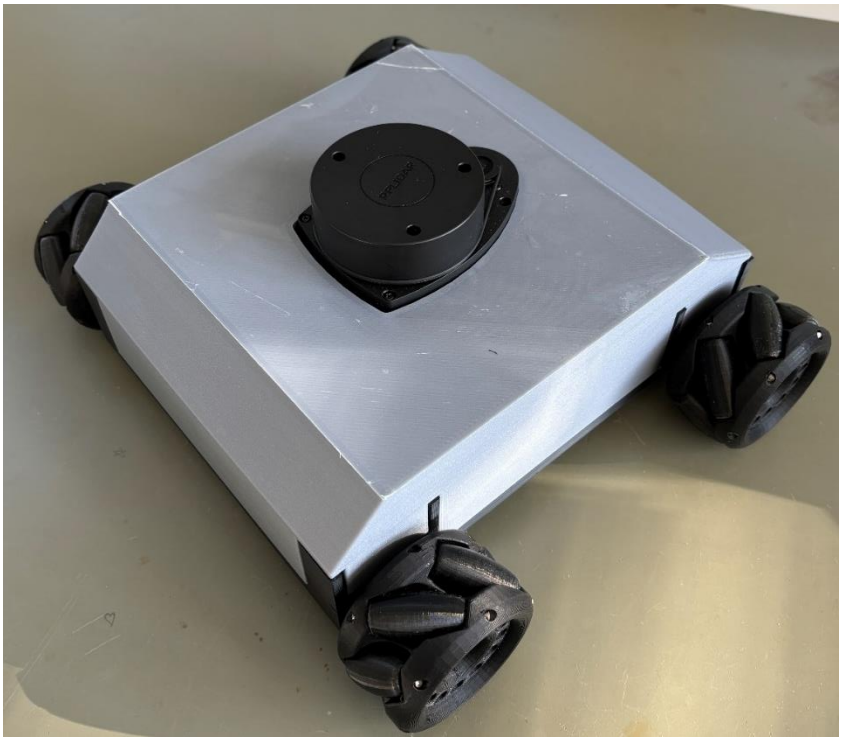
PROJET Rodar		
EMT – PORRENTRUY	ACP-MMC	28/01/2024 – 12/06/2024

CEJEF
DIVISION TECHNIQUE
ÉCOLE DES MÉTIERS TECHNIQUES

DIVISION TECHNIQUE DU CEJEF



« Rapport du projet Rodar »

Arnaud COUDERT

Ecole des Métiers Techniques (EMT)

Cité des Microtechniques

2900 Porrentruy

Date de remise : 12/06/2024

1 Introduction 2
1.1 Candidat 2
1.1.1 Description du projet 2
2 Développement du projet 3
2.1 LiDAR 3
2.1.1 Principe 3
2.1.2 Fonctionnement 3
2.1.3 Implémentation 6
2.2 Mouvement 7
2.2.1 Roues Mecanum 7
2.2.2 Moteurs 8
2.2.3 Modules L298N 8
2.3 Cartes logiques 8
2.3.1 Raspberry Pi Pico WLAN 8
2.3.2 Raspberry Pi 3B+ 9
2.4 Alimentation 9
3 Liste des composants 10
4 Fabrication 13
4.1 PCB 13
5 Programmation 14
5.1 Python et les versions utilisées 14
5.2 La communication entre les appareils 14
5.3 Le Deep Learning 16
5.3.1 Définition 16
5.3.2 Son développement 16
5.3.3 Les problèmes rencontrés 18
5.4 La solution algorithmique 18
6 Schéma du fonctionnement 19
7 Conclusion 20
7.1 Bilan du projet 20
7.2 Bilan personnel 20
7.3 Points d'amélioration 20

1 Introduction

1.1 Candidat

Je m'appelle Arnaud COUDERT et je réalise une année de connaissance professionnelle afin d'intégrer une HES. Lors de mon premier semestre à l'Ecole des Métiers Techniques, j'ai été formé aux bases de la modélisation, de la conception assistée par ordinateur à l'électronique. J'ai aussi reçu une brève formation dans l'atelier d'usinage.

1.1.1 Description du projet

Voulant poursuivre ma formation dans le domaine de la robotique, j'ai décidé de réaliser un véhicule autonome. Equipé d'un Lidar (explication p.3) comme moyen de perception de son environnement, le véhicule a pour objectif d'avancer dans un environnement inconnu jusqu'à un point prédéfini et de cartographier les obstacles rencontrés sur son chemin afin de prévoir le passage de futurs autres véhicules. Je tire mon inspiration des robots radiocommandé équipé d'une lance à eau utilisé par la BSSP (Brigade de sapeurs-pompiers de Paris) permettant de s'introduire dans un bâtiment inconnu, et d'approcher le feu.

Mon projet a pour but d'apporter une solution automatique lors de mission de sauvetage dans un environnement inconnu et inaccessible à l'Homme.

Pour le fonctionnement du robot, les programmes informatiques sont tous écrits dans le langage Python v3, car il offre une grande variété de bibliothèque et c'est aussi un langage que je maitrise.

Combinant les termes Lidar et Robot, mon projet a comme nom « Rodar ».

2 Développement du projet

2.1 LiDAR

2.1.1 Principe

Utilisant la technologie des lasers, les premiers LiDAR apparaissent peu de temps après la création de cette technologie. Leur création remonte à la conquête spatiale dans un but de cartographier la Lune.

Un LiDAR est un capteur de distance. Un faisceau laser est envoyé sur une surface et un capteur de lumière déduit la distance grâce au temps de propagation de la lumière.

Il existe plusieurs types de LiDAR. Le premier type sert à la modélisation et ressemble plus à un appareil photo. Nous retrouvons d'ailleurs un Lidar de modélisation dans les derniers iPhone en gamme Pro.

