forcer 2 par 2 ou patte par patte, cela très rapidement, ainsi, je consommerais moins. Même si les servos veulent consommer plus, ils auront de la marge. Cette après-midi, j'ai continué à travailler sur les mouvements, et j'ai essayé de faire avancer déjà une patte, mais le robot ne se lève pas assez haut. Je dois donc le faire monter assez pour que la patte puisse bouger aisément. Puis, comme j'ai dit ce matin, plus je force sur les servos en un temps court, plus j'ai l'impression qu'ils ont de la peine à forcer. Ce weekend, je vais essayer d'avancer un maximum la documentation, car lundi nous devons rendre le poster et la documentation ainsi que le résumé/abstract. J'ai déjà fait le poster. Je me suis inspiré du blueprint d'Iron Man. Mon poster ressemble à un dessin technique avec toutes les pièces en 3D que j'ai réalisé, avec aussi les logos des librairies/composants les plus importants.

V1 du poster:

# THE BIG BUG

## ROBOT HEXAPODE :

- CONTRÔLÉ PAR SMARTPHONE
- DÉTECTE LES OBSTACLES
- CAMÉRA EMBARQUÉE



Il y a le robot en 2D avec les pattes tendues, mais quand M. Bonvin a vu, il m'a conseillé de plutôt mettre une vue en 3D en position. Voici donc la V2:

# THE BIG BUG

## ROBOT HEXAPODE :

- CONTRÔLÉ PAR SMARTPHONE
- DÉTECTE LES OBSTACLES
- CONCEPTION ORIGINALE
- SUIVRE DES PERSONNES



SLAMTEC



PARALLAX



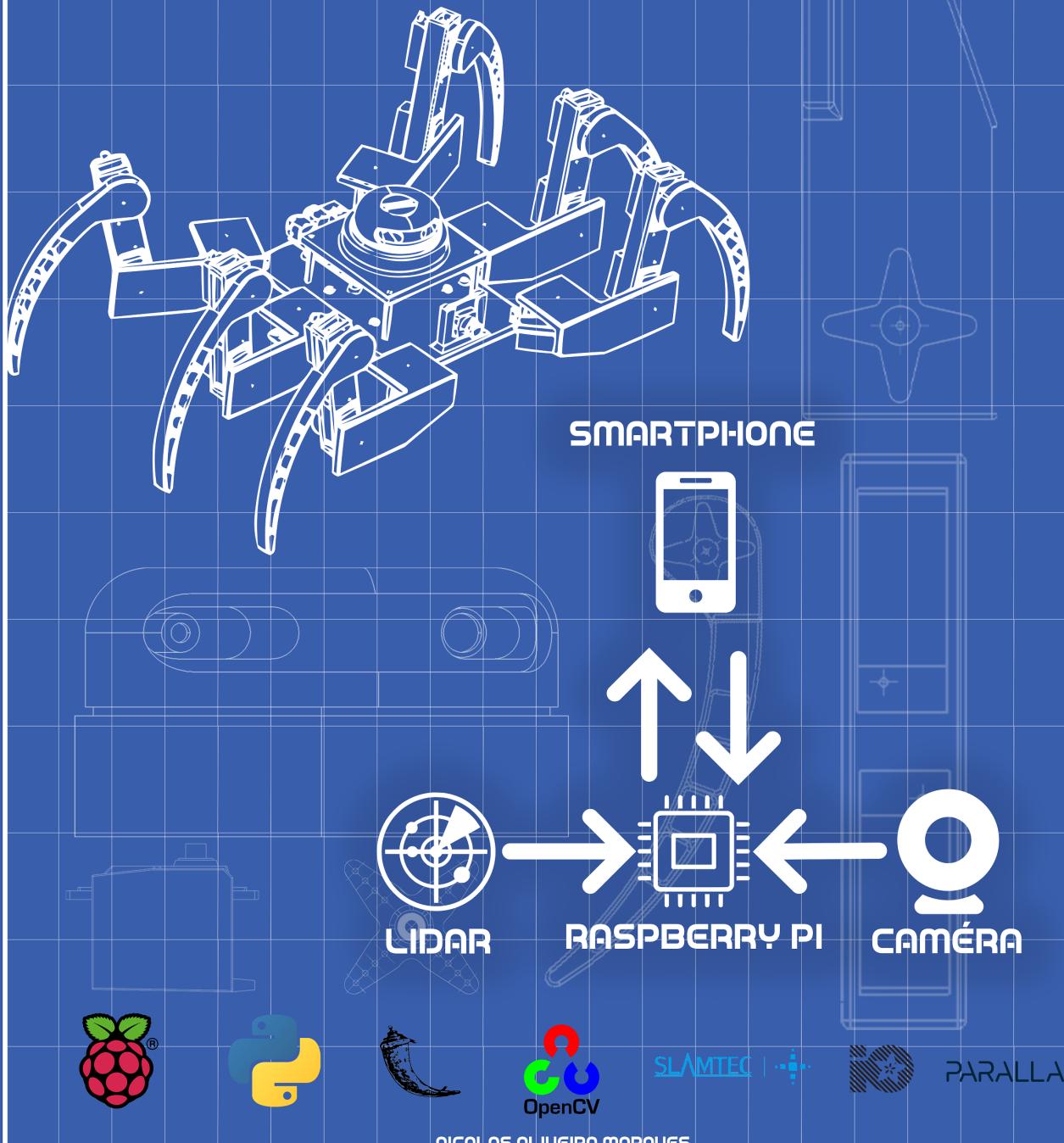
**09.05.2022**

Aujourd'hui, j'ai envoyé mon poster ainsi que mon abstract à M. Bonvin. Il m'a demandé de l'appeler, puis on a fait un débriefing par téléphone. Il m'a dit que le poster était pas mal, sauf que je ne présente rien d'informatique, je ne montre que des pièces mécaniques. Il m'a alors conseillé de faire un schéma qui explique ce qui communique ensemble, et de l'ajouter sur le poster. Ensuite, pour l'abstract, il m'a dit que mon français n'était vraiment pas bon, ce que je conçois, car j'ai écrit comme je pense. Il m'a ordonné de ne rien faire d'autre que ces deux choses toute la journée, ce que j'ai fait. Voici donc la version finale de mon poster:

