POLYTECH NICE SOPHIA

Robotique, Systèmes autonomes

UNIVERSITÉ CÔTE D'AZUR

Rapport Final de Projet

Année Scolaire 2024-2025

"SpyPolyBot"

Eleves :

Hugo DENIS--MARTIN, Brice MABILLE

Polytechnic University of Nice Sophia-Antipolis, robotics speciality (autonomous systems)
930 route des Colles, 06410 BIOT

Sommaire

Introduction

Dans un contexte où la surveillance autonome devient un enjeu majeur pour de nombreux domaines, tels que la sécurité, la logistique ou encore la recherche scientifique, le besoin de robots capables de naviguer de manière autonome dans des environnements devient primordial.

Conscient de ces enjeux, notre équipe a choisi de développer un robot de surveillance autonome dans le cadre de notre projet de troisième année en robotique.

Ce projet présente un robot à quatre roues équipé de capteurs avancés, notamment un LiDAR, un GPS, un laser, afin d'assurer une localisation précise et une cartographie efficace de son environnement.

L'objectif principal de ce robot est de fournir une solution de navigation autonome adaptée à divers environnements :

En extérieur, où les signaux GPS sont disponibles, le robot exploite ces données pour se localiser avec précision.

En intérieur, le robot s'appuie sur son LiDAR pour analyser l'environnement et maintenir une localisation fiable.

Cette approche garantit une adaptabilité optimale, répondant aux besoins d'applications variées de la cartographie d'espaces à l'inspection de zones d'accès.

Le développement de SpyPolyBot s'inscrit dans une démarche d'amélioration de la sécurité, d'optimisation des interventions humaines et de réduction des risques dans des situations dangereuses.

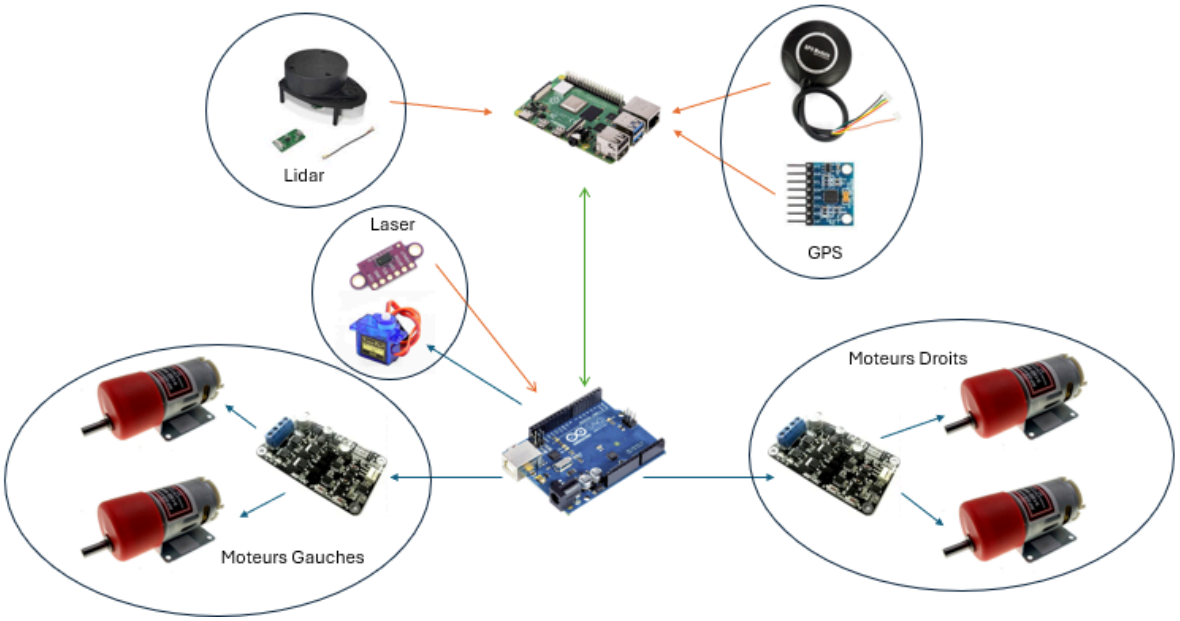
L'ensemble de notre production durant le projet peut être retrouvé dans les deux repository github que vous pourrez retrouver dans la bibliographie.

I. Le Robot SpyPolyBot

A. Description

Le robot SpyPolyBot est un robot de surveillance autonome conçu pour naviguer et se localiser dans des environnements variés.