L'autre type de Lidar, qui est beaucoup plus présent dans le monde de la robotique et que j'ai utilisé pour mon projet est un Lidar rotatif ; capable de scanner son environnement à 360°.

Pour mon projet, j'ai opté pour un Lidar d'entrée de gamme modèle RPLIDAR A1 de la marque Slamtec.

2.1.2 Fonctionnement

Le RPLIDAR A1 est un lidar à vision 360° compact, avec une plage de mesure de 0.15 m à 6 m. Un moteur alimenté avec du 5V fait tourner la partie principale comportant le laser et le capteur permettant d'avoir la mesure de distance. Utilisant l'UART/USB comme interface de communication, l'échange de données entre le Lidar et le microcontrôleur se fait facilement et surtout rapidement.

L'UART est un protocole de communication série qui permet l'échange de données entre deux périphériques via des lignes de transmission et de réception distinctes (TX et RX).

Pour formuler une requête au Lidar, la commande est composée de 2 octets : un premier d'initialisation (0xA5) et le second indiquant la requête voulue. Avec ce tableau, nous pouvons voir les différentes requêtes possibles ainsi que l'octet de ces dernières :

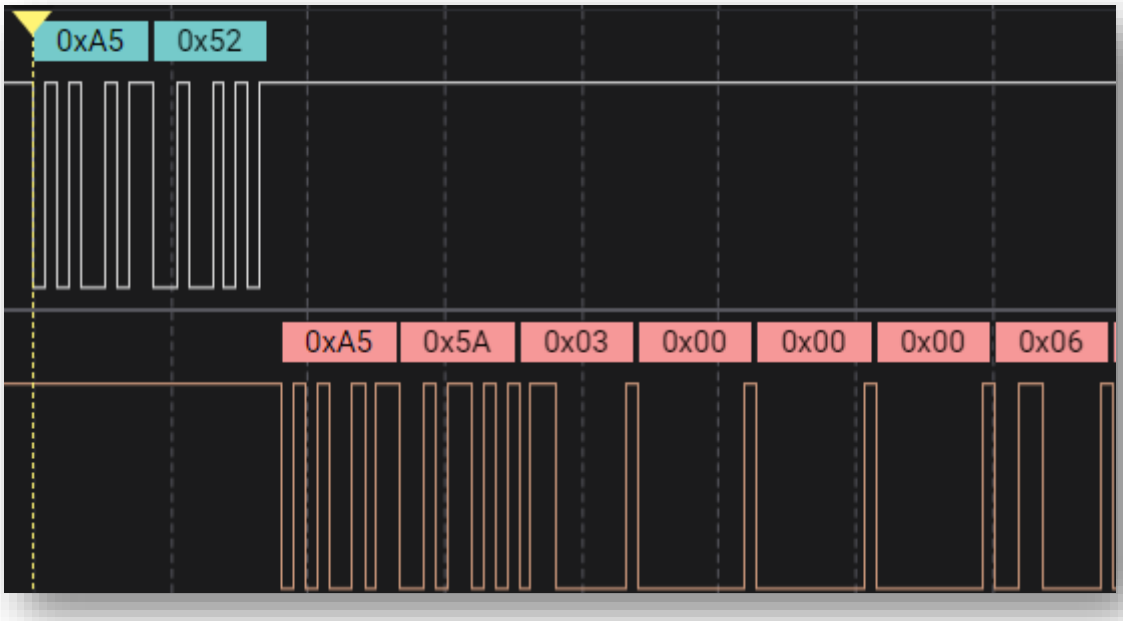
Request Name	Value	Payload	Response Mode	RPLIDAR Operation
STOP	0x25	N/A	No response	Exit the current state and enter the idle state
RESET	0x40	N/A		Reset(reboot) the RPLIDAR core
SCAN	0x20	N/A	Multiple response	Enter the scanning state
EXPRESS_SCAN	0x82	YES		Enter the scanning state and working at the highest speed
GET_INFO	0x50	N/A	Single response	Send out the device info (e.g. serial number)
GET_HEALTH	0x52	N/A		Send out the device health info
GET_SAMPLERATE	0x59	N/A		Send out single sampling time
GET_LIDAR_CONF	0x84	YES		Get LIDAR configuration

Par exemple, si nous voulons demander l'état du Lidar, nous envoyons la requête 0xA552.

Pour la réponse, le lidar envoie 7 octets, les deux premiers sont pour initialiser la réponse (0xA5 5A) et les 5 autres dépendent du type de requête. Dans mon utilisation, nous n'avons besoin que de 3 requêtes : GET_HEALTH, SCAN et STOP, donnant 2 formats de réponses.

Pour l'état du Lidar, la réponse est généralement 0xA5 5A 03 00 00 00 06. Après les deux octets d'initialisations, des données du Lidar et de son état sont encadrés entre le 0x03 et 0x06. Le premier octet donne l'état (0 : Good, 1 : Warning, 2 : Danger) et les deux autres donnent un code d'erreur s'il y en a un.

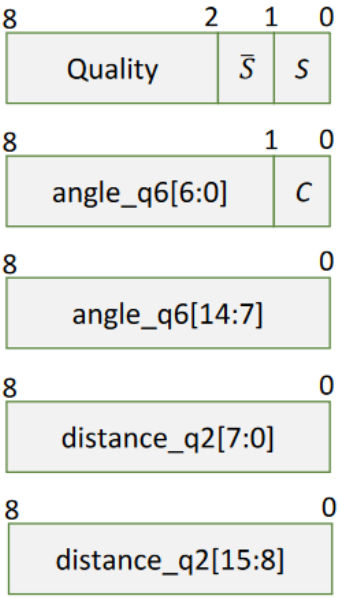
Voici une lecture analogique de la requête GET_HEALTH et sa réponse :



Le fonctionnement du démarrage du SCAN est un peu différent. Lorsque que la requête du scan est reçue par le Lidar, il ne fera qu'envoyer des réponses jusqu'à ce qu'il reçoive la commande STOP.

