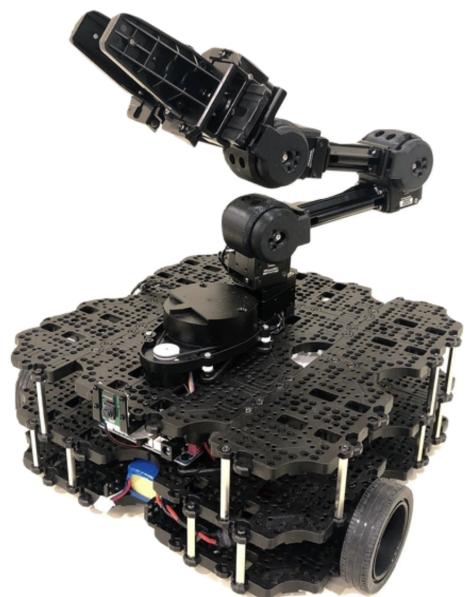


Aaron Amani

Willhem Liban

## Rapport robotique :

### Simulation d'un robot chercheur de balle



# **Sommaire :**

1. Introduction
2. Projet et sous-objectif
3. Choix du modèle de robot
4. Réalisation
5. Résultat
6. Conclusion

## Introduction

Dans le cadre du module d'initiation à la robotique, nous devons réaliser une simulation d'un robot en 3D sur gazebo avec ROS réalisant un scénario précis.

### Qu'est ce que ROS ?

ROS est une plateforme de développement logicielle pour robot. Il s'agit d'un méta-système d'exploitation qui peut fonctionner sur un ou plusieurs ordinateurs et qui fournit plusieurs fonctionnalités : abstraction du matériel, contrôle des périphériques de bas niveau, mise en œuvre de fonctionnalités couramment utilisées, transmission de messages entre les processus et gestions des packages installés.

Le logiciel ROS peut être séparé en trois groupes :

1. les outils utilisés pour créer, lancer et distribuer des logiciels basés sur ROS (roscore, roslaunch, catkin)
2. les clients ROS pour des langages : roscpp (C++) et rospy (python)
3. les packages contenant des programmes pour ROS utilisant un ou plusieurs clients ROS

### Qu'est ce que gazebo ?

En robotique, il est souvent très utile de pouvoir travailler en simulation. Un simulateur physique permet essentiellement de simuler des forces/couples sur des objets et des articulations. Gazebo est un environnement de simulation physique pour robotique, supporté par ROS. Nous nous servirons de la simulation dans Gazebo de manière transparente, "comme si" il s'agissait du véritable robot.

### Cahier des charges du projet :

- ROS Kinetic (conseillé)
- Gazebo 9 (conseillé)
- Robot roulant réalisant un scénario
- Open CV

## Projet et sous-Objectif :

Dans notre cas, nous avons choisi de réaliser un robot chercheur de balle de basket. Passionné tout les deux de basket, une des problématique, quand on joue ou on s'entraîne tout seul sur un terrain, est qu'une fois qu'on réalise un tir, nous sommes obligés de la récupérer le ballon par terre, atterri à X position, et revenir à notre position initiale, au contraire de perdre notre feeling sur le tir, qui peu affecté la qualité de l'entraînement. D'où notre idée de faire un robot chercheur de balle, qui détecte le ballon par terre à X position du terrain, évite les obstacle à jusqu'à la destination, récupère la ballon et le dépose devant le joueur.

Nous avons donc découpé notre projet en sous objectif car c'est un projet ambitieux:

- Objectif 1 : Faire un robot qui évite les obstacles
- Objectif 2 : Faire un robot qui détecte une balle et qui avance vers lui (objet sphérique de couleur rouge/orange)
- Objectif 3 : Faire un robot qui récupère une balle à l'aide d'un bras et le dépose à un endroit

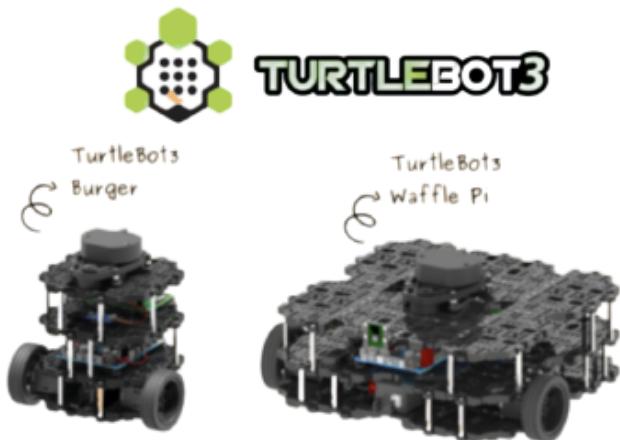
Challenge	
<b>Objectif 1</b>	Evitement d'obstacles
<b>Objectif 2</b>	Détection de ballon de couleur et avancer vers lui
<b>Objectif 3</b>	Prendre un ballon et l'emmener au joueur

## Choix du modèle de robot

Pour l'auto-guidage en évitement des obstacles et la détection de la balle nous avons choisi le turtlebot 3 version burger dans un premier temps, mais pour la détection de balle il y avait un problème avec le topic caméra sur la version burger. Cela nous a couté du temps à résoudre celui-ci et au final nous avons fini par utiliser la version waffle-pi, ou la caméra est totalement fonctionnelle afin de détecter les balles rouges.