# THE BIG BUG

## ROBOT HEXAPODE :

- CONTRÔLE VIA SMARTPHONE
- DÉTECTION DES OBSTACLES
- CONCEPTION ORIGINALE
- SUIVEUR DE PERSONNES



**10.05.2022**

Aujourd'hui, j'ai aidé un moment durant la matinée un camarade, car il avait besoin d'aide sur certaines notions d'électronique. J'ai donc passé un moment avec lui pour l'aider et l'orienter sur les bons choix. Ensuite, avec M. Hürlimann, nous avons eu notre réunion avec M. Bonvin. Heureusement, on ne reçoit pas la lettre chez nous, mais il faut que l'on s'active beaucoup plus, car on n'a rien de présentable pour l'instant. Durant l'après-midi, j'ai commencé à travailler sur les mouvements du robot, mais avec la classe «Patte», sauf que j'ai vite remarqué qu'en fait, cette classe mets inutile, car j'ai besoin de contrôler plusieurs pattes en même temps, et pas une à la fois. Je suis alors reparti sur le programme de base pour les animations, et j'ai remarqué que c'était très bizarre les valeurs que je recevais du gyroscope, je suis alors allé essayer, mais avec une boucle qui me montre en continu ce qu'il se passe. J'ai remarqué que les valeurs ne bougeaient plus dès que je bougeais plus le robot dans mes mains, alors que si je le bouge, là les valeurs changent. J'ai ainsi compris, que la librairie que j'utilise pour communiquer avec le MPU6050, avait inversé entre le gyroscope et l'accéléromètre. Ensuite, j'ai expliqué à M. Bonvin ce que je voulais faire, qui est d'utiliser le gyroscope pour stabiliser le robot, et ainsi qu'il peut lui-même mettre plus ou moins de puissance sur les servos. Il m'a alors parlé de boucles d'asservissement (je connaissais ce terme seulement par PID), connaissant la difficulté de faire un système asservi, j'ai fait mes recherches sur le Web, je n'ai rien qui m'explique bien la marche à suivre en programmation d'un PID, mais je pense savoir un peu près comment le faire.

**11.05.2022**

Aujourd'hui, j'ai passé toute la matinée à relire le fonctionnement du PID. J'ai demandé de l'aide à certains professeurs, et beaucoup m'ont dit de ne pas me lancer dans ça, et plutôt utiliser une librairie qui le fait pour moi. L'après-midi, j'ai essayé d'implémenter la librairie simple-pid, qui fait les calculs pour le PID à ma place. Mais je ne comprends pas où je dois mettre les valeurs qui me ressortent. J'ai fait quelques tests avec le robot, et soit il a le parkinson, et tous les servos tremblent dans tous les sens, soit il dit comme quoi c'est le plus juste et sort de la boucle du pid directement.

**12.05.2022**

Aujourd'hui, j'ai continué à implémenter le PID sur le robot, avec l'axe y. L'axe X sur calibre correctement, mais l'axe y n'y arrive pas vraiment. Ensuite, j'ai remarqué que les servos moteurs se fatiguaient très vite. Je suis obligé, afin d'avoir un bon résultat, de laisser reposer les servos afin qu'ils aient tous la puissance. De ce fait, chaque essai prend beaucoup de temps, car je dois laisser reposer entre 5 et 20 minutes. Puis, l'après-midi, M. Zanardi, m'a aidé avec le PID, car il y avait des notions que je n'avais pas, et il m'a expliqué un peu mieux comment fonctionne le PID, et que les valeurs de ki, kp et kd, je dois les trouver à tâtons. Puis, vers 14 h il y a eu des élèves qui sont venus pour qu'on leur présente nos projets. Et à partir de 16 h 10, il y a eu de la famille, ainsi que d'autres professeurs qui sont venus pour qu'on leur présente nos projets, et ce, jusqu'à 18 h.

**13.05.2022**

Aujourd'hui, j'ai travaillé sur l'animation pour que le robot avance. J'ai tout d'abord commencé avec une patte. En voyant que tout allait bien, j'ai décidé de faire l'avancement en quinconce. Puis, j'ai eu encore des problèmes avec le PID, j'ai essayé de le régler au mieux. Ça m'a pris beaucoup de temps d'essayer les mouvements, car les servos chauffent vite et je dois attendre beaucoup trop de temps avant d'essayer de nouveau. Puis, je pense que si les moteurs sont «fatigués», c'est parce que l'alimentation stabilisée, n'arrête pas de switché entre le c. c. et le c.v.. Il faudrait que je demande à un de mes anciens professeurs comment je peux faire pour que les servos consomment moins, tout en ayant (au moins l'impression) toute leur force.

**16.05.2022**

Aujourd'hui, j'ai commencé la journée en avançant le mouvement d'avancement du robot. Puis, vers la pause, M. Garcia est venu avec la commande des entretoises. J'ai monté le support pour le lidar, ainsi que mit le lidar sur le robot, et j'ai continué l'animation, sauf que le robot ne se levait plus du tout. Au début, je pensais que c'était à cause du lidar, alors je l'ai enlevé, mais le robot ne se levait toujours pas. J'ai eu peur d'avoir cassé tous les servos en même temps, sur un faux mouvement en allumant ce matin. Mais heureusement, après quelques minutes, le robot se levait de nouveau. Ensuite, durant l'après-midi, j'ai pensé qu'il serait temps de rassembler déjà toutes mes classes à l'interface. J'ai alors réglé le problème des plots qui prenait trop de temps à se charger, et ce en utilisant opencv et faire la même technique que pour la caméra. Puis, j'ai ajouté un select dans l'interface, afin de pouvoir facilement changer de mode.

