

DÉPLACEMENT DU ROBOT MECANUM

LOYS FORGET - YVES NGOUAMBEU

PROFESSEUR ENCADRANT : P. MELCHIOR

SOMMAIRE

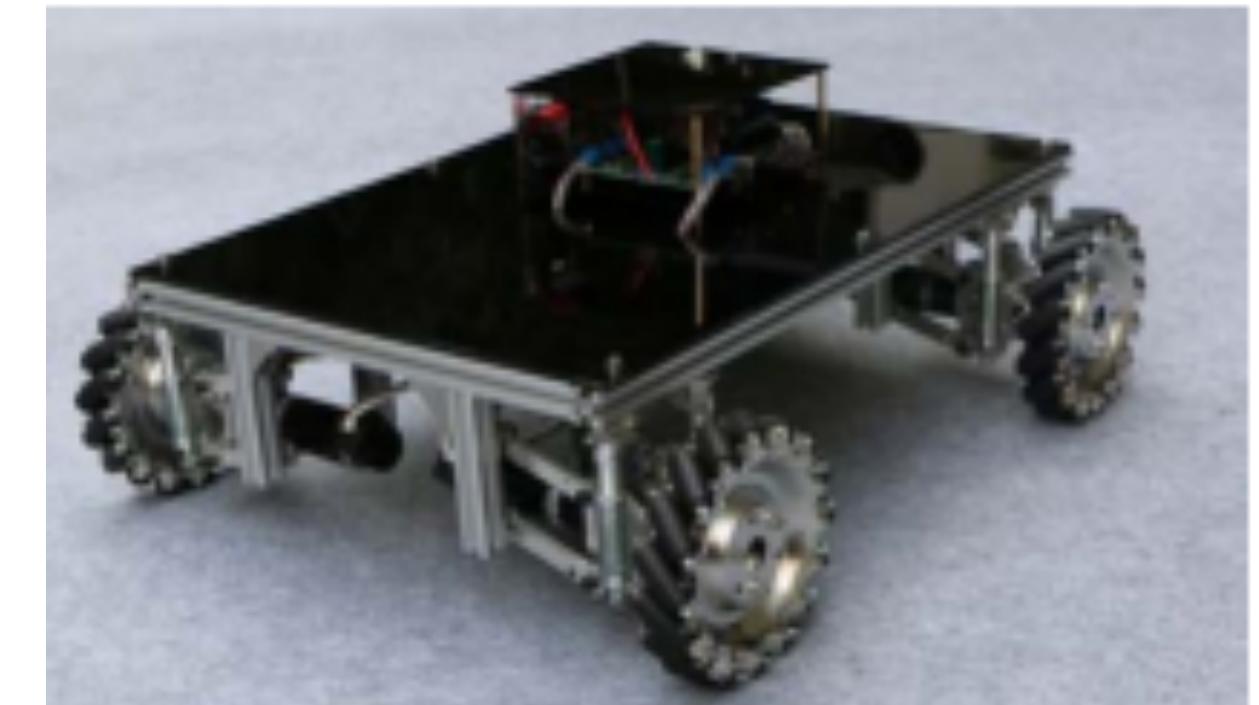
- 1. Présentation du projet**
- 2. Objectifs**
- 3. Principe de déplacement du robot**
- 4. Améliorations du code**
- 5. Interface utilisateur**
- 6. Problèmes rencontrés**
- 7. Support**
- 8. Explosion de la batterie**
- 9. Conclusion**

PRÉSENTATION DU PROJET

- Continuité d'un projet réalisé par un ancien élève
- Déplacement du robot exécuté à partir d'un code Python
- Mise en marche des roues par un code Arduino gérant le pilotage des 4 moteurs (24V 35W)
- Déplacement grâce à un LiDar et une caméra