Nous avons donc construit le robot pour pouvoir l'ajuster à des besoins de surveillance extérieurs et/ou intérieurs. Ainsi voici une description détaillée du robot :



SpyPolyBot possède une base centrale contenant les cartes électroniques Arduino et Rasperry Pi 4 pour la gestion des données :

La carte Arduino assure la gestion des commandes des moteurs ainsi que la connexion avec le laser de détection d'obstacles de proximité.

Elle communique avec la carte Raspberry Pi 4 qui elle s'occupe des principaux capteurs LIDAR et GPS Elle agit comme le "cerveau" du robot. En particulier, elle gère la localisation et la cartographie.

Sur cette même base, nous avons installé toute la partie déplacement du robot : drivers + moteurs.

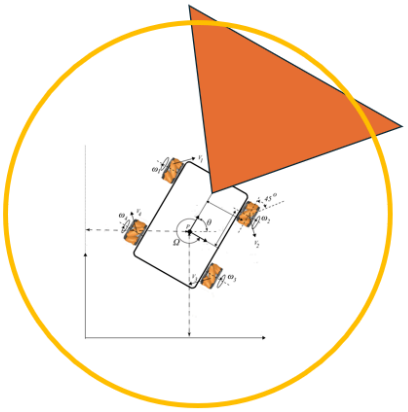
Chaque moteur est contrôlé par des drivers dédiés, qui reçoivent des commandes de la carte Arduino.

Ensuite, au sujet du GPS, un capteur indispensable pour la localisation en extérieur, ce module fournit les coordonnées géographiques du robot. Il est connecté à la Raspberry Pi pour intégrer les données de position au système global de navigation.

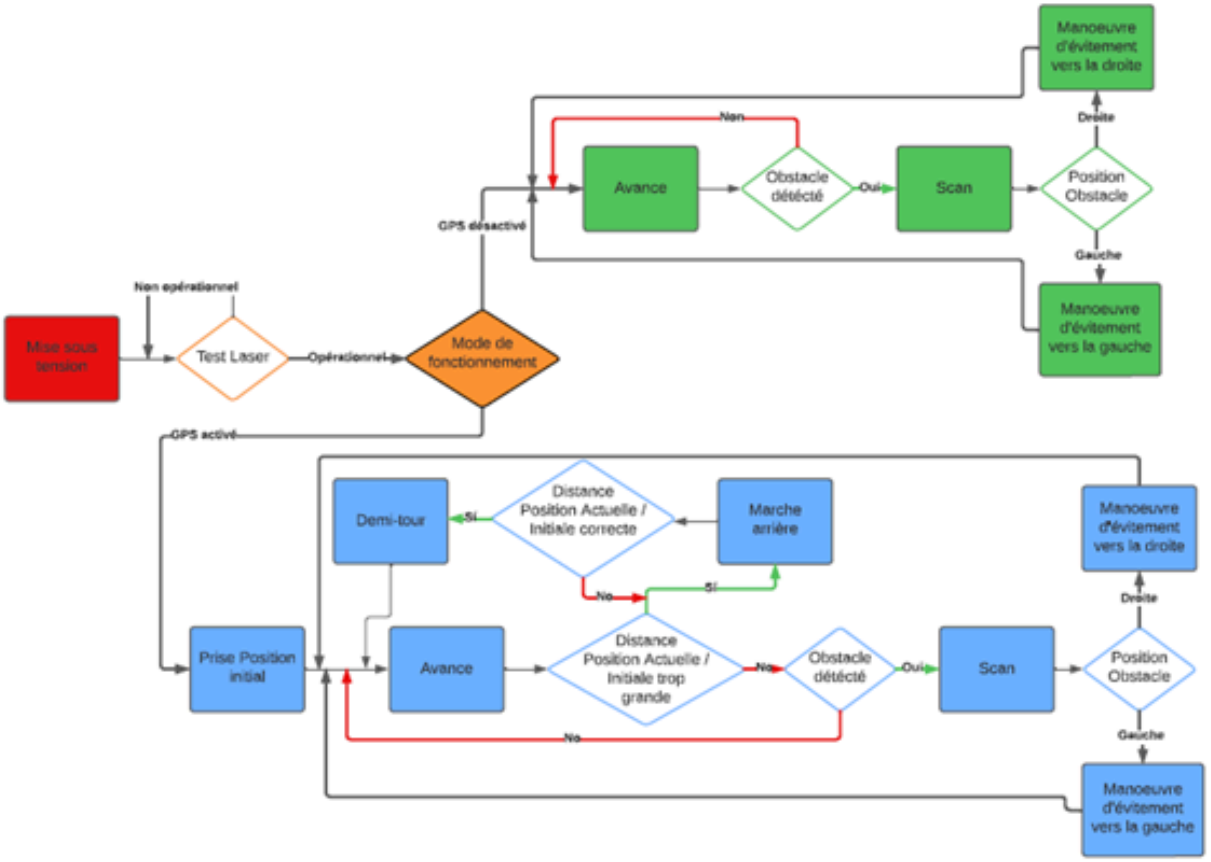
Nous avons ensuite ajouté un étage au robot pour pouvoir installer le LIDAR 2D et qu'il ne soit pas perturbé par d'autres composants.