**17.05.2022**

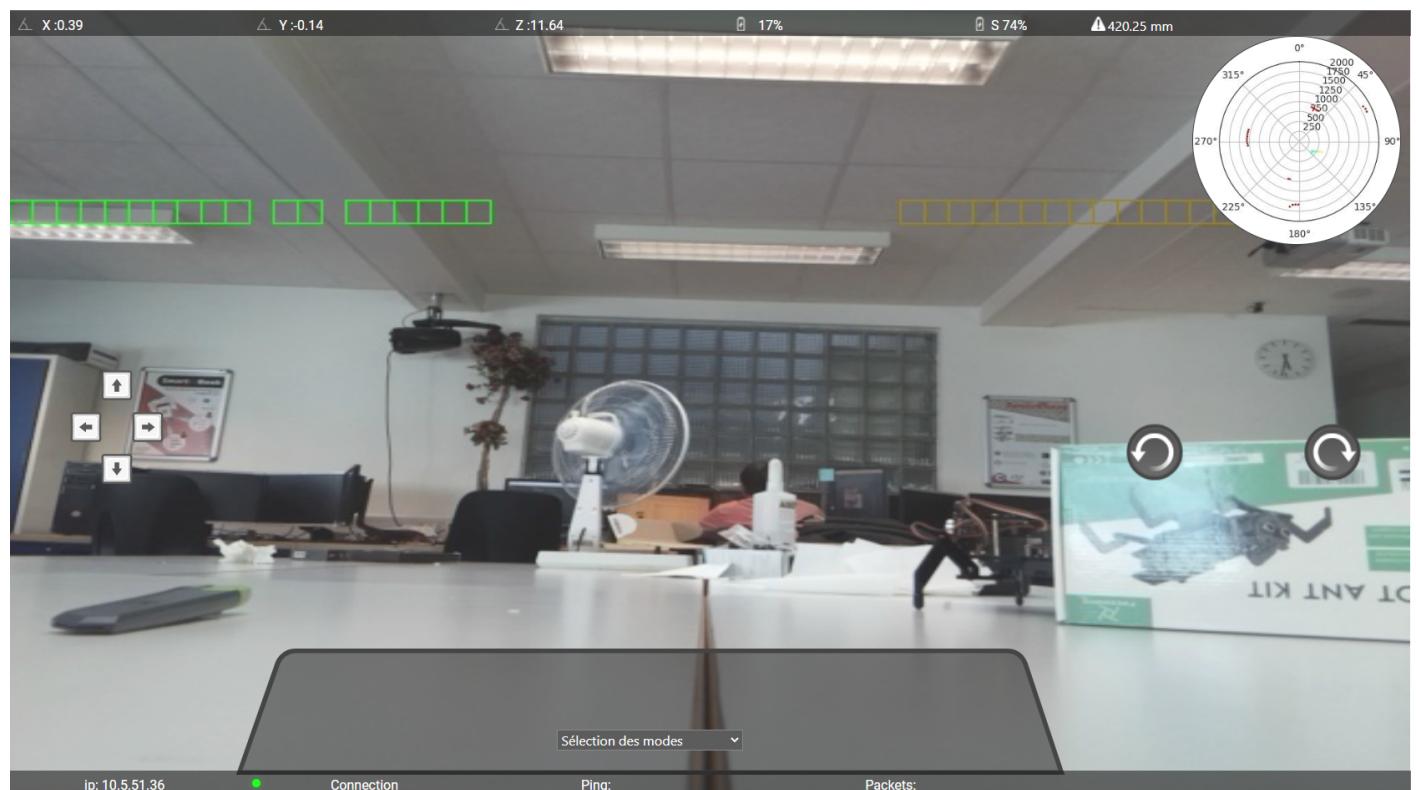
Ce matin, M. Bonvin était déjà là, nous avons un peu parlé et il m'a dit de faire un diagnostic par rapport aux problèmes liés au robot. Je pense qu'il veut que je parle déjà des problèmes rencontrés, ainsi que des améliorations à faire, et que ce robot là, ne soit qu'un prototype. Ensuite, j'ai continué de travailler sur l'animation pour avancer le robot. Je pense avoir fini l'animation, sauf que le robot ne se lève pas assez haut pour bien voir. En fin de journée, M. Bonvin est revenu pour analyser notre code voir s'il était en norme. Je lui ai fait la démonstration de l'interface, et il m'a donné plein d'idée et de chose à corriger. En termes de correction: rendre le site responsive, et en termes d'idée à ajouter sur l'interface: faire en sorte que lorsqu'on clique sur la vue radar elle s'agrandisse, faire une sorte de bannière verticale avec plein d'options dedans (choisir la taille de la vue radar, choisir que le fond de la vue radar soit en blanc ou transparent, etc.), puis faire un effet avec la caméra, qui grâce aux données du lidar, je puisse faire un quadrillage qui change de couleurs plus c'est près, et qui devient transparent quand c'est loin.

**18.05.2022**

Aujourd'hui, j'ai commencé la journée à regarder comment faire le quadrillage sur le retour caméra avec opencv. Puis, j'ai vu que je ne pouvais pas changer l'opacité du quadrillage, de ce fait, il y aura toujours un quadrillage vert, et quand un objet détecté par le lidar, ça changera la couleur. Ensuite, je me suis dit, pour que l'utilisateur comprenne, il faut que sur la vue radar, ce soit pareil. J'ai réussi à créer le quadrillage, et j'ai testé d'où à où le lidar détecte, comme ça le lidar ne prend pas toute la page, mais une ligne. Ensuite, je me suis dit qu'il faut que je rende le site responsive, ce que j'ai fait tout le reste de la journée.

**19.05.2022**

Aujourd'hui, j'ai passé ma journée à faire en sorte que le quadrillage change de couleur selon les valeurs du lidar. Ce que j'ai fait, c'est que je dessine des carrés (vert de base), puis en faisant un calcul en prenant en compte la largeur de l'image, les degrés et plein d'autres paramètres. Il y avait plein de problèmes, ce n'était pas centré, des fois c'était à l'envers, d'autres il manquait plein de carrés. Puis, en affinant de plus en plus le calcul, j'ai réussi à avoir un résultat, mais pas comme je le voulais. Basiquement, je ne peux pas remettre à zéro les valeurs, car sinon ça clignote de partout. voici à quoi ressemble la ligne des carrés:



**20.05.2022**

Aujourd'hui, j'ai continué à travailler sur la ligne de carré, sauf que je n'arrivais pas à empêcher le clignotement. J'ai donc essayé toute la journée plusieurs façons de faire, sauf que pas grand-chose ne marchait. J'ai parlé un petit moment avec M. Bonvin, il m'a vite expliqué comment faire et la logique à avoir. J'ai essayé, mais ça ne marchait pas. J'ai donc demandé de l'aide à mes camarades, puis ils m'ont conseillé de faire juste un else dans ma condition qui pourrait régler mon souci. Résultat, ça marche.

**23.05.2022**

Aujourd'hui, j'avais commencé à travailler sur les autres points que m'avait dits M. Bonvin la dernière fois. Puis, assez tôt il nous a envoyé un mail pour faire un meet sur ce qu'on avait fait en termes de code comme dans la documentation. On a passé 20-30 min entrain de discuté sur ce qui n'allait pas, et il m'a dit que maintenant je devais travailler encore plus, car je dois au moins avoir une interface fonctionnelle et professionnelle, ainsi qu'une documentation professionnelle. De ce fait, j'ai ensuite travaillé tout le reste de la journée sur la propreté du code, ainsi que de faire une refonte de la documentation technique afin que ça paraisse mieux.

**24.05.2022**

Aujourd'hui, j'ai continué à travailler sur la propreté de la documentation technique, et ajouté plein d'explications sur certains composants et bibliothèques. Puis, sur le reste de la journée, M. Bonvin est passé et m'a dit que ce que j'ai fait n'était pas adapté pour mes connaissances et que ça ne marchait pas. Alors, que ça marche, mais pas comme il l'avait imaginé. Du coup, j'ai travaillé sur la ligne de carré pour faire en sorte que ce soit plus ou moins comme ce qu'il souhaite. En le faisant, j'ai vu qu'il fallait que je fasse des dérivées afin d'agrandir ou de rapticir les carrés dépendement de la distance que le lidar détecte. Je ne sais pas faire de dérivées et je ne comprends pas cela. J'ai donc fait de mon mieux pour corriger l'affiche, et que ce soit droit par rapport à la caméra.

**25.05.2022**

Aujourd'hui, j'ai travaillé sur le JavaScript pour faire en sorte que dès qu'on clique sur la vue radar, qu'elle s'agrandisse. Comme je ne suis pas très fort en JavaScript, du fait que je n'en ai jamais fait, ou très peu, cela m'a pris beaucoup de temps pour le faire. Ensuite, j'ai commencé à faire le bouton qui active ou désactive la transparence de la vue radar, mais pour ça j'ai eu plein de problèmes. OpenCV quand il lisait le fichier «.png», il ne mettait pas l'alpha, du coup je n'arrêtais pas d'avoir un fond blanc dans la vue. Puis, en faisant quelques recherches, j'ai vu qu'il fallait mettre qu'un paramètre dans la fonction qui lit l'image. J'avais enfin de la transparence, mais maintenant il faut retourner sur le JavaScript. Et, je n'ai pas réussi à régler le background color en JavaScript.