1. LiDar : cartographie de l'environnement

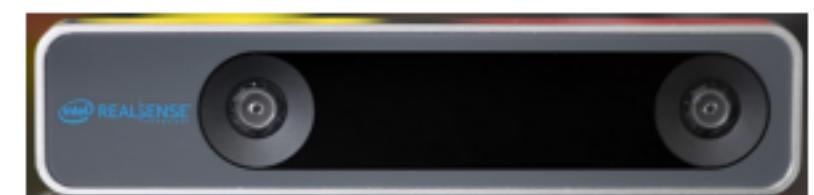
2. Caméra : + la centrale inertuelle



Robot Mecanum à suspensions



Scanner Laser 360° RPLiDAR A2M8



Intel Realsense Caméra Tracking T265

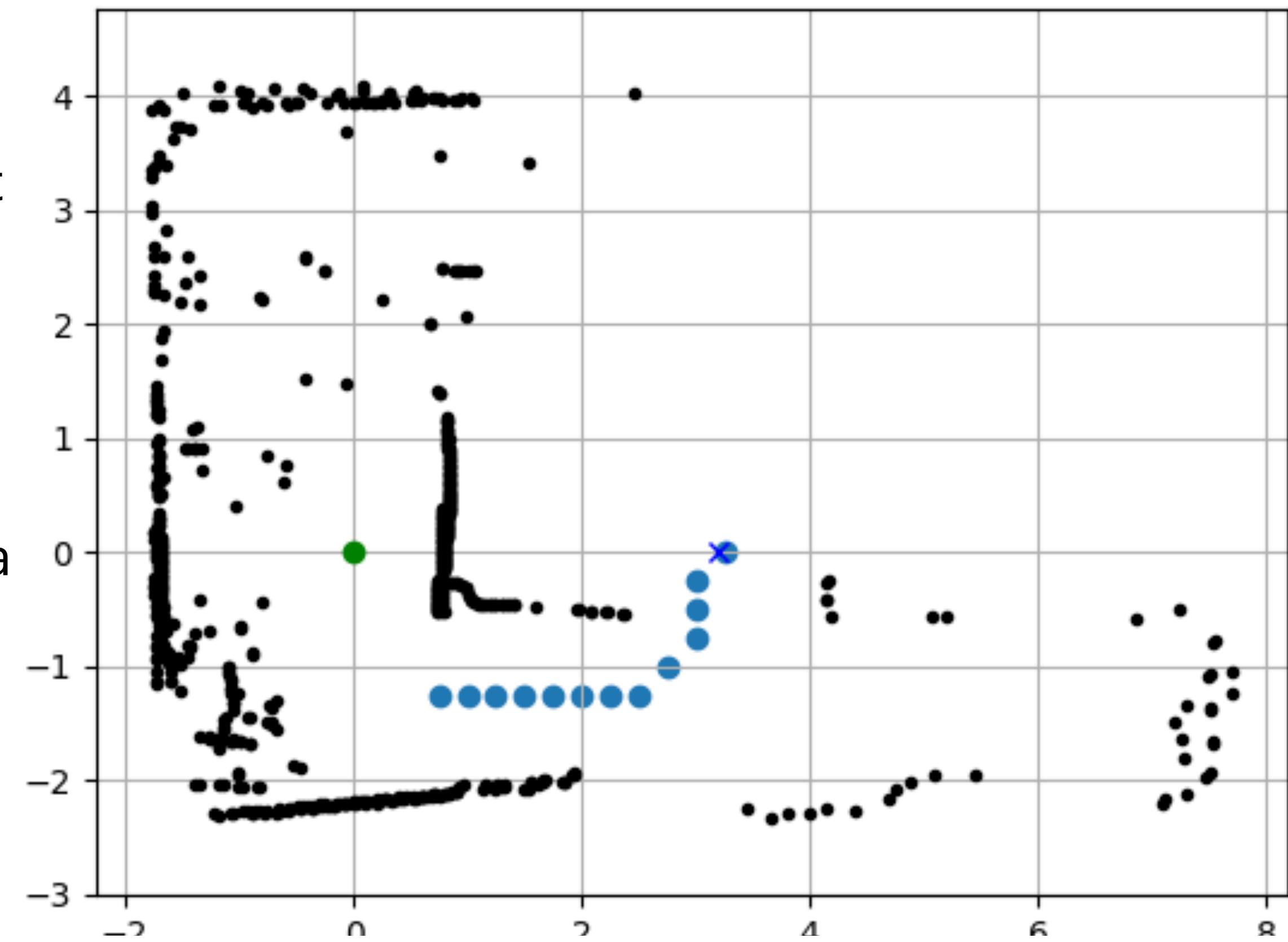
OBJECTIFS

- Appropriation du projet grâce aux dossiers de codes et rapport de l'ancien élève
- Simplifier l'utilisation/compréhension des codes
- Mise en place d'une interface graphique
- Optimisation des algorithmes pour un déplacement en temps réel du robot
- Améliorer le support du robot

PRINCIPE DE DÉPLACEMENT DU ROBOT

1/ MISE EN MARCHE SANS OBSTACLES

- Définition dans le code des coordonnées (2D) du point d'arrivée.
- Mise en route du LiDar qui cartographie 1 fois l'environnement.
- Tracé de la trajectoire à suivre (en bleu ci-contre) sur la map générée.
- Envoie des commandes de déplacements à l'Arduino et donc aux moteurs pour la rotation des roues.
- Suivi de la trajectoire jusqu'au point d'arrivée.



PRINCIPE DE DÉPLACEMENT DU ROBOT

2/ CODE DE DÉPLACEMENT PRINCIPAL

write_read(), update_map(), ObstacleProche(), update_path(), ObjectifAtteint()

```
def run(grid_size, robot_radius, show_animation, move_robot, ArduinoSerialPort, gx, gy, angle_orientation_finale):
    print(__file__ + " start !!!")
    #----Bloc 1-----
    pipe = cm.initcam()

>     if move_robot:
        angle_orientation = 0
        rx_old = 0
        ry_old = 0
        sx = 0
        sy = 0
        xr = 0
        yr = 0
        flag = 0
        Obstacle = False
        ox, oy = [], []
        ...
    while(True):

        #----Bloc 2-----
        """
        On verifie toujours au début si on est arrivé à destination et on arrête le programme si c'est le cas
        """
        if(not(ObjectifAtteint(xr, yr, gx, gy))):


            tmp1 = []
            tmp1 = update_map(pipe, angle_orientation)
            ox.extend(tmp1[0])
            oy.extend(tmp1[1])

>            if show_animation: # pragma: no cover...
                tmp2 = update_path(ox, oy, grid_size, robot_radius, xr, yr, gx, gy, show_animation)
                rx, ry = [], []
                rx, ry = tmp2[0], tmp2[1]

>            if show_animation: # pragma: no cover...