Nous utilisons ce capteur pour la compréhension de l'environnement du robot utilisé pour la cartographie de l'environnement et la détection d'obstacles en intérieur. Ce capteur fournit une mesure précise des distances en scannant les environs à 360 degrés.

Comme second capteur pour l'environnement du robot, le laser est utilisé comme sécurité pour le robot lorsqu'un obstacle survient dynamiquement proche de SpyPolyBot. Il fonctionne en complément du LiDAR pour affiner les mesures.



B. Algorithme de fonctionnement



C. Limites et Solutions

Le robot de surveillance autonome montre des capacités intéressantes, mais plusieurs limites peuvent freiner ses performances dans des environnements variés. La Raspberry Pi pourrait devenir un goulot d'étranglement pour le traitement des données complexes, comme celles du LiDAR ou des algorithmes de navigation. De plus, le LiDAR seul peut être insuffisant dans des conditions difficiles

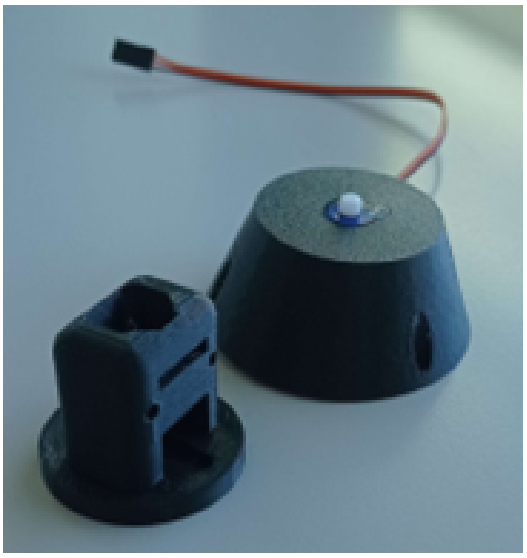
(poussière, surfaces réfléchissantes), et le GPS manque de précision en milieux urbains ou denses. Pour y remédier, des solutions incluent l'ajout d'une unité de traitement plus puissante comme une NVIDIA Jetson, et la fusion de capteurs (LiDAR, caméra, IMU, GPS) via des algorithmes tels qu'un filtre de Kalman.

En termes d'autonomie, l'énergie pourrait être optimisée par des batteries à haute capacité ou des systèmes de recharge. Enfin, pour garantir une fiabilité et une adaptabilité maximales, il serait bénéfique d'introduire une redondance des capteurs, une navigation basée sur l'apprentissage adaptatif, ainsi qu'une interface utilisateur pour la gestion et le diagnostic en temps réel.

II. Capteurs

A. Laser

Le laser est monté à l'avant du robot sur un bras que nous avons conçu et imprimé en 3D, qui est mis en mouvement par un micro servo qui permet au laser de prendre des mesures dans un cône de 120° en face du robot. Nous avons fait le choix de ne prendre que 9 points afin de ne pas devoir ralentir trop le déplacement du robot. Le laser peut détecter des surfaces distantes de 8 mètres au maximum, nous les traitons quand elles sont détectées à 20 cm afin d'avoir la place de manoeuvrer.

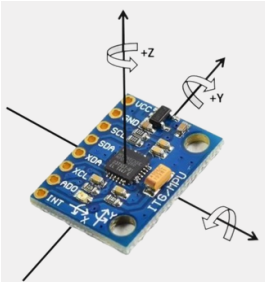


B. GPS - Filtre de Kalman

La précision de la localisation en extérieur est un enjeu dans des environnements dynamiques. L'association d'un GPS à un filtre de Kalman, combiné à un accéléromètre, permet d'améliorer considérablement cette précision.