Pour la réponse, il y a toujours les deux premiers octets d'initialisations, suivi de 1 octet contenant la qualité de mesure en pourcentage avec 2 bits permettant d'authentifier la réponse, suivi de 2 octets pour l'angle de rotation et deux octets pour la distance.

Par exemple : l'envoie du Lidar : A5 5A 3E 8F 65 27 2C correspond à : l'angle 203.11, la distance est de 2825.75 mm et la qualité de 15/15.



2.1.3 Implémentation

Pour l'implémentation du Lidar à mon robot, j'ai décidé d'utiliser la bibliothèque écrite par Adafruit disponible gratuitement sur GitHub sous le nom Adafruit_RPLidar que j'ai modifiée et simplifiée afin que le code soit plus adapté à mon utilisation.

Grâce à la liste de commande, je peux facilement envoyer une commande avec la fonction : « _send_cmd ».

```
def _send_cmd(self, cmd: bytes):
    """Sends `cmd` command to the sensor"""
    req = SYNC_BYTE + cmd
    self.port.write(req)
```

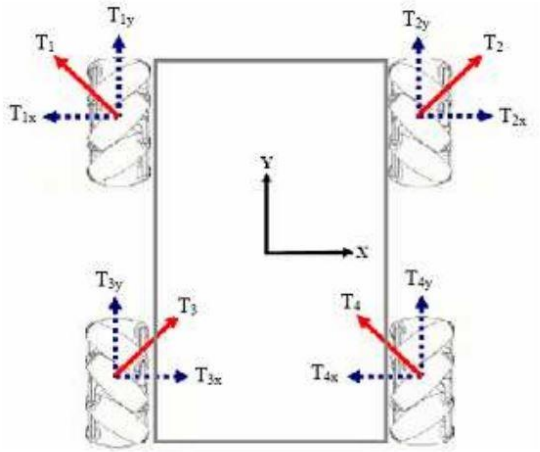
```
SYNC_BYTE = b"\xA5"
SYNC_BYTE2 = b"\x5A"

GET_INFO_BYTE = b"\x50"
GET_HEALTH_BYTE = b"\x52"

STOP_BYTE = b"\x25"
RESET_BYTE = b"\x40"
```

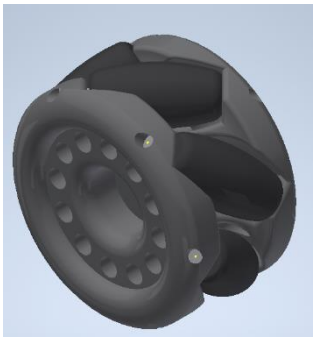
Une fois que la commande START a été envoyée au Lidar, la fonction « _process_scan » a pour but de récupérer ce que le Lidar a envoyé comme mesure et de transformer les bytes en données comme la qualité de mesure, l'angle de rotation et la distance de mesure.

```
def _process_scan(raw: bytes):
    """Processes input raw data and returns measurement data"""
    new_scan = bool(raw[0] & 0b1)
    inversed_new_scan = bool((raw[0] >> 1) & 0b1)
    if new_scan == inversed_new_scan:
        raise RPLidarException("New scan flags mismatch")
    check_bit = raw[1] & 0b1
    if check_bit != 1:
        raise RPLidarException("Check bit not equal to 1")
    quality = raw[0] >> 2
    angle = ((raw[1] >> 1) + (raw[2] << 7)) / 64.0
    distance = (raw[3] + (raw[4] << 8)) / 4.0
    return new_scan, quality, angle, distance
```

2.2 Mouvement

2.2.1 Roues Mecanum



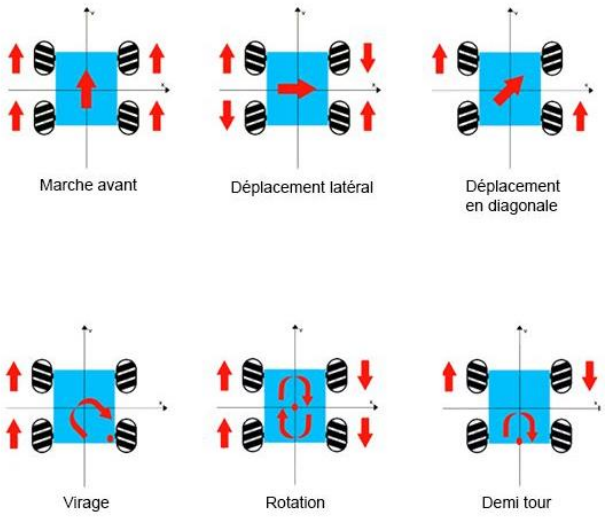


Toujours intrigué lorsque je visionnais des vidéos montrant des roues Mecanum, j'ai décidé d'en utiliser sur mon robot.

Contrairement à des roues classiques, les roues Mecanum sont entourées de pignons arrondis avec un angle de 45°.

Ainsi, lorsque la roue tourne, elle n'applique pas une force vers l'avant comme une roue classique mais avec un angle de 45° sur la surface.

En manipulant chaque roue individuellement avec des moteurs, cela nous permet d'avoir une multitude de mouvements qui sont impossibles avec des roues ordinaires. Les roues Mecanum permettent de déplacer le véhicule vers l'avant et l'arrière, mais aussi sur les côtés, en diagonale et permet de faire des rotations.