```

```
        #-----Bloc 3-----
        if move_robot:

            #----Bloc 3.1-----
            #envoie de la commande pour déplacer le robot suivant le chemin trouvé
            j=0
            for j in range(len(rx)):


                xr = rx_old*0.001
                yr = ry_old*0.001

        #----Bloc 4-----
        else:


            angle_orientation_finale_str = str(angle_orientation_finale)
            write_read(angle_orientation_finale_str + "zAt", arduino)
            print("Rotation finale: " + angle_orientation_finale_str + "zAt")

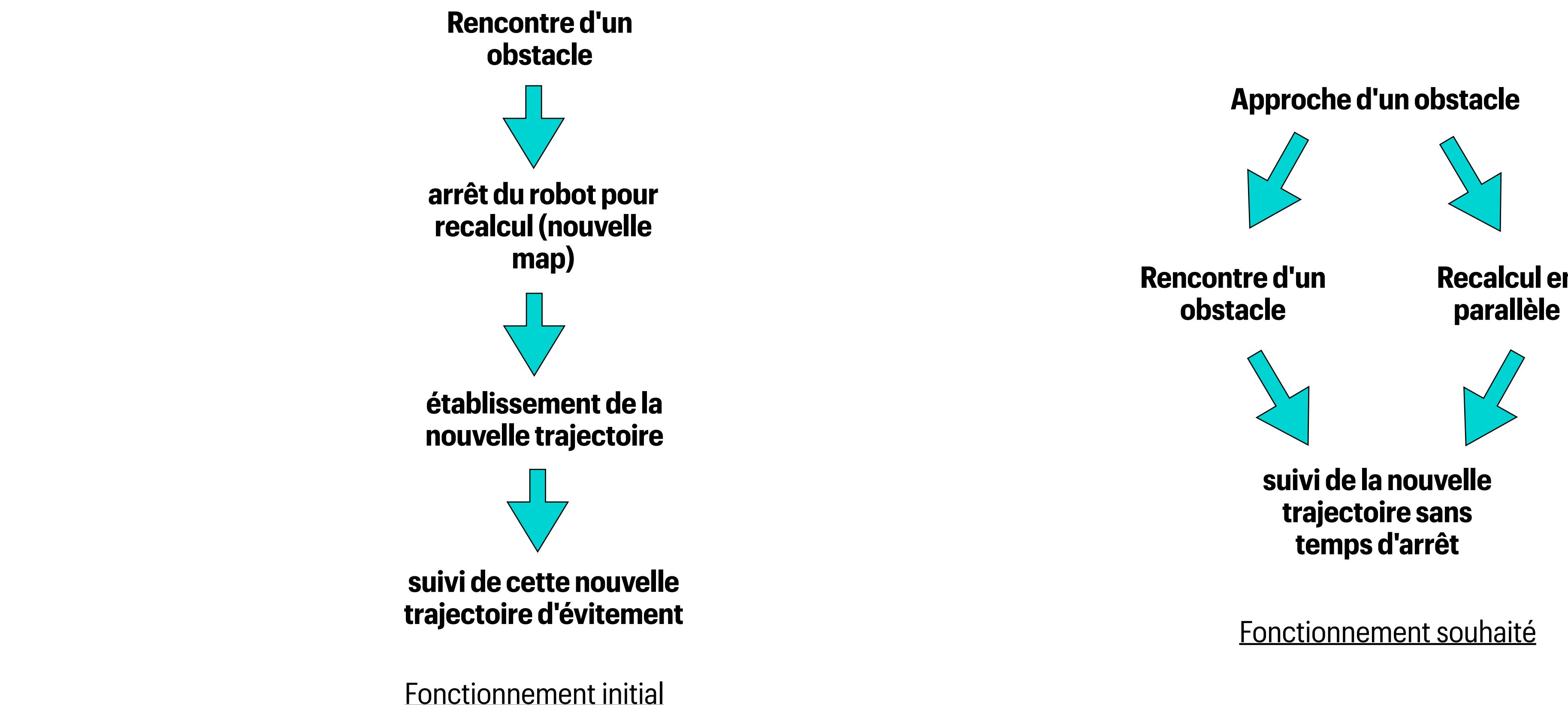
            print("")
            print("Objectif atteint !!!")

            pipe.stop()

            pass
```