Le GPS fournit une estimation globale de la position, en effet selon la datasheet du modèle utilisé, il possède une incertitude de 2 mètres. Ce qui est considérable pour un robot de petite taille.

L'accéléromètre va donc nous permettre de compenser ses limitations en fournissant des données sur les variations de mouvement.



Nous avons donc fusionné les deux capteurs pour pouvoir réduire les incertitudes et assurer une localisation fiable.

Pour l'implémentation du filtre de Kalman, nous avons utilisé un vecteur d'état à deux composantes : la position et la vitesse :

$$X_k = \begin{bmatrix} x \\ v \end{bmatrix} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$$

x position statique à un instant

v vitesse dynamique à un instant

Ainsi nous pouvons utiliser l'étape d'extrapolation du filtre de Kalman qui est régie par cette équation :

$$X_{k+1} = \Phi_d \, X_k + G_d \, u_k$$

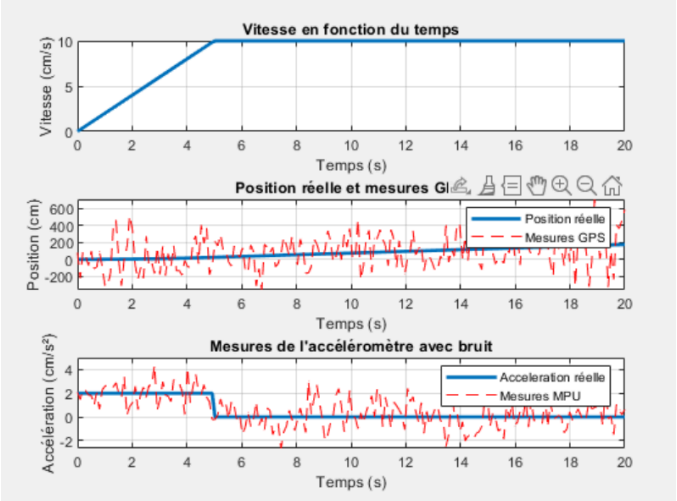
Où u_k = *accélération données par l'accéléromètre en temps réel*

G_d = *Matrice de contrôle* = $G_d = \left(\int_0^{T_s} \Phi(t)dt\right) G = \begin{bmatrix} -\frac{T_s^2}{2} \\ -T_s \end{bmatrix} = \begin{bmatrix} -0{,}005 \\ -0{,}1 \end{bmatrix}$; $G = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

Φ_d = *Matrice dynamique du filtre* = $\Phi_d = \begin{bmatrix} 1 & 0{,}1 \\ 0 & 1 \end{bmatrix} = I + FT_s$; $F = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$

Et X_{k+1} = *Position future estimée* ; X_k = *position réel à l'instant discret k*

D'abord implémenté dans MATLAB, nous obtenons la simulation suivante :

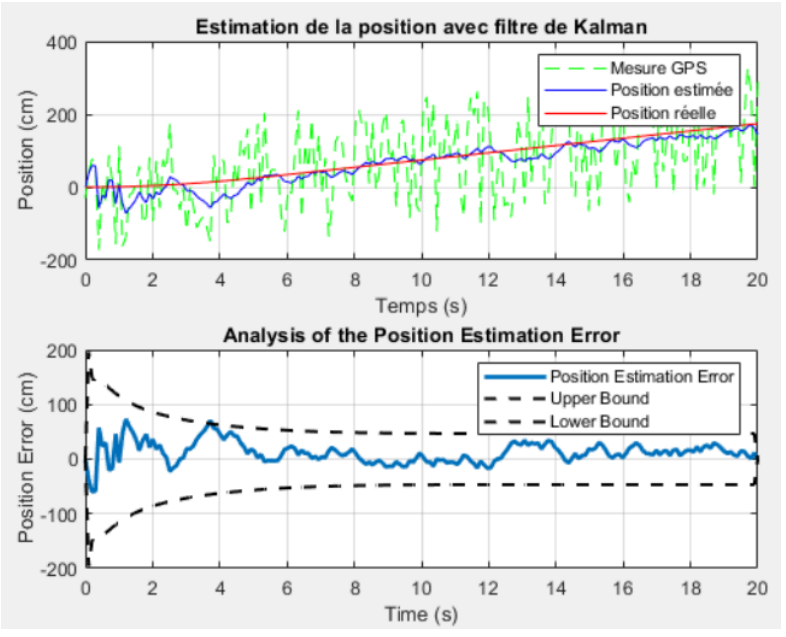


Ces graphiques montrent la simulation du robot avec des mesures de GPS incertaines à 2 mètres près.