Pour mon robot, c'est pour moi le meilleur choix car cela permet de se faufiler dans des endroits assez étroits et de pouvoir réaliser des manœuvres sans nécessiter un grand espace.

2.2.2 Moteurs

Pour faire fonctionner les roues Mecanum, il me faut 4 moteurs différents pour mettre en rotation les 4 roues. J'ai ainsi opter pour 4 moteurs à engrenages 12 V à courant continu pouvant réaliser 620 tours par minute. Les engrenages permettent d'avoir un couple important, afin que les roues ne s'arrêtent pas sur le premier obstacle venu. Cela permet aussi de faire de petit mouvement précis, sans avoir besoin d'élan.

2.2.3 Modules L298N

L'avantage d'un moteur à courant continu est qu'il peut tourner dans les deux sens, suivant le sens du courant. Pour pouvoir alterner à quelle borne le courant est envoyé, j'ai décidé d'utiliser deux modules L298N.

Chaque module contrôle le sens de rotation ainsi que la vitesse de 2 moteurs grâce au PWM (Pulse With Modulation) qui consiste à envoyer du courant pendant 40% du temps pour avoir 40% de la vitesse maximale par exemple.



De plus, chaque module transforme le courant d'entrée en un courant de 5V, ce qui alimente tous les composants nécessitant cette tension pour fonctionner

2.3 Cartes logiques

2.3.1 Raspberry Pi Pico WLAN

Comme écrit plus haut, Python est un langage de programmation que j'apprécie et que je maitrise. J'ai donc cherché un cerveau pour mon robot capable d'exécuter des programmes Python et je me suis orienté vers le Raspberry Pi Pico WLAN. Que j'appellerai Pico dans ce rapport pour des raisons de simplicité.

Un Pico est un microcontrôleur peu cher, ≈ 7 CHF, et qui est performant grâce à la puce RP2040. Le Pico possède 30 branches GPIO configurables comme on le souhaite, ce qui est nécessaire pour brancher les 8 branches PWM pour les moteurs et 2 branches UART pour la communication avec le Lidar.

La capacité WLAN permet au microcontrôleur de se connecter à un réseau WI-FI ou Bluetooth, permettant ainsi une connexion avec un ordinateur ou un smartphone.

Lors de mes tests de communications entre le Lidar et le Pico, je me suis rendu compte que le bus d'envoi de données n'était pas assez grand sur le Pico et que mon programme cessait de fonctionner du fait d'un surplus de données. Il fallait donc que je trouve une alternative au Pico, capable de lire toutes les informations du Lidar.

2.3.2 Raspberry Pi 3B+

En restant chez la marque Raspberry pour sa compatibilité avec Python, je me suis alors orienté vers la gamme Pi, qui est nettement plus puissante que le Pico.

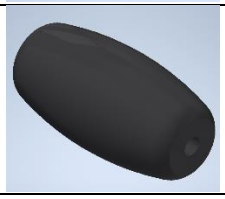
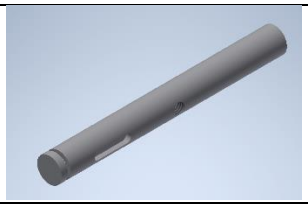
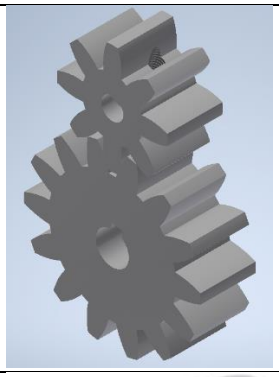
Après la configuration du Pi avec un OS simplifié et adapté de Linux, les tests sont revenus positifs et le Pi était capable de lire les informations du Lidar. Cependant, ce que je n'avais pas étudié avant était sa capacité à contrôler le Lidar et les moteurs. Malheureusement, le Pi ne possède pas assez de branches PWM pour contrôler les moteurs. J'ai donc été obligé d'utiliser les deux microcontrôleur, relié à un ordinateur afin qu'ils se coordonnent. Tous les détails de ce fonctionnement seront expliqués dans le chapitre sur la programmation.

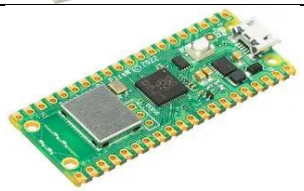
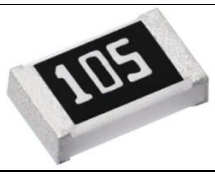
2.4 Alimentation

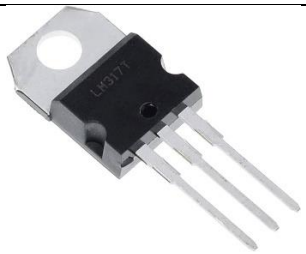
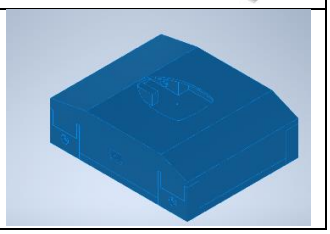
Pour alimenter mon robot, je me suis tourné vers une batterie destinée initialement pour les voitures de courses télécommandées. C'est une batterie LiPo rechargeable de 2 cellules avec courant nominal de 7.4V et une capacité de 5500 mAh. Mon robot consommant environ 1 A, la batterie est capable de faire fonctionner le robot pendant environ 5h30, ce qui est suffisant pour atteindre son objectif.

Les deux modules L298N, transformant le courant de la batterie en 5V, alimentent le Lidar, son moteur et le Pico. Le Pi quant à lui, demande un peu plus de courant (≈5.2V). J'ai donc créé un abaisseur de tension grâce à un LM317T et des résistances.

