

# *Projet GSE : mise en place de la navigation autonome d'un robot Turtlebot, modèle burger*

## Introduction

Notre projet consiste en la bonne application et mise en œuvre de 2 tâches principales :

- 1) La cartographie de l'accueil (étage 0) du bâtiment « templiers 2 » de Polytech
- 2) La navigation autonome de notre robot Turtlebot utilisant cette même carte

On observe donc qu'il est nécessaire de passer par l'étape 1 avant de procéder à l'étape 2. Chacune de ces étapes fut parsemée d'embûches et de questionnements, ainsi que de résolutions et de développements dont je tâcherais de vous faire part au mieux ici, dans ce rapport.

L'idée générale du projet est la suivante :

Nous disposons d'un Lidar (indépendant du robot), un type de capteur à technologie laser qui en étant correctement configuré, est capable de prendre des mesures de la distance, pour le notre à une distance d'environ 2 mètres maximum, de la position des objets qui nous entourent. La lumière est émise par le Lidar et se dirige vers sa cible. Elle est réfléchiée sur la surface de cette cible et revient à sa source le Lidar. En fonction du temps de parcours notamment, nous sommes ainsi capables d'en déduire avec une bonne précision quel est l'environnement du Lidar, notamment quelles cloisons l'entourent, et de quantifier ces distances entre les différentes cloisons en mètre.

Ainsi, on peut par l'intermédiaire du logiciel ROS sous Linux être capable d'établir puis de conserver/sauvegarder une représentation 2D de ces mesures et de l'emplacement chiffré de ces cloisons.

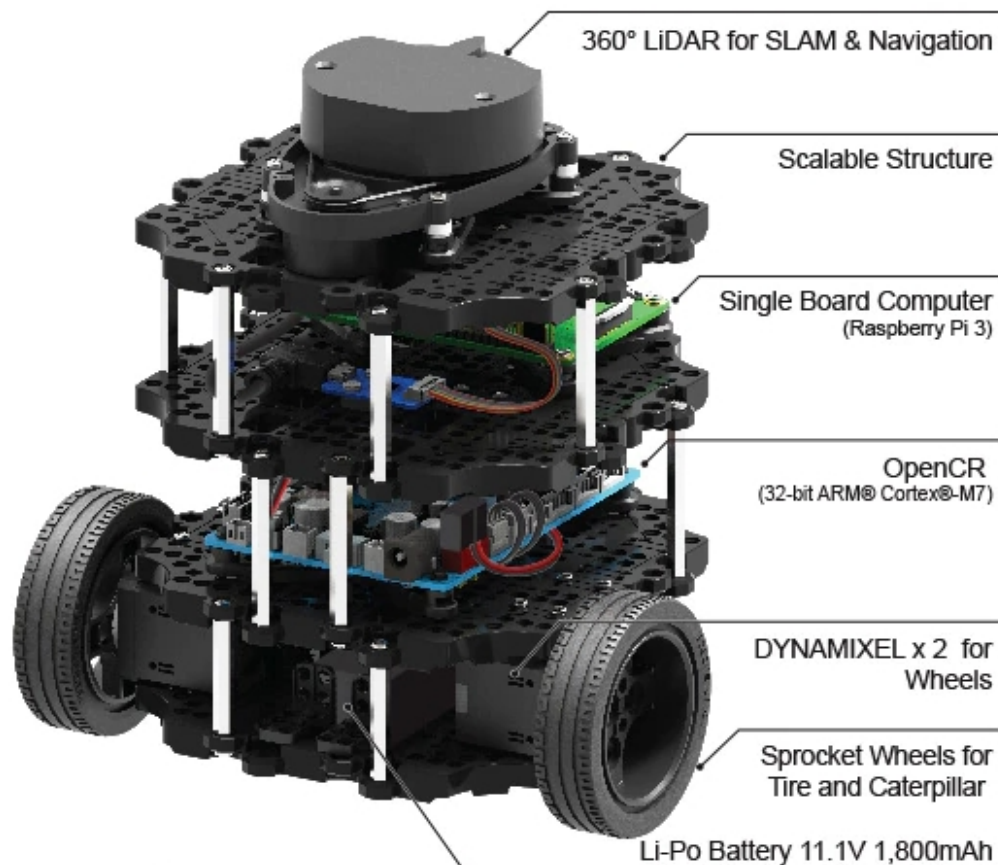
Ensuite, nous pouvons exploiter cette carte, notamment dans un cadre de la navigation autonome où le but serait d'utiliser les informations sur les différentes positions des objets pour savoir de combien se déplacer pour se rendre sur telle ou telle désignation. L'apport de la carte est nécessaire pour l'évaluation des distances mais n'est pas le seul critère. Le robot Turtlebot naviguant en temps réel sera aussi équipé d'un Lidar, qui lui tournera constamment en situation de navigation autonome. En fait, pour tout imprévu, par exemple si une table est mise au milieu du chemin, ou bien dans le cas d'une poubelle qui tombe, le robot sera capable de les détecter et d'adapter son chemin pour atteindre sa destination. Il contournera les obstacles à la fois de la carte pré-fabriquée et des détections faits en temps réel.