**27.05.2022**

Aujourd'hui, durant la matinée j'ai recommencé à travailler sur le JavaScript qui ne fonctionnait pas. En regardant sur plein de forums, je vois que la fonction "rgba()" ne fonctionne pas. J'ai alors essayé dans l'inspecteur du navigateur web de changer les couleurs à la main (voir si c'était possible de mettre le fond transparent d'une png en blanc). Et de ce fait, j'ai donc changé mon code et au lieu de mettre la fonction "rgba()", j'y ai mis directement la couleur. Ensuite, le reste de la journée j'ai avancé ma documentation, ainsi que faire les corrections d'orthographe et de grammaire.

**30.05.2022**

Aujourd'hui, j'ai passé la journée en faisant de la documentation, j'ai fait tout le manuel d'installation. Puis, j'ai essayé de faire fonctionner le plug-in de mkdocs "mkdocstring", ça fonctionne pour certaines classes, mais pas pour toutes, il faut que demain j'essaie de faire fonctionner cela.

**31.05.2022**

Ce matin, j'ai passé toute la matinée à encore avancer la documentation. Mkdocstring n'est pas très simple à utiliser, mais j'ai finalement réussi à le faire fonctionner. Sauf que dès qu'il y a des bibliothèques pas installer sur l'ordinateur, il crash. Du coup, il beaucoup crasher, car j'ai des bibliothèques qui sont de Raspbian, et donc je ne peux pas les avoir sur mon PC Windows avec lequel je fais ma documentation. J'ai cherché un bon moment, et j'ai trouvé une de leur version spécialisée pour le Python qui ne prend pas

en compte des librairies. Ensuite, j'ai refait la mise en page de chaque script afin que ça paraisse correct sur le pdf et le site mkdocs. L'après-midi, j'ai essayé de faire fonctionner le binding des touches en JS. Elles marchent, sauf qu'elles ne vont pas jusqu'à la fonction pour faire bouger le robot.

### 6.3.3 Juin

---

#### 01.06.2022

Aujourd'hui, j'ai fait fonctionner le binding des boutons, il fallait que je rajoute une condition dans le script app.py. Puis, j'ai ajouté un bouton afin de faire lever le robot, ainsi que pour éteindre le robot, car j'ai remarqué que je n'avais rien prévu pour l'arrêter à par en ligne de commande. L'ajout, des boutons fut compliqué et long à cause du CSS, et le raspberry pi qui n'arrête pas de freeze.

#### 02.06.2022

Aujourd'hui, j'ai essayé de débugger le freeze du raspberry, et en le faisant j'avais changer que j'avais renommé un de mes dossiers pour le JavaScript. Ça m'affichait donc plus rien avec le JS. Ensuite, j'ai fait le côté backend pour éteindre correctement le robot, ainsi que de lever le robot. Puis, j'ai aussi préparé les fonctions des mouvements et si j'ai le temps, j'avancerais les mouvements. Et j'ai aussi regardé pour les données telles que les packets ainsi que le ping. Pour le ping il faudrait que j'ajoute une librairie pour ça, il faut encore que je fasse de plus amples recherches afin de voir si je ne peux pas le coder directement. Puis, pour les packets, je ne trouve rien. Voir si je change ces données pour d'autres.

#### 03.06.2022

Aujourd'hui, j'ai débuggé un problème que je viens de remarquer. Les sélections ne marchent plus. Je ne peux plus changer de mode, alors j'ai regardé et pourtant mes boutons pour me déplacer fonctionnent. j'ai donc mis le même nom pour le select que pour les boutons, et ça a fonctionné. Ensuite, je me suis lancé sur la détection des QRCode grâce à la caméra pour essayer d'implémenter quand même la logique du suiveur. À ma grande surprise, l'implémentation de celui-ci fut assez simple. Je n'ai eu que très peu de difficulté à le faire. Mais, je n'arrivais pas à faire en sorte qu'avec le lidar je puisse différencier la personne à suivre avec le QRCode et s'il y avait un obstacle entre les deux. J'ai donc laissé la logique du suiveur pour lundi, en espérant qu'un jour ou deux me suffiront pour l'implémenter. Ensuite, vers la fin de la journée, j'ai fait toute la logique pour le mode autonome. Ce n'est pas un algorithme très poussé, mais ça suffit pour qu'il puisse marcher continuellement.

#### 07.06.2022

Aujourd'hui j'ai avancé un maximum ma documentation technique, car le temps commence à devenir court. J'ai commencé à faire le manuel utilisateur, étape par étape. Ensuite, plus tard dans la matinée, M. Bonvin est venu, et nous a un peu tirés les oreilles, et qu'on était cuit avec notre code, donc il fallait que l'on gère au maximum la documentation. J'ai donc continué la journée, à revoir mes tournures de phrases, ajouter des images, etc.

#### 08.06.2022

Aujourd'hui, rebelote je continue à travaillé sur la documentation et j'ai bien changé toute l'introduction, et je l'ai rendu beaucoup plus technique qu'avant. Puis, j'ai remis à jour mon planning, créé plein de diagrammes qui expliquent mon code. Puis demain, j'essayerais de finir le manuel utilisateur, ainsi que rendre le code un peu plus propre.

#### 09.06.2022

Aujourd'hui, j'ai travaillé toute la journée sur la remise au propre de certaines pages de la documentation. Puis, j'ai fait la correction de l'orthographe de tous mes .md

#### 10.06.2022

Aujourd'hui, j'ai travaillé toute la nuit pour rendre la documentation la plus propre que je puisse faire. Mais malheureusement, je ne peux pas générer le pdf, car pour les sources, mkdocstring ne veut toujours pas fonctionner. Puis, j'ai réussi à régler certains bugs, et optimiser un peu le code.

## 6.4 Code Source



### 6.4.1 python

#### Classe Animations

`Animations`

Bases: `object`

Source code in `mod_classes/Animations_Hexapode.py`

`Avance()`

brief: Animation permettant de faire avancer le robot.

parameters

None

returns

None

Source code in `mod_classes/Animations_Hexapode.py`

`Droite()`

brief: Animation qui permet au robot de se déplacer vers la droite comme un crabe.

parameters

None

returns

None

Source code in `mod_classes/Animations_Hexapode.py`

`Gauche()`

brief: Animation qui permet au robot de se déplacer vers la gauche comme un crabe.

parameters

None

returns

None

Source code in `mod_classes/Animations_Hexapode.py`

**Init()**    ✓    ✓    ✓

brief: Animation qui lève le robot.

parameters

None

returns

None

Source code in `mod_classes/Animations_Hexapode.py`

**Init2()**

brief: Deuxième animation qui lève le robot.

parameters

None

returns

None

Source code in `mod_classes/Animations_Hexapode.py`

**Maintiens()**

brief: Fonction qui permet de rester droit grâce au PID et au Gyroscope.

parameters

None

returns

None

Source code in `mod_classes/Animations_Hexapode.py`

**Off()**

brief: Animation qui permet au robot d'arrêter tout les servomoteurs.