3 Liste des composants

Nom	Quantité	Description	Fournisseur/Prix	Image
Roue Mecanum	4	Roue du véhicule	Impression 3D	
Rondin pour les roues	24		Impression 3D	
Petite tige	24	Tige liant les rondins et les roues	Usinage	Tige Ø3±0.1 x 48±2
Axe	4	Axe des roues	Usinage	
Clavette	4	Liant mécanique entre les engrenages et les roues	Atelier Micromécanique	Clavette parallèle DIN 6885-2-2-10-A
Roulement à billes	8	Soutient l'axe	Conrad : 0.60 CHF/pcs	Ø6mm int/ Ø12mm ext
Circlip	4	Bloque l'axe en place	Atelier Micromécanique	Segment d'arrêt DIN 6799-4
Engrenages	4x2	Lient les axes avec les moteurs	Impression 3D	
Moteur	4	Moteur 12V 620rpm	Aliexpress : 2.54 CHF/pcs	

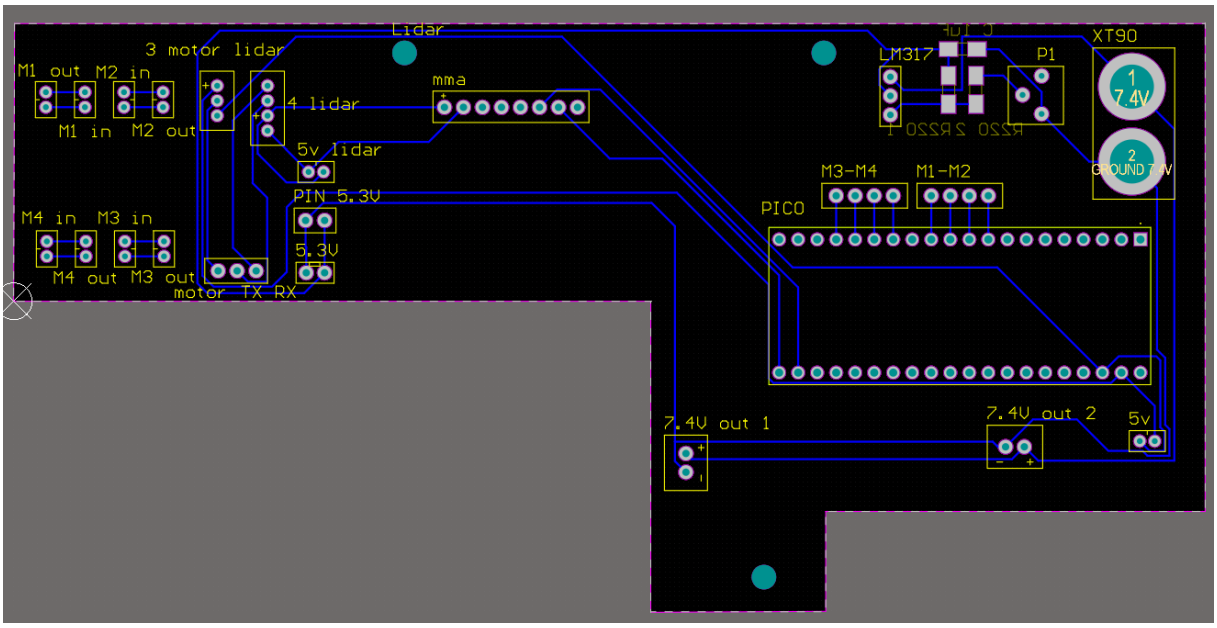
Module L298N	2	Module de contrôle pour les moteurs	Conrad : 6.80 CHF/pcs	
Raspberry Pi PICO WLAN	1	Microcontrôleur pour les moteurs	Farnell : 5.38 CHF	
Raspberry Pi 3B+	1	Microcontrôleur pour le Lidar	Prêt envers l'atelier d'informatique : ≈ 34 CHF	
RPLiDAR DFR0315	1	Les « yeux » du robot	Farnell : 88.86 CHF	
Batterie Conrad Energy	1	Batterie 7.4V 5500 mAh pour alimenter le robot	Conrad : 27.95 CHF	
Bouton	1	Bouton ON/OFF		
Résistance 220Ω	2	Résistances pour le réducteur de tension		

Condensateur 1 µF	1	Condensateur pour le réducteur de tension		
Potentiomètre 500Ω	1	Potentiomètre nécessaire pour régler précisément le réducteur de tension		
LM317T	1	Composant principal du réducteur de tension		
Carrosserie	1	Structure du véhicule	Impression 3D	

4 Fabrication

4.1 PCB

Pour connecter les composants entre eux, j'ai décidé de réaliser un PCB.



Il est entièrement fait en Bottom layer. Mais il faut faire attention car la plupart des composants comme les connecteurs sont traversant, c'est-à-dire qu'il se brase sur la face arrière mais sont utilisé sur la face avant.

Mes différents connecteurs relient mes moteurs, le lidar et ma batterie à mon PCB. Ce sont des connecteurs GST que l'ont peu se fournir chez distrelec et qu'il y'avait en stock dans l'atelier d'électronique.

Le PCB a une forme très spécifique me permettant de l'intégrer parfaitement avec le reste de mes module comme la batterie et les modules sur mon robot.

Les trois trou bleu au centre de mon PCB permettent de fixée le lidar au-dessus du PCB.

J'ai pu le faire imprimer directement dans l'atelier d'électronique équipé avec les machines nécessaires à cette effet.

5 Programmation

5.1 Python et les versions utilisées

Pour réaliser la partie programmation de mon robot, j'ai décidé d'utiliser le langage Python pour sa simplicité, son efficacité et avant tout ses bibliothèques pour entrainer une intelligence artificielle que je détaillerai dans une autre partie.