Il y a donc les considérations de la carte 2D mis aussi de l'environnement 3D.

## Partie hardware

### A) Construction et mise en place

Au rez-de-chaussée se trouve les roues du robot permettant sa motorisation, au premier étage se trouve une carte arduino reliée à une batterie portable pour une bonne distribution de l'alimentation aux roues du robot, au second étage une carte raspberry intégrant le logiciel ROS (via une carte SD à insérer) et nous permettant de gérer la partie software du projet (soit donner des ordres au robot), et enfin au sommet un Lidar fixé sur le robot, c'est la partie qui va concrètement s'occuper de la détection en temps réel 3D pour compléter la navigation 2D.



C'est nous qui avons construit ce robot de toute pièces.  
Nous disposons aussi d'un Lidar supplémentaire séparément du robot.

## B) Test réel

### B.1) La cartographie

Après quelques manipulations sur le PC (voir pdf), nous montons à l'accueil de Polytech afin de cartographier l'environnement pour pouvoir représenter sur une map 2D le résultat, exploitable par la suite. On tient le Lidar d'une main, le PC de l'autre, les 2 étant reliés par un câble.

La difficulté majeure fut de conserver le lidar bien horizontal et stable afin que les mesures aient peu d'erreur. Pour parer au mieux à cela, nous avons placé le Lidar sur une chaise en hauteur qu'on déplaçait manuellement.

Une autre difficulté majeure rencontrée fut que la présence de vitre sur certains lieux empêchait une bonne cartographie. Nous avons pu obtenir un résultat un peu plus optimal en faisant en sorte que les vitres réfléchissent moins → en cartographiant de nuit.



### B.2) La navigation autonome

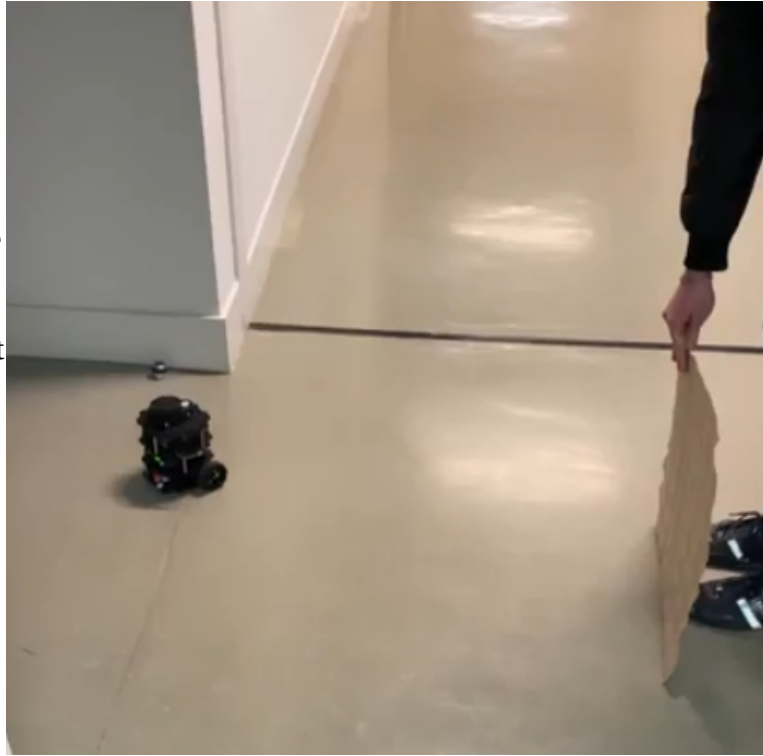
On se connecte au préalable sur une même connexion entre PC et robot (ici la mienne). Après lancement de quelques utilitaires sur les 2 plateformes, on se rend à l'accueil de Polytech pour y déposer le robot.