PRINCIPE DE DÉPLACEMENT DU ROBOT

3/ ÉVITEMENT D'UN OBSTACLE

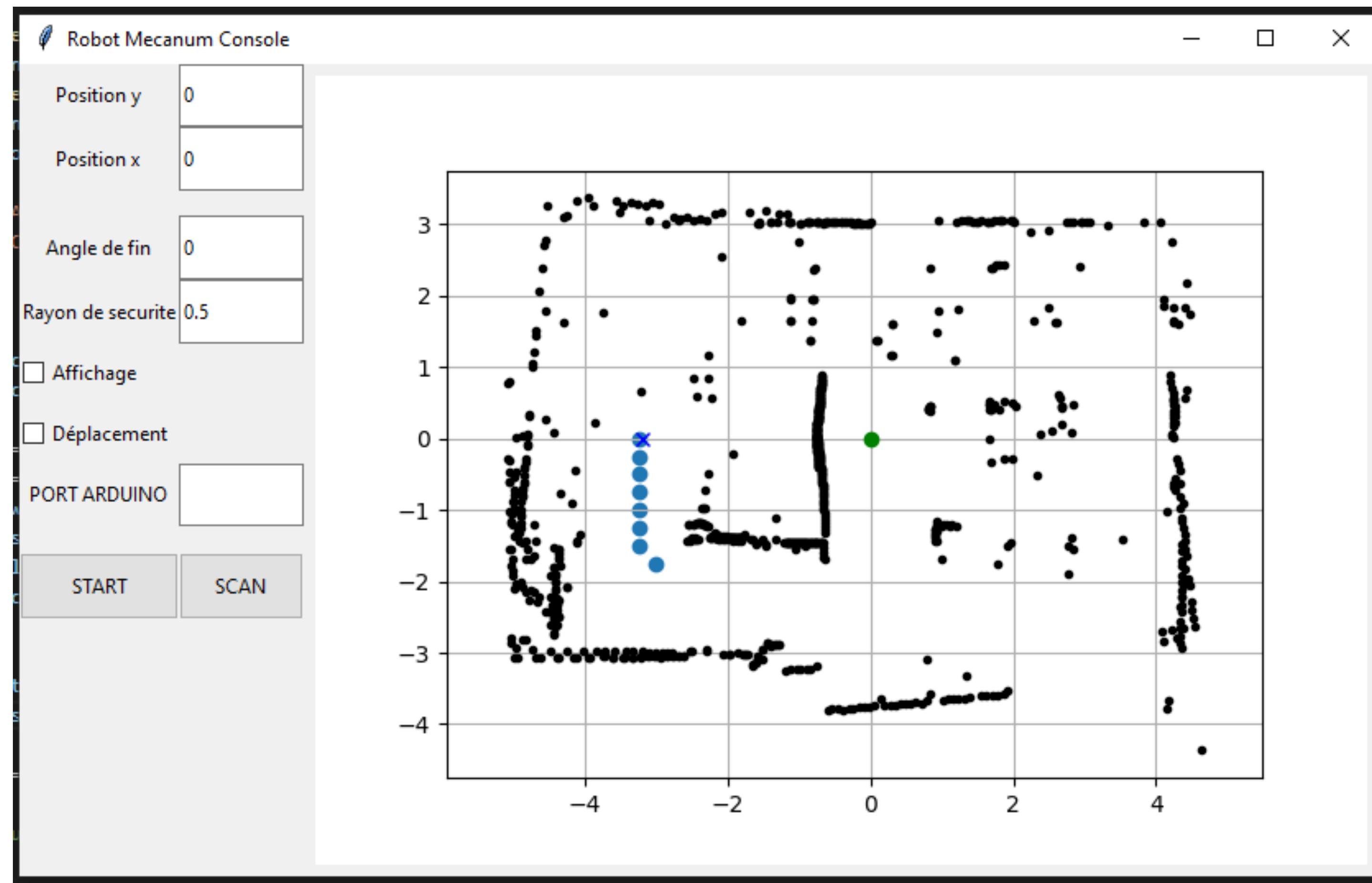


AMÉLIORATIONS DU CODE

- Stratégie adoptée pour pallier les temps d'arrêts : tentative d'optimiser la boucle pour réaliser une cartographie en "temps réel" - mise à jour de la carte avant la fin de la trajectoire.
- Possibilité de modifier le point central du robot en fonction des éléments posés sur le plateau (ex : bras robotisé)
 - ➡ Choix du couple (g_x , g_y) correspondant à la position du LiDar sur le plateau
 - ➡ Utilisation du pipelining et d'un traitement multicoeur (**PAS FAIT**)
 - ➡ Utilisation du language C++ afin d'accélérer le code (**PAS FAIT**)