parameters

None

returns

None

Source code in `mod_classes/Animations_Hexapode.py`

```
Recule()   ▼   ▼   ▼
```

brief: Animation qui permet au robot de reculer.

parameters

None

returns

None

Source code in `mod_classes/Animations_Hexapode.py`

```
Rotation_AntiHoraire()
```

brief: Animation qui permet au robot de se tourner vers la gauche.

parameters

None

returns

None

Source code in `mod_classes/Animations_Hexapode.py`

```
Rotation_Horaire()
```

brief: Animation qui permet au robot de se tourner vers la droite.

parameters

None

returns

None

Source code in `mod_classes/Animations_Hexapode.py`

```
__init__(gyroscope)
```

brief: Initialisation des servomoteurs et du Gyroscope.

parameters

Gyroscope

returns

None

Source code in `mod_classes/Animations_Hexapode.py`

**Classe VideoCamera**

▼

**VideoCamera****Bases:** object

Source code in mod\_classes/Camera.py

**QRCodeDetect()****brief:** Detecte les QR Code dans une image.**parameters**

None

**returns**

Image

Source code in mod\_classes/Camera.py

**\_\_del\_\_()****brief:** Arrêt de la saisie de la caméra.**parameters**

None

**returns**

None

Source code in mod\_classes/Camera.py

**\_\_init\_\_()****brief:** Initialisation de la caméra et du flux vidéo.**parameters**

None

**returns**

None

Source code in mod\_classes/Camera.py

**get\_frame(scans)****brief:** Capture du flux vidéo de la caméra et retourne le flux vidéo.**parameters**

list[int] ↴  
returns  
bytes

Source code in `mod_classes/Camera.py`

**Classe Gyroscope****Gyroscope**

Bases: object

Source code in mod\_classes/Gyroscope.py

`__init__()`

brief: Initialisation du module Gyroscope.

parameters

None

returns

None

Source code in mod\_classes/Gyroscope.py

`get_acceleration()`

brief: Permet de récupérer les accélérations du robot.

parameters

None

returns

dict[str,float]

Source code in mod\_classes/Gyroscope.py

`get_angle()`

brief: Permet de récupérer les angles du robot.

parameters

None

returns

dict[str,float]

Source code in mod\_classes/Gyroscope.py

**Classe Leg**

**Leg**

Bases: object

Source code in mod\_classes/Leg.py

**Baisser\_Pointe(Angle, Force)**

brief: Baisse la pointe de la patte.

parameters

int int

returns

None

Source code in mod\_classes/Leg.py

**Baisser\_Tibia(Angle, Force)**

brief: Baisse le tibia de la patte.

parameters

int int

returns

None

Source code in mod\_classes/Leg.py

**Lever\_Pointe(Angle, Force)**

brief: Lève la pointe de la patte.

parameters

int int

returns

None

Source code in mod\_classes/Leg.py

**Lever\_Tibia(Angle, Force)**

brief: Lève le tibia de la patte.

parameters

```
int int  ✓  ✓
           ✓
returns
None
```

Source code in `mod_classes/Leg.py`

`Tourner(Angle, Force)`

brief: Tourne la patte.

parameters

int int

returns

None

Source code in `mod_classes/Leg.py`

`WithoutForce()`

brief: Enlève la force sur les moteurs.

parameters

None

returns

None

Source code in `mod_classes/Leg.py`

`__init__(Cote, Position)`

brief: Constructeur de la classe Leg, initialise les moteurs par rapport à l'emplacement de la patte.

parameters

string string

returns

None

Source code in `mod_classes/Leg.py`

**Classe LidarAsync****Lidarasync**

Bases: object

Source code in mod\_classes/LidarAsync.py

**DoScan()**

brief: Fonction effectuant le scan avec le lidar.

parameters

None

returns

None

Source code in mod\_classes/LidarAsync.py

**Get\_Data()**

brief: Fonction permettant de récupérer les données du lidar.

parameters

None

returns

list(list(float,float,float))

Source code in mod\_classes/LidarAsync.py

**StartLidar()**

brief: Fonction qui permet de démarrer le moteur du lidar.

parameters

None

returns

None

Source code in mod\_classes/LidarAsync.py

**StopLidar(withmotor=True)**

brief: Fonction permettant d'arrêter le lidar, et le déconnecte.

parameters

bool      ✓    ✓  
              ✓  
returns  
None

Source code in `mod_classes/LidarAsync.py`

`__init__()`

brief: Constructeur de la classe Lidar, initialise les valeurs et le lidar.  
parameters  
None  
returns  
lidar

Source code in `mod_classes/LidarAsync.py`

`__new__()`

brief: Je rends la classe en tant que singleton.  
parameters  
None  
returns  
instance

Source code in `mod_classes/LidarAsync.py`

**Classe Motor**

▼

▼

**ServoMotor**

Bases: object

Source code in mod\_classes/Motor.py

**SetAngleRel(Angle, Force)**

brief: Fonction permettant de définir un angle avec une force demandée(en pourcent).

parameters

int int

returns

None

Source code in mod\_classes/Motor.py

**StayWithForce(direction='+')**

brief: Fonction qui permet de garder le servomoteur avec un peu de force sur le sol.

parameters

string

returns

None

Source code in mod\_classes/Motor.py

**WithoutForce()**

brief: Fonction permettant de garder le servomoteur sans force.

parameters

None

returns

None

Source code in mod\_classes/Motor.py

**\_\_init\_\_(Channel='Gauche', Position=0)**

brief: Initialisation du module pour les servomoteurs ainsi que les valeurs.

parameters

```
string int
    ^
returns
None
```

Source code in `mod_classes/Motor.py`

**Classe Radar****Radar**

Bases: object

Source code in mod\_classes/Plot.py

`CreatePlot(scans)`

brief: Grâce aux obstacles détectés avec le lidar, et mets les points sur le plot.

parameters

list

returns

bytes

Source code in mod\_classes/Plot.py

`__init__()`

brief: Constructeur de la classe Radar, créer la base de l'image.

parameters

None

returns

None

Source code in mod\_classes/Plot.py

**Classe RepeatTimer****RepeatTimer**Bases: **Timer**Source code in `mod_classes/RepeatTimer.py``run()`

brief: fonction qui permet de faire du multithreading avec un Timer.

parameters

None

returns

None

Source code in `mod_classes/RepeatTimer.py`

**Script principal****add\_header(r)**

brief: Fonction executée après chaque requête.

parameters

request

returns

request

Source code in app/app.py

**before\_first\_request()**

brief: Fonction permettant de lancer le rafraîchissement de la page sur un thread.

parameters

None

returns

None

Source code in app/app.py

**generate\_frames(camera)**

brief: Fonction permettant de générer les images de la caméra.

parameters

VideoCamera

returns

bytes

Source code in app/app.py

**generate\_radar(radarparam)**

brief: Fonction permettant de générer la vue radar.

parameters

Radar

returns

bytes

Source code in app/app.py

```
index() ▾ ▾ ▾
```

brief: Route principale du site web.