Sur notre PC, qu'on tient en main, on indique où il se trouve sur la carte puis on lui donne une directive de déplacement.

On observe sur la carte qu'en plus des contours de la map, des encadrés verdâtres apparaissent. Il s'agit des objets détectés en temps réel par le Lidar fixé sur la tête du robot comme nous le disions.

Nous avons fait un test avec un carton placé en plein milieu du couloir, et le robot est bien capable d'adapter sa trajectoire pour atteindre son objectif (emplacement final) sans traverser l'obstacle mais bien en le contournant.

Expérience du robot avec l'obstacle du carton :



## Partie software

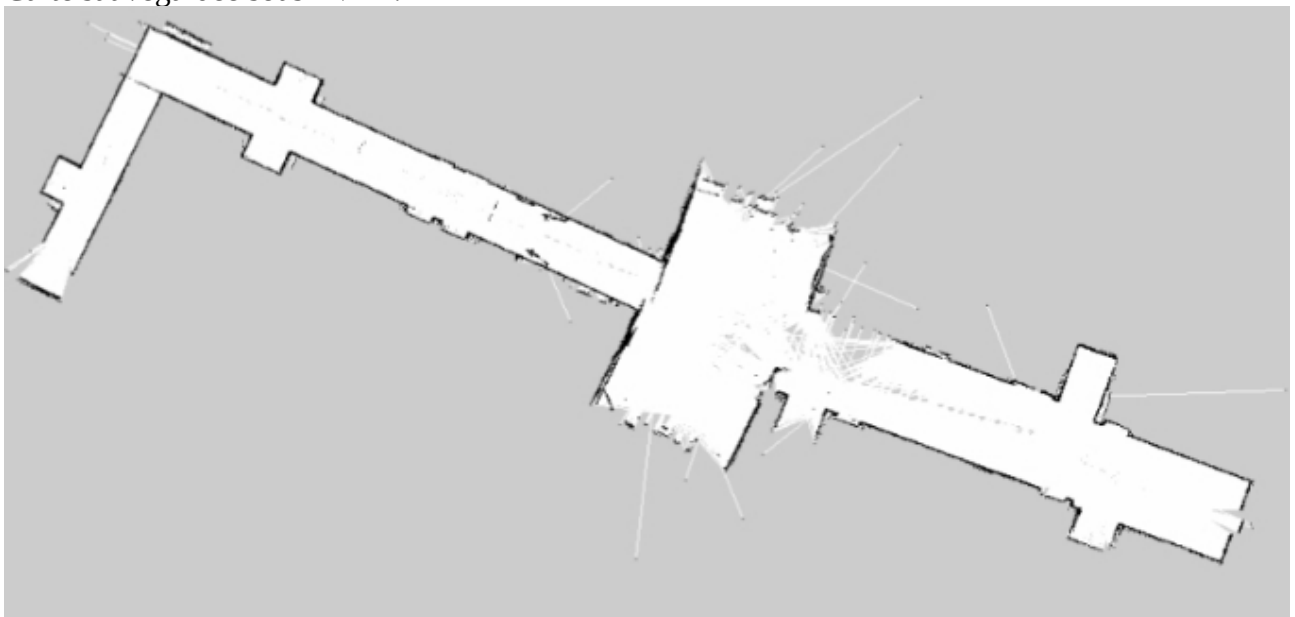
### A) Les maps

#### A.1) La carte cartographiée pour RVIZ

Nous utilisons la librairie hector\_slam et non gmapping, pour plus de simplicité.

À partir de quelques commandes que vous retrouverez aisément dans le pdf adéquat, nous sommes capables d'activer la rotation de notre Lidar et de commencer la prise de mesures, qui fera progresser la carte sur notre PC au fur et à mesure de notre avancement en temps réel. Quand nous sommes satisfaits, on sauvegarde le résultat sous 2 formats .yaml et .pgm, Réexploitable ensuite dans le cadre de la navigation autonome

Carte sauvegardée sous RVIZ :









### A.3) Lien robot/PC

Pour la navigation autonome, au niveau software, nous avons 2 possibilités, la première étant de passer par la manipulation « ssh ». En fait, elle nous permettait d'accéder aux librairies du robot en communiquant avec la raspberry, directement en passant les commandes sur notre PC. Mais comme nous utilisons une version relativement ancienne des librairies ROS, Kinetic, il nous fut plus simple de commander directement le robot à partir de notre PC, avec la librairie Noetic. Pour cela, il nous a suffi de modifier le bashrc en renseignant notamment les informations nécessaires pour le wifi avec les bonnes adresses ip côté master et hôte. Sur la carte SD du robot, nous avons également réalisé l'opération du bringup consistant à mettre à jour les topics et publisher, entre robot/carte SD et PC. Nous voilà maintenant prêts à simuler.