En implémentant le filtre de Kalman à ces mesures, et en faisant varier les paramètres du filtre pour équilibrer la partie de confiance envers les prédictions de l'étape d'extrapolation et les nouvelles mesures choisies, nous obtenons les résultats suivants :

Nous remarquons que la position estimée en bleue est bien plus précise que la position récupérée par le GPS.
Nous obtenons une réduction de l'incertitude de 1,5 mètre.
Ceci est une réelle amélioration pour le robot pour pouvoir le localiser en temps réel dans un environnement extérieur.

Sur le second graphique, nous visualisons l'erreur en estimation de la position et nous pouvons remarquer que ces données sont comprises dans les bornes de confiance du filtre de Kalman. Nos données sont donc vérifiées et nous avons pu être sur de l'implémentation sur le robot directement.



C. LIDAR - ROS

L'un des objectifs principaux de ce projet était de nous familiariser avec le LiDAR, un capteur à la fois complexe à mettre en œuvre et largement utilisé dans le monde professionnel.

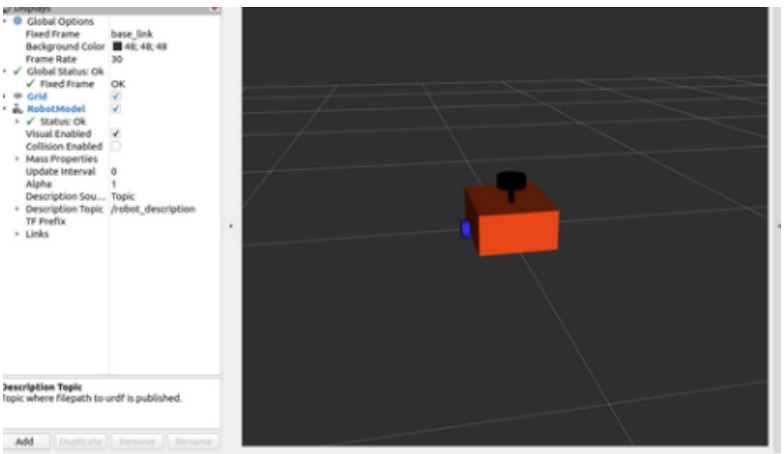
Pour cela, nous devions intégrer le robot sur ROS afin de traiter les données fournies par le capteur. Nous avons opté pour ROS2 Humble, la dernière version stable disponible. Cependant, il nous fallait une carte adaptée pour faire tourner ce système. Dans un premier temps, nous avons essayé une carte Nvidia, mais elle était configurée pour démarrer directement depuis sa mémoire interne plutôt que depuis une carte SD, ce qui rendait impossible son redémarrage. Ensuite, nous avons testé une Raspberry Pi 3, mais elle ne permettait pas d'installer Ubuntu 22.04, version minimale requise pour ROS2 Humble. Finalement, nous avons pu utiliser une Raspberry Pi 4, sur laquelle nous avons réussi à installer tout ce dont nous avions besoin.

Nous avons donc créé un package ROS dans le repository github suivant:

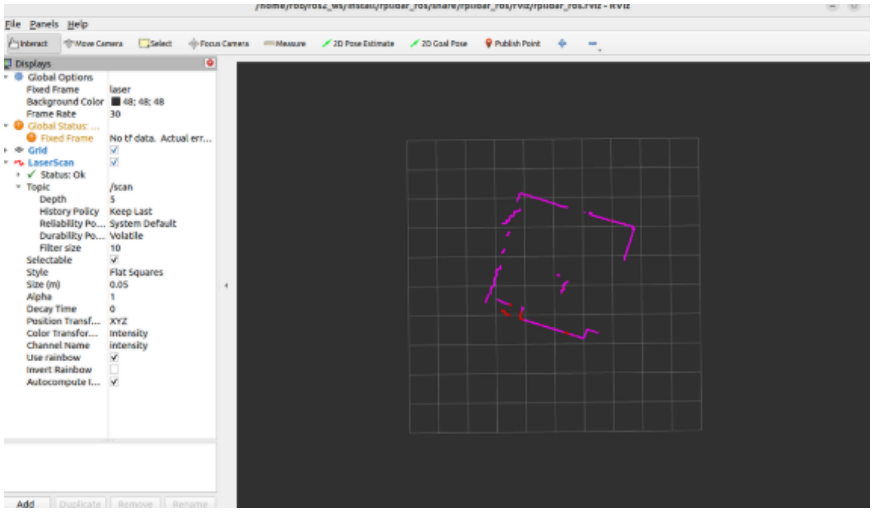
https://github.com/LBlaze911/robot_lidar

```
robot_lidar/
├── description/
│   ├── robot.urdf.xacro
│   ├── robot_core.xacro
│   ├── inertial_macros.xacro
│   ├── lidar.xacro
│   └── gazebo_control.xacro
├── launch/
│   ├── rsp.launch.py
│   └── rplidar.launch.py
└── worlds/
│   └── emplty.world
├── CMakeLists.txt
└── packages.xml
```