Même si le cœur de Python version 3.12 que j'utilise reste le même sur chaque appareil, différentes implémentations sont nécessaires afin de s'adapter à chaque spécificités des machines :

- Pour le Pico, j'ai utilisé une version de Python nommé MicroPython. C'est une version allégée du langage avec très peu de bibliothèques préinstallée afin que cela reste léger à utiliser pour le Pico. L'environnement de programmation pour MicroPython s'appelle Thonny et permet d'écrire facilement du code Python ainsi que d'installer des bibliothèques.

- Pour l'ordinateur, j'ai utilisé un environnement dans la programmation Python nommé Spyder. Il permet d'utiliser Python dans toute son intégralité.

- Pour le Raspberry Pi, l'implémentation de Python est plus problématique. Contrairement à Pico, un Pi est un ordinateur à part entière. Il lui faut donc un système d'exploitation afin d'exécuter un programme Python.

  Après une multitude d'essais et de réinstallations, je me suis arrêté sur une version de Linux appelé Raspberry OS Lite, avec laquelle je me connecte avec un portail SSH offrant ainsi l'accès complet au Pi d'un ordinateur ou un téléphone sans devoir connecter un écran ou un clavier.

5.2 La communication entre les appareils

Afin de recevoir les données des différents appareils sur le robot et qu'ils puissent communiquer entre eux, j'ai utilisé le protocole TCP (Transmission Control Protocol). Ce dernier permet à des appareils d'échanger des messages via un même réseau local avec l'adresse IP de chaque appareil en tant d'identifiant.

Pour son implémentation, il faut que l'un des appareils se transforme en serveur et se connecte à l'autre. Par praticité, j'ai choisi comme serveur l'ordinateur.

- Un appareil va s'initialiser en ouvrant le port 12345 afin d'écouter ce qu'il se passe sur le réseau, et se préparer à une future connexion.

```
class communication_robot():
    def __init__(self):
        self.HOST = '0.0.0.0'  # Adresse IP du serveur (0.0.0.0 pour toutes les interfaces)
        self.PORT = 12345 # Port d'écoute du serveur
        self.s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

- Ensuite la fonction connexion va attendre qu'une demande de connexion arrive, identifier si cette connexion provient du Pico (moteur) ou du Pi 3B+ (Lidar) et finalise en indiquant que tous les composants sont connectés.

```
def connection(self,nbr_apareil=2):
    nbr_connection = 0
    self.s.bind((self.HOST, self.PORT)) # Initialise la plage d'adresses et le port d'écoute
    while nbr_connection !=nbr_apareil:
        self.s.listen() # Ecoute si un appareil souhaite se connecter
        print("En attente de connexions...")
        # Accepter une connexion entrante
        self.conn, self.addr = self.s.accept()
        if self.addr[0] == ADDR_LIDAR and self.conn_lidar == None:
            self.conn_lidar = self.conn
            print("Connecté au Lidar")
            nbr_connection += 1
        if self.addr[0] == ADDR_MOTEUR and self.conn_moteur == None:
            self.conn_moteur = self.conn
            print("Connecté au Moteur")
            nbr_connection += 1
    print("Tous les composants sont connectés")
```

Le code des composants est plus simple que celui du serveur :

- Il y a la même initialisation mais cette fois, je rentre l'adresse IP de l'ordinateur pour que l'appareil sache où se connecter sur le réseau

```
class communication_ordinateur():
    def __init__(self):
        self.server_host = '172.20.10.3'  # Adresse IP du serveur
        self.server_port = 12345 # Port d'écoute du serveur
        self.s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

- Puis il va simplement envoyer une requête de connexion et finaliser la communication.

```
def connexion(self):
    print("Connexion ...")
    self.s.connect((self.server_host, self.server_port))
    print("Connecté")
```

- A présent, je n'ai besoin que d'une ligne de code afin d'envoyer une information à un appareil.

```
self.conn_lidar.sendall(str(data).encode())
```

5.3 Le Deep Learning

5.3.1 Définition

Le Deep Learning, ou Machine Learning, est une forme d'intelligence artificielle tournée vers l'essai et l'erreur, simulant l'évolution des espèces vivantes en ne gardant que la plus forte variété de chaque génération.

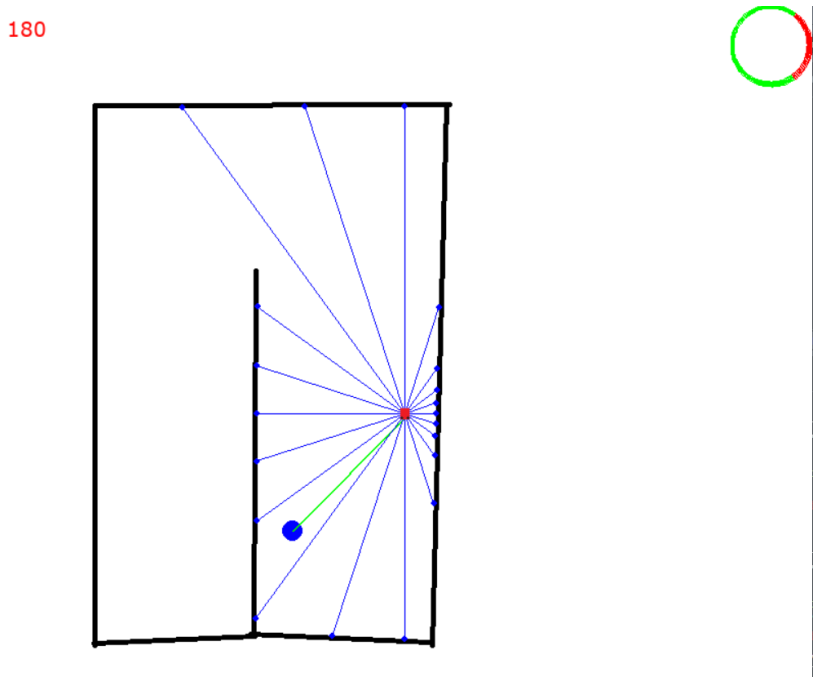
Un réseau de neurones artificielles utilise un environnement d'essai avec un objectif et des récompenses sur le chemin afin de faire comprendre à l'intelligence artificielle qu'elle réalise une bonne action ; ou non.