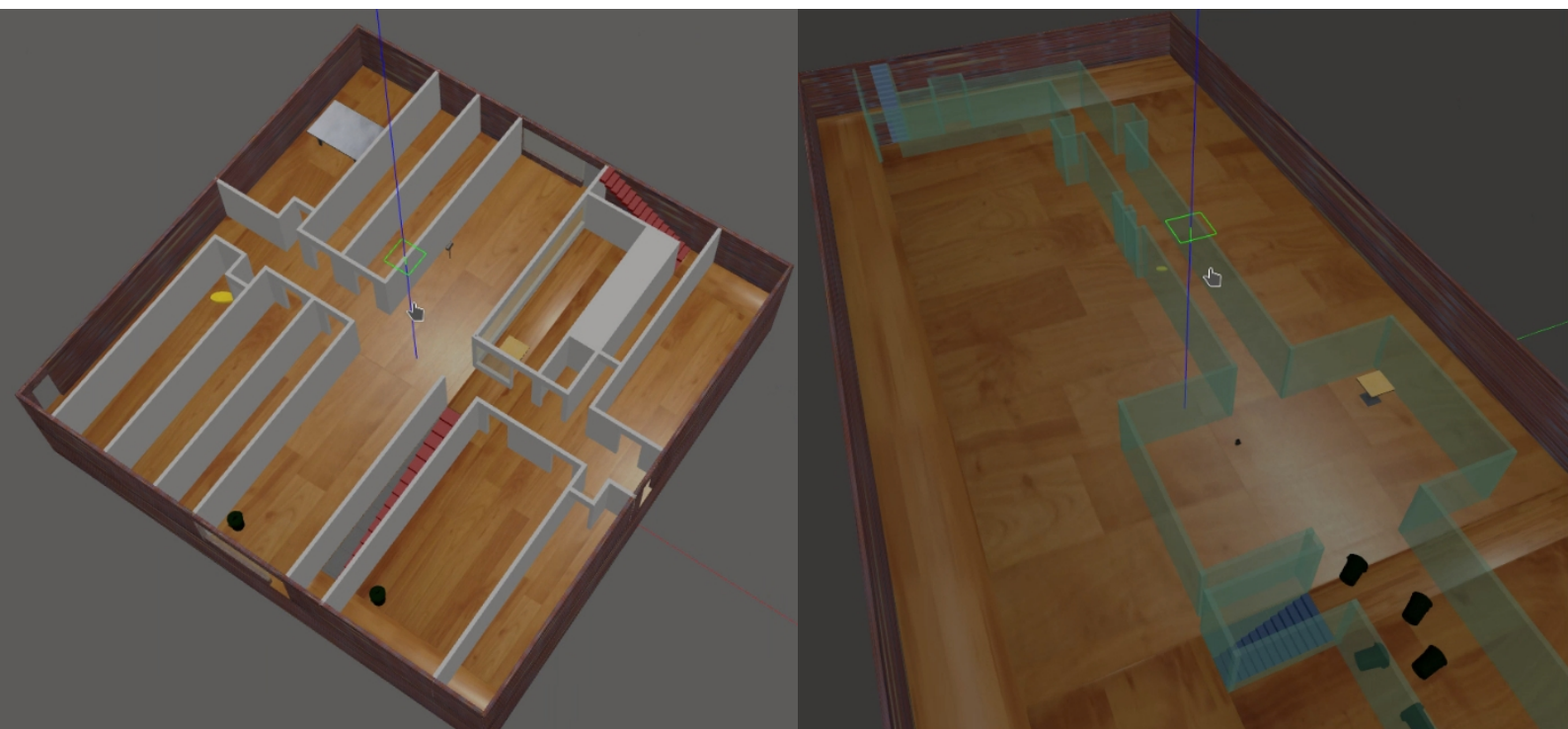
### A.2) La map construite pour Gazebo

Les librairies de ce petit exécutable nous donne déjà accès à 3 cartes de test. Mais nous souhaitons en développer une ressemblant à Polytech pour une meilleure démonstration.