parameters

None

returns

string

Source code in `app/app.py`

```
inject_load()
```

brief: Fonction permettant d'actualiser les valeurs sur la bannière.

parameters

None

returns

dict[string, Any]

Source code in `app/app.py`

```
plot()
```

brief: Route qui affiche la vue radar.

parameters

None

returns

Response

Source code in `app/app.py`

```
radar()
```

brief: Fonction récupère les données du lidar, et enlève les doublons.

parameters

None

returns

list(list(float,float,float))

Source code in `app/app.py`

`update_load()`

brief: Fonction permettant de rafraîchir le site.

parameters

None

returns

None

Source code in `app/app.py`

`video()`

brief: Route qui affiche le retour de la caméra.

parameters

None

returns

Response

Source code in `app/app.py`

## 6.4.2 Web

### Interface Web

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0, shrink-to-fit=no">
    <title>Interface_Flaskv3</title>
    <link rel="stylesheet" href="static/bootstrap/css/bootstrap.min.css">
    <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Roboto&display=swap">
    <link rel="stylesheet" href="static/css/styles.css">
    <link rel="stylesheet" href="static/css/css.css">
    <link rel="icon" href="static/icon/favicon.ico" />
    {{ turbo() }}
</head>

<body>
    <div class="container-fluid">
        <div id="load" class="row" style="text-align: center; margin-bottom: 10px;">
            <div class="col-sm-2" style="text-align: center; margin-right: 10px;">
                <div class="row" style="text-align: center; margin-bottom: 10px;">
                    <div class="col d-sm-flex align-items-sm-center" style="text-align: center; margin-right: 10px;">
                        
                        <p class="d-sm-flex justify-content-sm-center align-items-sm-end Valeurs_Banniere" style="margin: 0; font-size: 1em;">X : {{angleX}}
                    </div>
                </div>
            </div>
            <div class="col-sm-2" style="text-align: center; margin-right: 10px;">
                <div class="row" style="text-align: center; margin-bottom: 10px;">
                    <div class="col d-sm-flex align-items-sm-center" style="text-align: center; margin-right: 10px;">
                        
                        <p class="d-sm-flex justify-content-sm-center align-items-sm-end Valeurs_Banniere" style="margin: 0; font-size: 1em;">Y : {{angleY}}
                    </div>
                </div>
            </div>
            <div class="col-sm-2" style="text-align: center; margin-right: 10px;">
                <div class="row" style="text-align: center; margin-bottom: 10px;">
                    <div class="col d-sm-flex align-items-sm-center" style="text-align: center; margin-right: 10px;">
                        
                        <p class="d-sm-flex justify-content-sm-center align-items-sm-end Valeurs_Banniere" style="margin: 0; font-size: 1em;">Z : {{angleZ}}
                    </div>
                </div>
            </div>
            <div class="col-sm-2" style="text-align: center; margin-right: 10px;">
                <div class="row" style="text-align: center; margin-bottom: 10px;">
                    <div class="col d-sm-flex align-items-sm-center" style="text-align: center; margin-right: 10px;">
                        
                        <p class="d-sm-flex justify-content-sm-center align-items-sm-end Valeurs_Banniere" style="margin: 0; font-size: 1em;">{{batterie}}%
                    </div>
                </div>
            </div>
            <div class="col-sm-2" style="text-align: center; margin-right: 10px;">
                <div class="row" style="text-align: center; margin-bottom: 10px;">
                    <div class="col d-sm-flex align-items-sm-center" style="text-align: center; margin-right: 10px;">
                        
                        <p class="d-sm-flex justify-content-sm-center align-items-sm-end Valeurs_Banniere" style="margin: 0; font-size: 1em;">{{batterieServo}}%
                    </div>
                </div>
            </div>
            <div class="col-sm-2" style="text-align: center; margin-right: 10px;">
                <div class="row" style="text-align: center; margin-bottom: 10px;">
                    <div class="col d-sm-flex align-items-sm-center" style="text-align: center; margin-right: 10px;">
                        
                        <p class="d-sm-flex justify-content-sm-center align-items-sm-end Valeurs_Banniere" style="margin: 0; font-size: 1em;">{{obstacleDist}} mm
                    </div>
                </div>
            </div>
        </div>
        <div id="Row_Center" style="text-align: center; margin-bottom: 10px;">
            <div class="col-sm-4" style="text-align: center; margin-right: 10px;">
                <div class="row" style="text-align: center; margin-bottom: 10px;">
                    <div id="plotdiv" class="col-sm-4 d-sm-flex d-xl-flex justify-content-sm-end justify-content-xl-end align-items-xl-center" style="text-align: center; margin-right: 10px;">
                        
                    </div>
                </div>
            </div>
        </div>
        <form method="post" action="#">
            <div class="row" style="text-align: center; margin-bottom: 10px;">
                <div class="col-sm-3 col-xl-2" id="div_Buttons" style="text-align: center; margin-right: 10px;">
                    <div class="row" style="text-align: center; margin-bottom: 10px;">
                        <div class="col d-sm-flex justify-content-sm-center align-items-sm-center" style="text-align: center; margin-right: 10px;">
                            <button id="Avance" type="submit" name="submit_button" value="Avance" style="border: 0; padding: 0; background: none;"><span>Avance</span></button>
                        </div>
                    </div>
                    <div class="row" style="text-align: center; margin-bottom: 10px;">
                        <div class="col d-sm-flex justify-content-sm-end align-items-sm-center" style="text-align: center; margin-right: 10px;">
                            <button id="Gauche" type="submit" name="submit_button" value="Gauche" style="border: 0; padding: 0; background: none;"><span>Gauche</span></button>
                        </div>
                        <div class="col d-sm-flex justify-content-sm-start align-items-sm-center" style="text-align: center; margin-right: 10px;">
                            <button id="Droite" type="submit" name="submit_button" value="Droite" style="border: 0; padding: 0; background: none;"><span>Droite</span></button>
                        </div>
                    </div>
                </div>
            </div>
        </form>
    </div>

```

```

<div class="col d-sm-flex justify-content-sm-center align-items-sm-center">
    <button id="Arrière" type="submit" name="submit_button" value="Recule" style="border: 0; padding: 0; background: none;"><img class="img_Recule" alt="Recule button icon" /></button>
</div>
</div>
<div class="col">
    <div class="row" style="height: 100%;">
        <div class="col-lg-12 d-lg-flex justify-content-lg-center align-items-lg-end" style="background: url('static/img/Rectangle.svg') bottom">
            <div class="row">
                <div class="col">
                    <button id="btn_Up_Robot" type="submit" name="submit_button" value="Init">
                        
                    </button>
                    <button id="btn_ShutDown" type="submit" name="submit_button" value="Shut_Down">
                        
                    </button>
                </div>
            </div>
            <div id="div_Select" class="col d-lg-flex justify-content-lg-center align-items-lg-end" >
                <select id="Select_Modes" name="Select_Modes" onchange='if(this.value!=0) {this.form.submit();}'>
                    <optgroup label="Modes">
                        <option value="0">-Sélection des modes</option>
                        <option value="controle">Mode Contrôle utilisateur</option>
                        <option value="auto">Mode Autonome</option>
                        <option value="suiveur">Mode Suiveur</option>
                    </optgroup>
                </select>
            </div>
            <div class="col">
                