Un réseau de neurones est composé d'autant de neurones que l'on souhaite contenant autant de neurones que l'on souhaite aussi. Chaque neurone effectue une opération mathématique impliquant un poids unique afin d'atteindre à la fin, une probabilité ; qui correspond au mouvement qui serait le plus adapté au du robot. C'est le poids des neurones qui est modifié à chaque itération en fonction du dernier résultat. Idéalement, après assez d'essais, tous les poids seront parfaitement calibrés à la tâche qui est demandée à l'IA.

5.3.2 Son développement

Afin de préparer le robot et ses déplacements, il fallait que je recrée une simulation de son fonctionnement. Je me suis donc tourné vers la librairie PyGame, une librairie native de Python, qui permet de créer des jeux sur Python, et aussi des simulations utilisables pour un entrainement d'IA.

J'arrive à recréer un environnement basique avec plusieurs cartes évoluant en termes de difficultés. Les trais bleus permettent de visualiser ce que le Lidar verra et le cercle vert permet de dire lorsque le robot est trop proche d'un mur. L'objectif est relié au robot par une ligne verte permettant de connaitre la distance restant à parcourir.



Après avoir développé l'environnement d'essai, il fallait que j'implémente l'intelligence artificielle.

Pour ce faire, j'ai décidé d'utiliser la bibliothèque PyTorch développer par l'entreprise Meta et disponible gratuitement. C'est l'une des bibliothèques d'intelligence artificielle la plus connue et utilisée par les plus grandes entreprises comme NVIDIA, Microsoft ou encore Google Cloud. Cette bibliothèque me permet aussi de mettre un pied dans ce monde qui ne fera qu'évoluer dans les décennies à venir.

Pour démarrer l'entrainement d'une IA, il faut lui donner 3 types d'informations :

- Les informations du robot : sa distance par rapport à l'objectif, son angle de rotation, les informations du Lidar (s'il est proche d'un mur)
- Un système de récompense : après chaque action effectuée, il faut réussir à dire si c'était une bonne action, ou pas. Par exemple, récompenser l'IA si elle s'est rapprochée de l'objectif et la punir si elle touche un mur ; afin de lui faire comprendre son objectif.
- Le catalogue d'action possible : déplacement, rotation, scan du Lidar

Les informations du robot ont été représentées dans le code par une liste d'éléments qui évoluent après chaque mouvement afin de donner toutes les informations possibles :

```
state = [
    danger[0], # 0°
    danger[1], # 45°
    danger[2], # 90°
    danger[3], # 135°
    danger[4], # 180°
    danger[5], # 225°
    danger[6], # 270°
    danger[7], # 315°

    angle_obj, # ° de différence entre robot et objectif
    distance_obj, # distance du robot à l'objectif

    angle_rotation_robot, # angle de rotation du robot
    sur_obj # sur objectif ?
    ]
```

5.3.3 Les problèmes rencontrés

5.3.3.1 Développement de l'IA

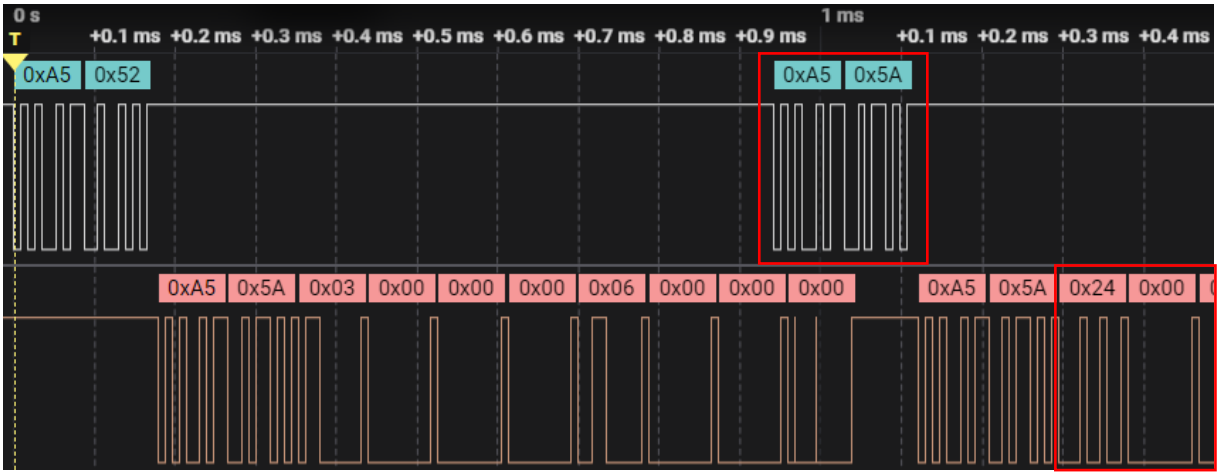
J'ai évoqué auparavant les différentes étapes dans une IA mais je n'ai pas parlé en détail du réseau de neurones et de sa création. Ce dernier est régi par de nombreux paramètres : le nombre de couches et de neurones, mais aussi le pourcentage d'actions aléatoires afin d'apporter des mutations, le taux d'apprentissage lors des essais, les opérations mathématiques entre et dans les neurones et surtout le système de récompense.