Voici l'échantillon des fichiers principaux qu'il nous faut modifier/coder pour arriver à nos fins, et en particulier le .sdf :

Nom	
	model.config
	model.sdf
	my_map_pierre_nuit.pgm
	my_map_pierre_nuit.yaml
	turtlebot3_house.launch
	turtlebot3_house.world

C'est les cartes Gazebo suivante qui représentent lors de la simulation le déplacement 3D du robot. Versions représentatives de l'accueil de Polytech (à gauche pas à l'échelle) :



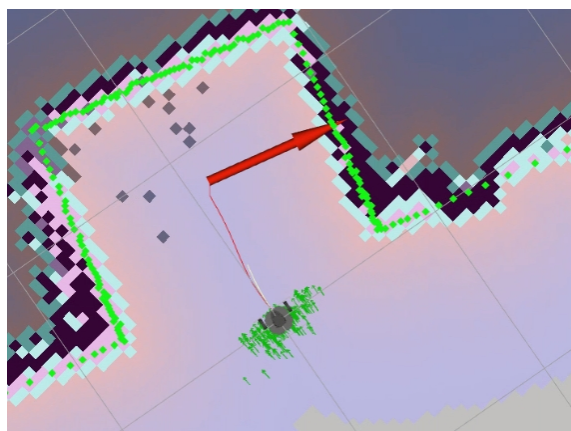
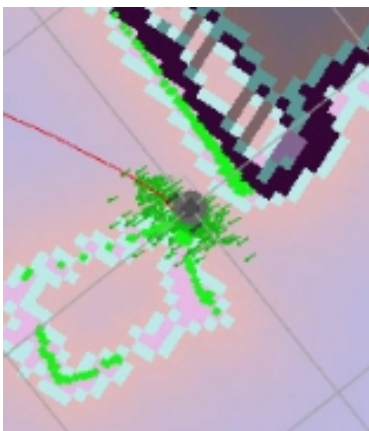
Idée du code que nous avons dû réaliser, en .XML :

```
1 <?xml version='1.0'?>
2 <sdf version='1.7'>
3   <model name='Version_precise_map_gazebo'>
4     <pose>0 0 0 0 -0 0</pose>
5     <link name='Floor_1'>
6       <pose>0 0 2.5 0 -0 0</pose>
7       <visual name='Floor_1_Visual_0'>
8         <pose>-24.7594 0 0.05 0 -0 0</pose>
9         <geometry>
10          <box>
11            <size>0.631175 20.15 0.1</size>
12          </box>
13        </geometry>
14        <material>
15          <script>
16            <uri>file://media/materials/scripts/gazebo.material</uri>
17            <name>Gazebo/Wood</name>
18          </script>
19          <ambient>1 1 1</ambient>
20        </material>
21        <meta>
22          <layer>1</layer>
23        </meta>
24      </visual>
25      <collision name='Floor_1_Collision_0'>
26        <geometry>
27          <box>
28            <size>0.631175 20.15 0.1</size>
29          </box>
30        </geometry>
31        <pose>-24.7594 0 0.05 0 -0 0</pose>
32      </collision>
33      <visual name='Floor_1_Visual_1'>
34        <pose>0.315588 -8.71984 0.05 0 -0 0</pose>
35      <geometry>
```

## B) La simulation

Nous avons fait une simulation à la fois software et hardware. Dans tous les cas le principe est le même : on définit une position de départ et un point d'arrivée sur notre carte 2D RVIZ, puis on constate l'avancée de notre robot. La différence majeure étant que pour une simulation totale nous avons Gazebo en parallèle, qui remplace la visualisation 3D réelle.

Représentation d'un objet rectangulaire initialement absent mais détecté en temps réel par le robot (verdâtre), du robot modèle burger en tant que tel (au centre de l'image) ainsi que des murs de la carte pré-enregistrée (en noir) et le chemin disponible (grisé) :



## Conclusion

Nous avons avec succès réalisé les opérations de cartographie et de navigation autonome de notre robot Turtlebot modèle burger, en témoignent nos démos en vidéo. Notre projet GSE est un succès.