            </div>
        </div>
    </div>
</div>
<div class="col-sm-3 col-md-3 col-xl-3">
    <div class="row d-sm-flex" style="margin-top: 70px;">
        <div class="col-sm-6 d-sm-flex justify-content-sm-center">
            <button id="Anti-Horaire" type="submit" name="submit_button" value="Rotation_AntiHoraire" style="border: 0; padding: 0; background: none;"><img alt="Anti-Horaire button icon" /></button>
        </div>
        <div class="col-sm-6 d-sm-flex justify-content-sm-center">
            <button id="Horaire" type="submit" name="submit_button" value="Rotation_Horaire" style="border: 0; padding: 0; background: none;"><img alt="Horaire button icon" /></button>
        </div>
    </div>
</div>
</form>
<div id="info" class="row" >
    <div class="col-sm-2">
        <div class="row">
            <div class="col d-sm-flex align-items-sm-center">
                <p class="d-sm-flex justify-content-sm-center align-items-sm-end Valeurs_Banniere">ip: 10.5.51.36</p>
            </div>
        </div>
    </div>
    <div class="col-sm-2 col-lg-2">
        <div class="row">
            <div class="col d-sm-flex align-items-sm-center">
                <div id="Pulse_Connection" ></div>
                <p class="d-sm-flex justify-content-sm-center align-items-sm-end Valeurs_Banniere">Connection</p>
            </div>
        </div>
    </div>
    <div class="col-sm-2 col-lg-2">
        <div class="row">
            <div class="col d-sm-flex align-items-sm-center">
                <p class="d-sm-flex justify-content-sm-center align-items-sm-end Valeurs_Banniere">Ping: </p>
            </div>
        </div>
    </div>
    <div class="col-sm-2 col-lg-3">
        <div class="row">
            <div class="col d-sm-flex align-items-sm-center">
                <p class="d-sm-flex justify-content-sm-center align-items-sm-end Valeurs_Banniere">Packets:</p>
            </div>
        </div>
    </div>
</div>
<script src="../static/bootstrap/js/bootstrap.min.js"></script>
<script src="../static/javascript/js.js"></script>
</body>
</html>

```

## Script Javascript

```

let toggleRadar=false
let toggleBackGroundRadar=false

/***
 * @function toggleSizeRadar
 * @description Toggle la taille du radar.
 * @param {void}
 * @returns {void}
 * @author Nicolas Oliveira
 */
document.getElementById('plot').addEventListener('click', () => {
    toggleRadar=!toggleRadar
    if(toggleRadar){
        document.getElementById('plot').style.transform = "translate(-75%, 50%) scale(2)";
    }
    else{
        document.getElementById('plot').style.transform = "scale(1) translate(0, 0)";
    }
});
/***
 * @function toggleBGRadar
 * @description Toggle l'arrière plan du radar.
 * @param {void}
 * @returns {void}
 * @author Nicolas Oliveira
 */
document.getElementById('Radar_Transparency').addEventListener('click', () => {
    toggleBackGroundRadar=!toggleBackGroundRadar
    if(toggleBackGroundRadar){
        document.getElementById('plot').style.backgroundColor = 'WHITE' ;
    }
    else{
        document.getElementById('plot').style.backgroundColor = 'TRANSPARENT' ;
    }
});

/***
 * @function binding
 * @description Bind les évènements sur les boutons.
 * @param {void}
 * @returns {void}
 * @author Nicolas Oliveira
 */
//si j'appuye sur T sur le clavier, le bouton "Radar_Transparency" s'active
document.addEventListener('keydown', (event) => {
    if (event.key == 't') {
        document.getElementById('Radar_Transparency').click();
    }
});

//si j'appuye sur W sur le clavier, le bouton "Avance" s'active
document.addEventListener('keydown', (event) => {
    if (event.key == 'w') {
        document.getElementById('Avance').click();
    }
});

//si j'appuye sur A sur le clavier, le bouton "Gauche" s'active
document.addEventListener('keydown', (event) => {
    if (event.key == 'a') {
        document.getElementById('Gauche').click();
    }
});

//si j'appuye sur S sur le clavier, le bouton "Arrière" s'active
document.addEventListener('keydown', (event) => {
    if (event.key == 's') {
        document.getElementById('Arrière').click();
    }
});

//si j'appuye sur D sur le clavier, le bouton "Droit" s'active
document.addEventListener('keydown', (event) => {
    if (event.key == 'd') {
        document.getElementById('Droit').click();
    }
});

//si j'appuye sur flèche gauche sur le clavier, le bouton "Anti-Horaire" s'active
document.addEventListener('keydown', (event) => {
    if (event.key == 'ArrowLeft') {
        document.getElementById('Anti-Horaire').click();
    }
});

//si j'appuye sur flèche droit sur le clavier, le bouton "Horaire" s'active
document.addEventListener('keydown', (event) => {
    if (event.key == 'ArrowRight') {
        document.getElementById('Horaire').click();
    }
})

```

```
});  
  
//si j'appuye sur I sur le clavier, le bouton "Init" s'active  
document.addEventListener('keydown', (event)=>{  
    if (event.key == 'i'){  
        document.getElementById('btn_Up_Robot').click()  
    }  
});  
  
//si j'appuye sur X sur le clavier, le bouton "ShutDown" s'active  
document.addEventListener('keydown', (event)=>{  
    if (event.key == 'x'){  
        document.getElementById('btn_ShutDown').click()  
    }  
});
```

## CSS

```

body{
    width: 100vw!important;
    height: 100vh!important;
    overflow: hidden!important;
}

#Camera{
    height: 100%;
    width: 100%;
    position: absolute;
    bottom: -1px;
    left: -1px;
    right: -1px;
    top: -1px;
    z-index: -50;
}

#load{
    background: rgba(0,0,0,0.6);
    width: 100%;
    left: 10px;
    right: 0;
}

(Icons{
    width: 17px;
    height: 17px;
}

.Valeurs_Banniere{
    width: 100%;
    height: 30px;
    font-size: 16px;
    margin-bottom: 0px;
    color: var(--bs-white);
    font-family: Roboto, sans-serif;
}

#info{
    position: absolute;
    bottom: 0;
    right: 0;
    left: 10px;
    background: rgba(0,0,0,0.6);
}

#Row_Center{
    height: 150px;
}

#plot{
    width: 250px;
    height: 250px;
    background-color: rgb(0, 0, 0, 0.2);
}

.Buttons_Direction{
    width: 50px;
}

.Buttons_Rotation{
    width: 70px;
}

#btn_Up_Robot{
    margin-top: 130px;
    height: 100x;
    width: 100px;
    background: none;
    border: 0px;
    transform: translateY(200);
}