C'est ce système de récompense qui est le plus problématique. Lors de son implémentation, il faut trouver le juste milieu entre donner assez de récompenses quand l'IA fait exactement ce qu'on veut d'elle et ne pas trop la punir quand elle essaie quelque chose. De plus, l'IA n'est pas modulable dans sa réflexion ; c'est-à-dire que si je souhaite ajouter un deuxième objectif, cela demanderait de réentraîner le module.

Son implémentation serait donc chronophage, pour un résultat pas forcément à la hauteur de mes attentes.

5.3.3.2 Implémentation du Lidar

Lors de mes essaie de communication avec le Lidar, j'avais une erreur avec mon Pico. Lorsque j'envoyais mon signal depuis le Pico pour par exemple avoir l'état du Lidar, mon Pico renvoyait sans que je lui demande ma demande pendant la réponse du Lidar. Mon Lidar ne comprenait donc plus rien.



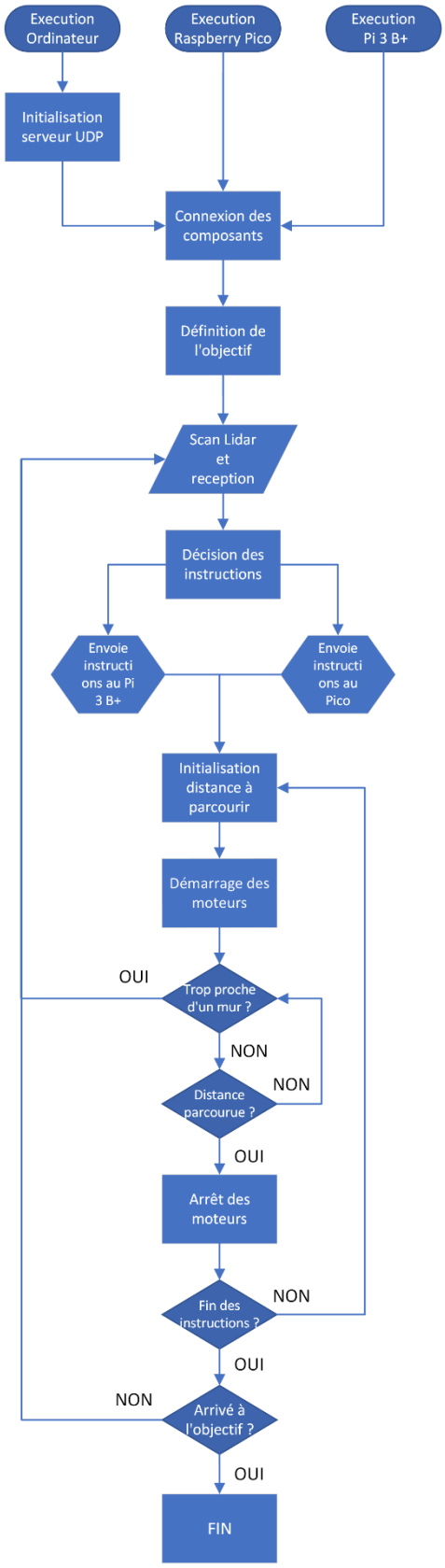
Pour résoudre ce problème, j'ai changé l'OS de mon Pico en sélectionnant une OS plus vieille. Je suis donc passé de l'OS Bookworm à la Bullseye.

Ce problème m'as fait perdre environ 2 semaines de travail. Temps qui m'aurait surement permis de terminer entièrement mon projet.

5.4 La solution algorithmique

Par manque de temps je n'ai pas pu avoir une version définitifs de mon Algorithme.

6 Schéma du fonctionnement



7 Conclusion

7.1 Bilan du projet

L'objectif initial du projet Rodar était de créer un robot autonome capable de se déplacer dans un environnement inconnu et de cartographier les lieux qu'il visite.

- La communication entre les appareils a été correctement mise en place, ce qui permet l'envoi des informations du LiDAR à l'ordinateur permettant de créé une carte des alentours.
- J'ai également initialisé une intelligence artificielle, même si elle ne m'a pas été utile pour l'instant.
- Le déplacement est également possible malgré qui ne soit pas excellent.

Le début du projet a été assez lent à cause de ma formation en usinage et du projet qui n'était pas tout à fait défini, mais une fois les éléments en place, La réalisation du projet s’est accélérée de manière significative.

Cependant, j'ai eu beaucoup de mal à faire fonctionner le LiDAR avec mes appareils, rencontrant des problèmes imprévus qui ont compliqué et ralentit le développement du projet. Ce qui m’a empêché de faire fonctionner de mes modules ensembles correctement.

7.2 Bilan personnel

Durant ce projet, j'ai compris comment implémenter le protocole UCP en Python, appris à créer un jeu avec Pygame et y intégrer une IA.

Mes compétences en modélisation 3D sur Inventor et en conception de PCB sur Altium se sont grandement améliorées, tout comme mes connaissances en électronique en général.

J'ai également appris qu'il y a toujours un moyen de contourner ou de réparer un problème lorsqu'il survient.

7.3 Points d’amélioration

Les déplacements du robot restent médiocres. Les engrenages et les roues en plastique ne sont pas optimaux, et l'utilisation d'acier pour les axes a conduit à des problèmes de rouille. Utiliser des matériaux plus appropriés et durables améliorerait grandement les performances.

En termes de gestion de projet, je n'ai pas rencontré de problèmes particuliers, car j'étais seul à travailler dessus.

Pour l'avenir, je pourrais ajouter un autre robot qui reçoit les informations du Rodar et qui, avec toutes les données de cartographie, serait capable de trouver et réaliser le trajet de façon optimisée et sans problème, renforçant ainsi l'objectif initial du projet.