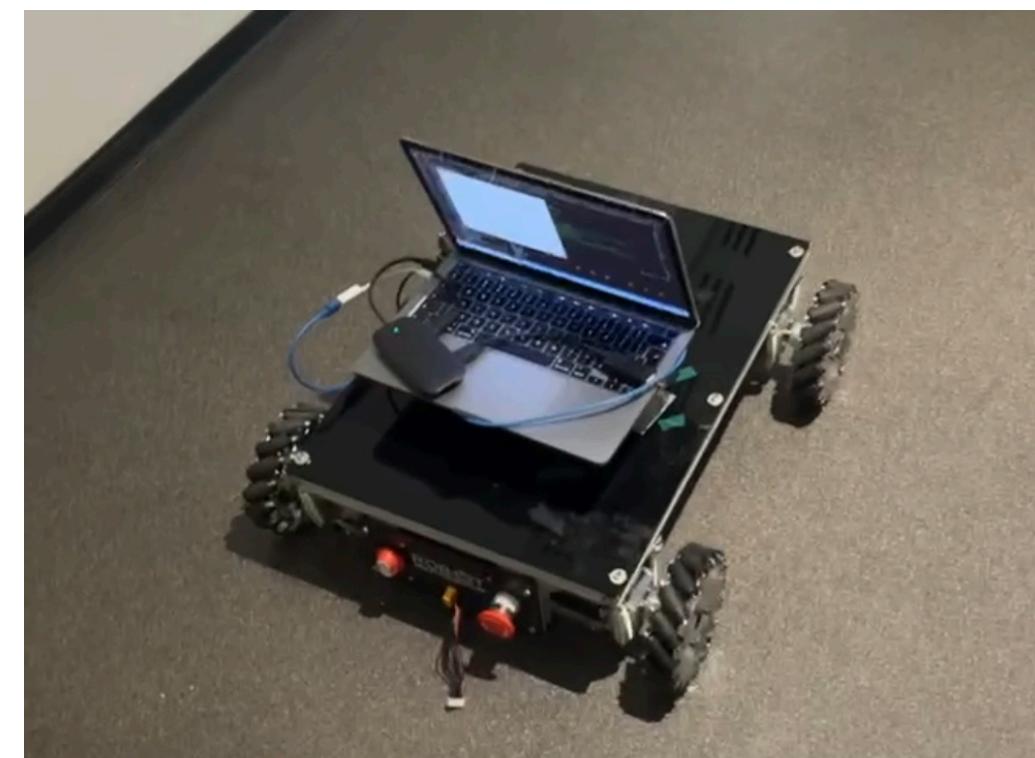
INTERFACE UTILISATEUR

CODÉE EN PYTHON AVEC LA BIBLIOTHÈQUE TKINTER



PROBLÈMES RENCONTRÉS

- Non suivi des trajectoires avec virages par le robot : cherche à réaliser une ligne droite vers le point d'arrivée peu importe la présence de murs/obstacles
- Rotation non synchrone des roues : [vidéo](#)
 - Solution : fil commandant les moteurs mal branché sur la carte Arduino
- PC positionné juste derrière le LiDar sur le plateau empêchant le robot de reculer (le PC est pris pour un obstacle)