Tous les fichiers dans le dossier description/ permettent de décrire le robot afin de le représenter dans ROS, nous avons fait une version simplifié avec le corps, les roues et le lidar, nous n'avons pas essayé de détailler plus que ça le robot car chaque partie du robot sont plusieurs lignes de codes. Nous obtenons donc cette visualisation:

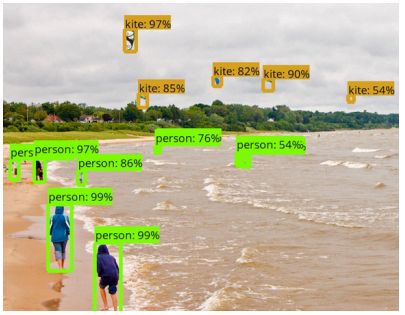


Nous pouvons ensuite brancher le RPLIDAR A1 que l'on nous a fourni et nous avons pu observer cette visualisation des mesures sur ROS, on y observe les murs de la salle et au milieu, une personne qui tient le LIDAR qui est détecté comme un obstacle également. Nous sommes partis sur un LIDAR 2D qui ne visualise que ce qu'il se passe à sa hauteur et non pas un 3D, bien qu'il soit plus performant, mais il coûte plus cher et est plus difficile à intégrer.



III. Perspectives

Bien que le robot de surveillance autonome actuel réponde à ses objectifs de navigation et de collecte d'informations, il existe de nombreuses opportunités pour étendre ses fonctionnalités et renforcer ses performances.



En installant au robot une caméra Logitech couplée à une approche basée sur l'intelligence artificielle, il sera possible de développer un système de reconnaissance visuelle performant. Cette fonctionnalité pourrait être mise en œuvre grâce à OpenCV, une bibliothèque de traitement d'images et de vision par ordinateur.

L'ajout de la reconnaissance d'objets permettrait au robot d'identifier différents éléments dans son environnement, tels que des personnes, des animaux, des objets spécifiques ou même des anomalies à surveiller. Cette fonctionnalité ajoutée rendrait le robot adaptatif à tous types de besoins en matière de surveillance.

Techniquement, cette amélioration impliquerait :

- L'entraînement d'un modèle d'intelligence artificielle sur des ensembles de données pour permettre une reconnaissance d'objets.
- Le développement et l'optimisation des algorithmes de traitement en temps réel via OpenCV.

Cette évolution ouvrirait des perspectives d'applications dans de nouveaux domaines pour la surveillance notamment.

En suivant ce tutoriel : https://github.com/techwithtim/OpenCV-Tutorials ; nous pourrions implémenter cet algorithme de détection d'objets.

IV. Conclusions

Le développement de ce robot de surveillance autonome a marqué une avancée dédiée à la navigation et à la cartographie en environnement complexe.

Au cours de ce projet, nous avons intégré :

- **La navigation autonome** : grâce à la fusion de données provenant du GPS et du LiDAR, le robot est capable de se localiser et de se déplacer efficacement dans des environnements extérieurs et intérieurs.
- **La cartographie dynamique** : en utilisant le LiDAR, le robot peut générer une représentation précise de son environnement.

Ce projet a permis d'explorer des concepts fondamentaux en capteurs embarqués pour la robotique, notamment l'algorithmie pour la localisation et la navigation et la gestion des capteurs.

V. Bibliographie

Project Github
https://github.com/LBlaze911/-Projet-ROB5-2024-2025

Package ROS Github
https://github.com/LBlaze911/robot_lidar

Documentation ROS2
https://docs.ros.org/en/humble/index.html

Rasperry Pi Documentation
https://www.raspberrypi.com/documentation/computers/getting-started.html

Video LiDAR sur ROS
https://www.youtube.com/watch?v=-BObt8inVs8&t=112s

Tuto OpenCV
https://github.com/techwithtim/OpenCV-Tutorials