#Up_Icon{
    filter: invert(100%) sepia(0%) saturate(0%) hue-rotate(75deg) brightness(100%) contrast(100%)
}
#ShutDown_Icon{
    filter: invert(100%) sepia(0%) saturate(0%) hue-rotate(75deg) brightness(100%) contrast(100%)
}

#btn_ShutDown{
    margin-top: 0px;
    height: 100px;
    width: 100px;
    background: none;
    border: 0px;
    transform: translateX(-150px) translateY(-50px);
}

#Select_Modes{

```

```

color: rgb(255, 255, 255);
background: rgb(68, 68, 68);
margin-top: 110px;
}

#div_Select{
padding-bottom: 10%;

}

#Radar_Transparency{
width: 100px;
height: 100px;
margin-top: 130px;
transform:translateX(50px) ;
}
.d-lg-flex {
display: flex!important;
}
.align-items-lg-end {
align-items: flex-end!important;
}
.justify-content-lg-center {
justify-content: center!important;
}
form{
position: absolute;
bottom: 25px;
left: 0;
right: 0;
}

#div.Buttons{
padding-bottom: 20%;

}

#Pulse_Connection{
background: rgba(0,255,0,1);
width: 10px;
height: 10px;
padding-right: 10px;
box-shadow: 10px 10px 20px green;
border-radius: 50%;
animation: pulsate 2s infinite;
}

@keyframes pulsate {
0% {
box-shadow: 0 0 4px 4px rgba(0, 255, 0, 0.2);
}
50% {
box-shadow: 0 0 5px 5px rgba(0, 255, 0, 0.1);
}
100% {
box-shadow: 0 0 4px 4px rgba(0, 255, 0, 0.2);
}
}

/* CSS for Extra Large (xl) screen */
@media only screen and (max-width: 1440px) {
.plot{
width: 250px;
height: 250px;
}
#Radar_Transparency{
width: 75px;
height: 75px;
margin-top: 110px;
transform:translateX(50px);
}

#btn_Up_Robot{
margin-top: 110px;
height: 75x;
width: 75px;
background: none;
border: 0px;
transform: translateX(50px);
}

#btn_ShutDown{
height: 75px;
width: 75px;
background: none;
border: 0px;
transform: translateX(0px) translateY(-75px);
}

}

/* CSS for Extra Large (xl) screen */
@media only screen and (max-height: 550px) {

```

```

#div_Buttons{
    padding-bottom: 0;
}
}

/* CSS for Extra Large (xl) screen */
@media only screen and (max-width: 1366px) {
    #plot{
        width: 200px;
        height: 200px;
    }
    #Radar_Transparency{
        width: 75px;
        height: 75px;
        margin-top: 150px;
        transform:translateX(50px) translateY(40px);
    }
    #btn_Up_Robot{
        margin-top: 150px;
        height: 75x;
        width: 75px;
        background: none;
        border: 0px;
        transform: translateX(50px) translateY(40px);
    }
    #btn_ShutDown{
        height: 75px;
        width: 75px;
        background: none;
        border: 0px;
        transform: translateX(-30px) translateY(-25px);
    }
}

/* CSS for Large (lg) screen */
@media only screen and (max-width: 1280px) {
    #plot{
        width: 180px;
        height: 180px;
    }
}

/* CSS for Large (lg) screen */
@media only screen and (max-width: 1280px) {
    #plot{
        width: 180px;
        height: 180px;
    }
}

/* CSS for Large (lg) screen */
@media only screen and (max-width: 1152px) {
    #plot{
        width: 170px;
        height: 170px;
    }
    #Radar_Transparency{
        width: 50px;
        height: 50px;
        margin-top: 110px;
        transform:translateX(50px) translateY(40px);
    }
    #btn_Up_Robot{
        margin-top: 120px;
        height: 50px;
        width: 50px;
        background: none;
        border: 0px;
        transform: translateX(50px) translateY(30px);
    }
    #btn_ShutDown{
        height: 50px;
        width: 50px;
        background: none;
        border: 0px;
        transform: translateX(0px) translateY(-20px);
    }
}

/* CSS for Large (lg) screen */
@media only screen and (max-width: 1024px) {
    #plot{
        width: 160px;
        height: 160px;
    }
}

/* CSS for Large (lg) screen */
@media only screen and (max-width: 992px) {

```

```

.plot{
    width: 150px;
    height: 150px;
}

/* CSS for Medium (md) screen */
@media only screen and (max-width: 800px) {
    .plot{
        width: 140px;
        height: 140px;
    }
    #Radar_Transparency{
        width: 50px;
        height: 50px;
        margin-top: 110px;
        transform:translateX(25px);
    }

    #btn_Up_Robot{
        margin-top: 110px;
        height: 50px;
        width: 50px;
        background: none;
        border: 0px;
        transform: translateX(25px);
    }
    #btn_ShutDown{
        margin-top: 0px;
        height: 50px;
        width: 50px;
        background: none;
        border: 0px;
        transform: translateX(0px) translateY(-25px);
    }
}

/* CSS for Medium (md) screen */
@media only screen and (max-width: 768px) {
    .plot{
        width: 130px;
        height: 130px;
    }
    #Radar_Transparency{
        width: 25px;
        height: 25px;
        margin-top: 120px;
        transform:translateX(0px);
    }

    #btn_Up_Robot{
        margin-top: 110px;
        height: 25px;
        width: 25px;
        background: none;
        border: 0px;
        transform: translateX(0px);
    }
    #btn_ShutDown{
        margin-top: 0px;
        height: 25px;
        width: 25px;
        background: none;
        border: 0px;
        transform: translateX(0px) translateY(-25px);
    }
}

/* CSS for Medium (md) screen */
@media only screen and (max-width: 600px) {
    .plot{
        width: 120px;
        height: 120px;
    }
    #Radar_Transparency{
        width: 25px;
        height: 25px;
        margin-top: 120px;
        transform:translateX(0px);
    }

    #div_Select{
        padding-bottom: 0%;
        transform: translateY(130px);
    }

    #btn_Up_Robot{
        margin-top: 120px;
        height: 25px;
        width: 25px;
        background: none;
        border: 0px;
    }
}

```

```
        transform: translateX(0px);
    }
#btn_ShutDown{
    margin-top: 110px;
    height: 25px;
    width: 25px;
    background: none;
    border: 0px;
    transform: translateX(0px);
}

}

/* CSS for Extra Small (xs) screen */
@media only screen and (max-width: 414px) {
    #plot{
        width: 110px;
        height: 110px;
    }
}

/* CSS for Extra Small (xs) screen */
@media only screen and (max-width: 394px) {
    #plot{
        width: 100px;
        height: 100px;
    }
}

/* CSS for Extra Small (xs) screen */
@media only screen and (max-width: 375px) {
    #plot{
        width: 90px;
        height: 90px;
    }
}

/* CSS for Extra Small (xs) screen */
@media only screen and (max-width: 360px) {
    #plot{
        width: 80px;
        height: 80px;
    }
}

/* CSS for Extra Small (xs) screen */
@media only screen and (max-width: 320px) {
    #plot{
        width: 70px;
        height: 70px;
    }
}
```