Robot en fonctionnement

SUPPORT

IMPRIMÉ EN 3D AVEC L'ULTIMAKER S5 - MODÉLISÉ SUR SOLIDWORKS ET CURA

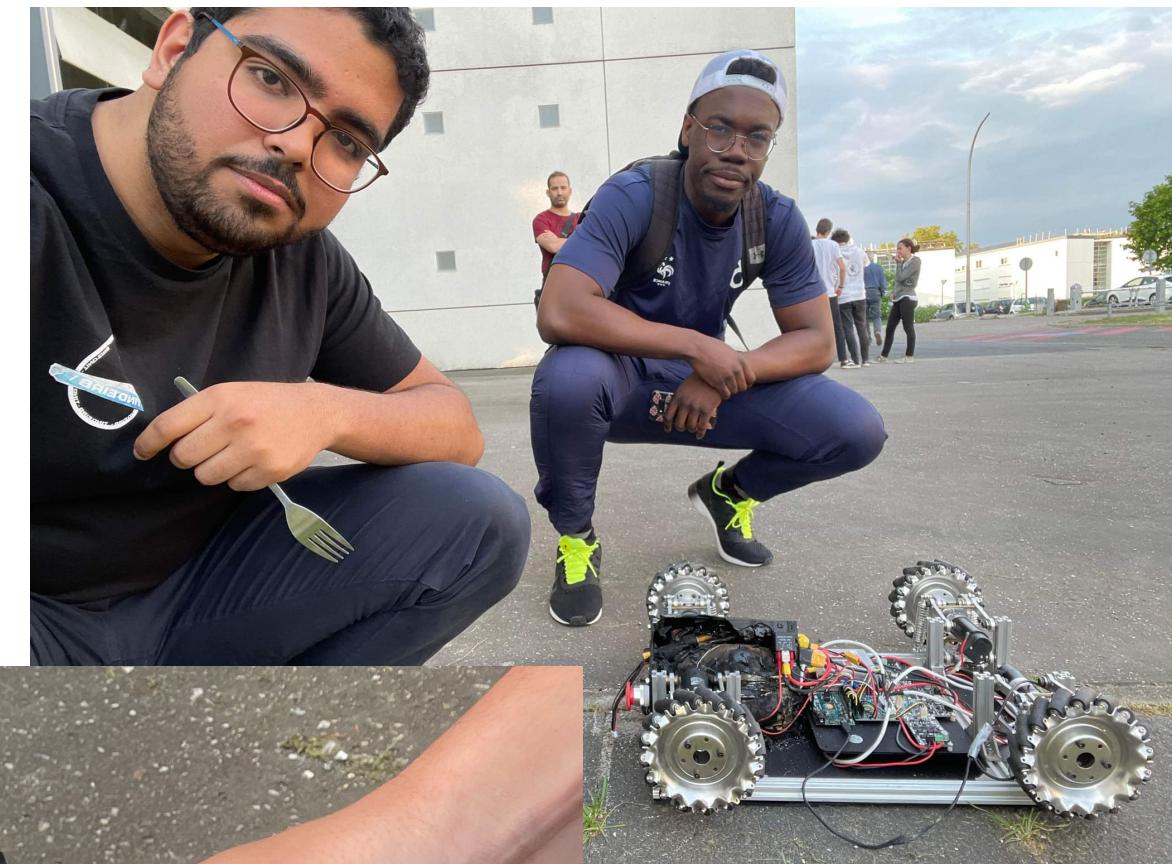
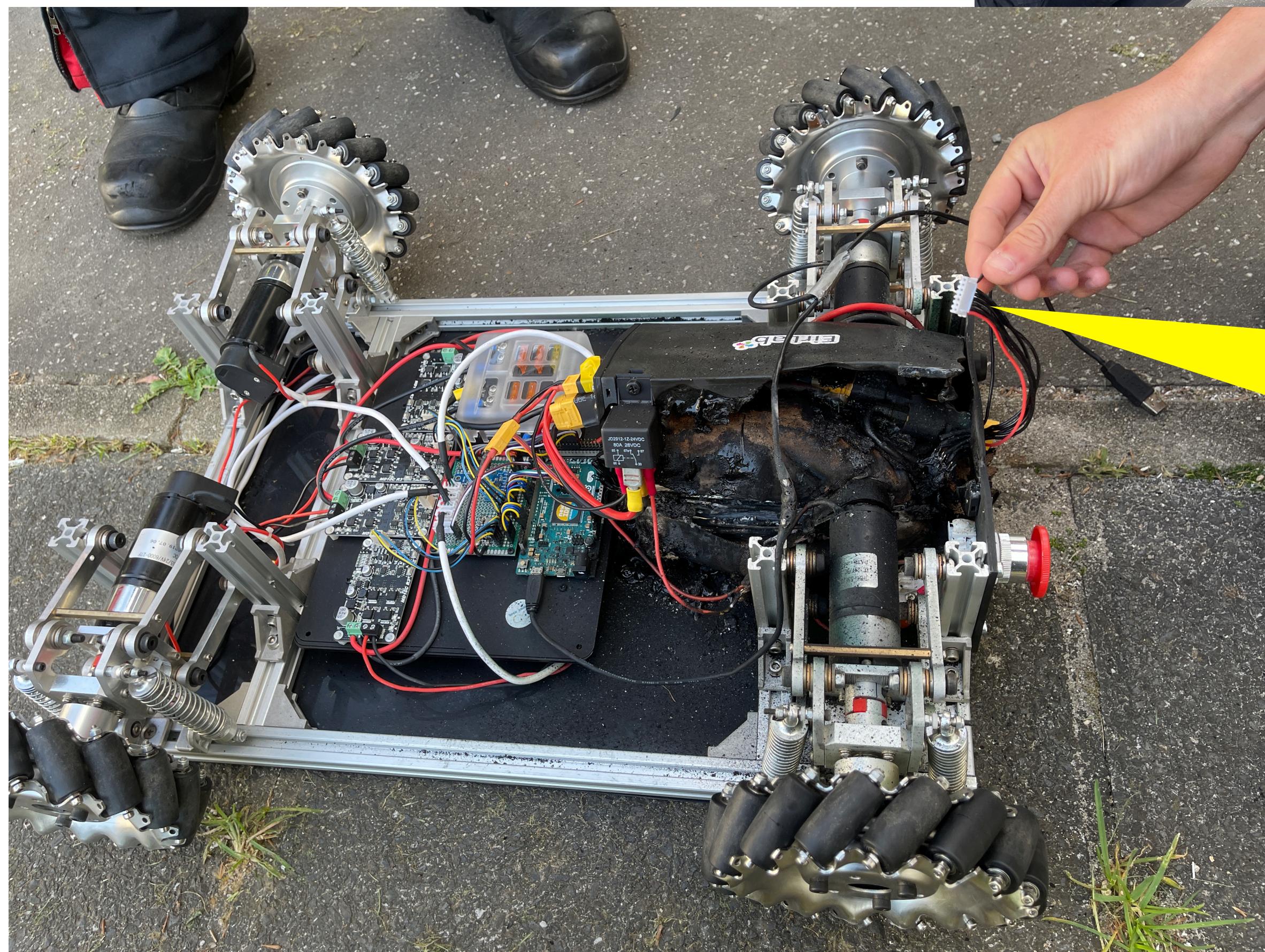
Pour pallier au problème de cartographie à 360° et minimiser l'empreinte du support sur la cartographie 2D générée par le LiDar

→ Confection d'un support afin de surélever le PC

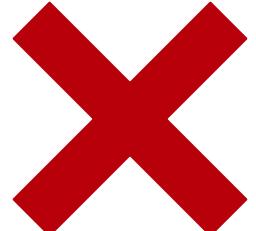
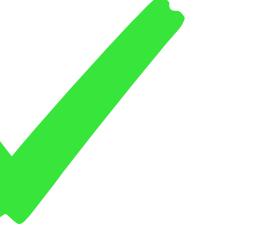


EXPLOSION DE LA BATTERIE

- 03/05 : explosion de la batterie Lipo pour cause de mauvaise technique de rechargement
- Batterie + 2 moteurs arrières endommagés
- Robot renvoyé à Robot-Maker pour réparation



CONCLUSION

- Appropriation du code pour le déplacement et l'évitement d'obstacles avec arrêts 
- Modifications pour optimisation de l'évitement des obstacles en continu 
- Réalisation de l'interface graphique 
- Fabrication du support pour cartographie 360° 
- Participation à la RCup 2023 