Pour l'objectif 3, avec la prise en main du ballon pour le déposer devant le joueur

### Objectif 1 & 2 :



### Objectif 3



## Réalisation :

1. Installation d'ubuntu 16.04 sur machine virtuelle : [lien](#)
2. Installation de [ROS Kinetic](#) et de [Gazebo 9](#)
3. Création des monde pour les objectifs via Gazebo :

Image du monde pour l'objectif 1 :

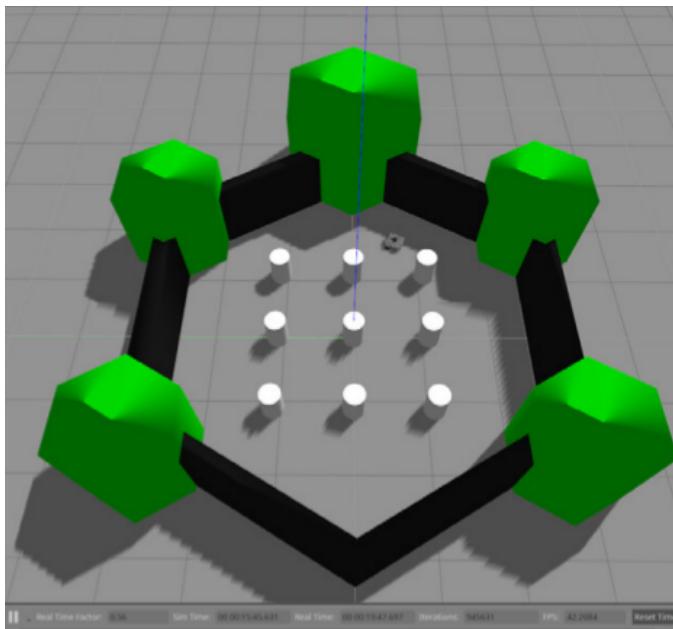
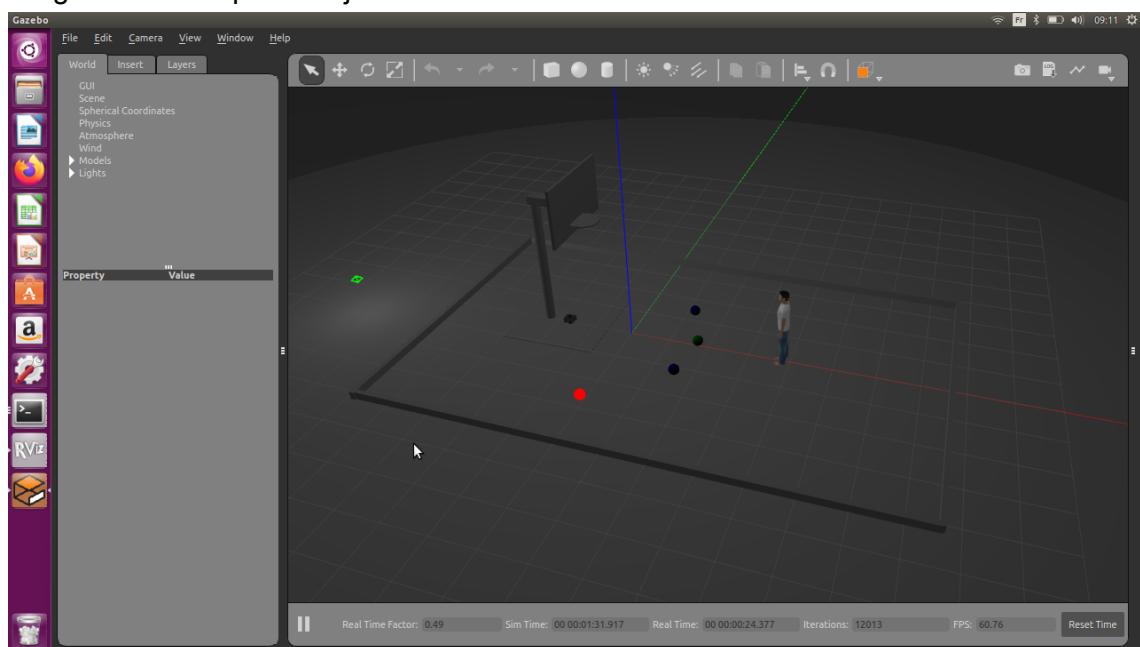
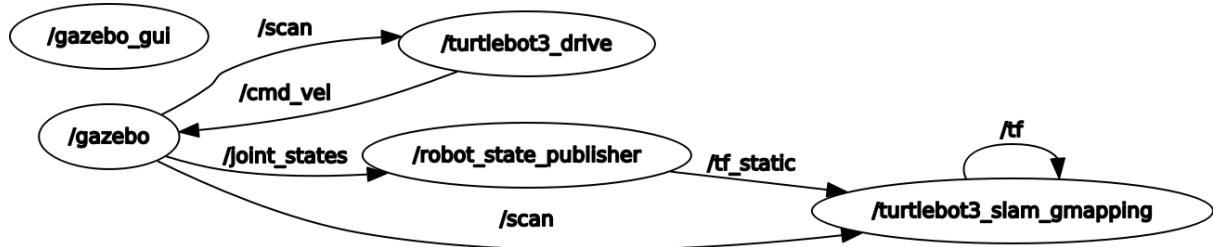


Image du monde pour l'objectif 2 & 3 :



#### 4. Réalisation du robot se conduit tout seul en évitant les obstacle

Rqt graph lors de l'auto guidage :



Commandes utilisé : [lien](#)

#### 5. Réalisation du script avec OpenCV : [lien du tuto](#) et [lien notre script](#)

Image du script basket.launch qui demarre le monde avec Rviz :

```

dio Code
  basket.launch u  chase_the_ball.py u
src > turtlebot3_simulations > turtlebot3_gazebo > launch > basket.launch
  1 <launch>
  2
  3   <include file="$(find turtlebot3_gazebo)/launch/demarerMonde.launch" />
  4   <include file="$(find turtlebot3_slam)/launch/turtlebot3_slam.launch" />
  5
  6 </launch>
  7

```

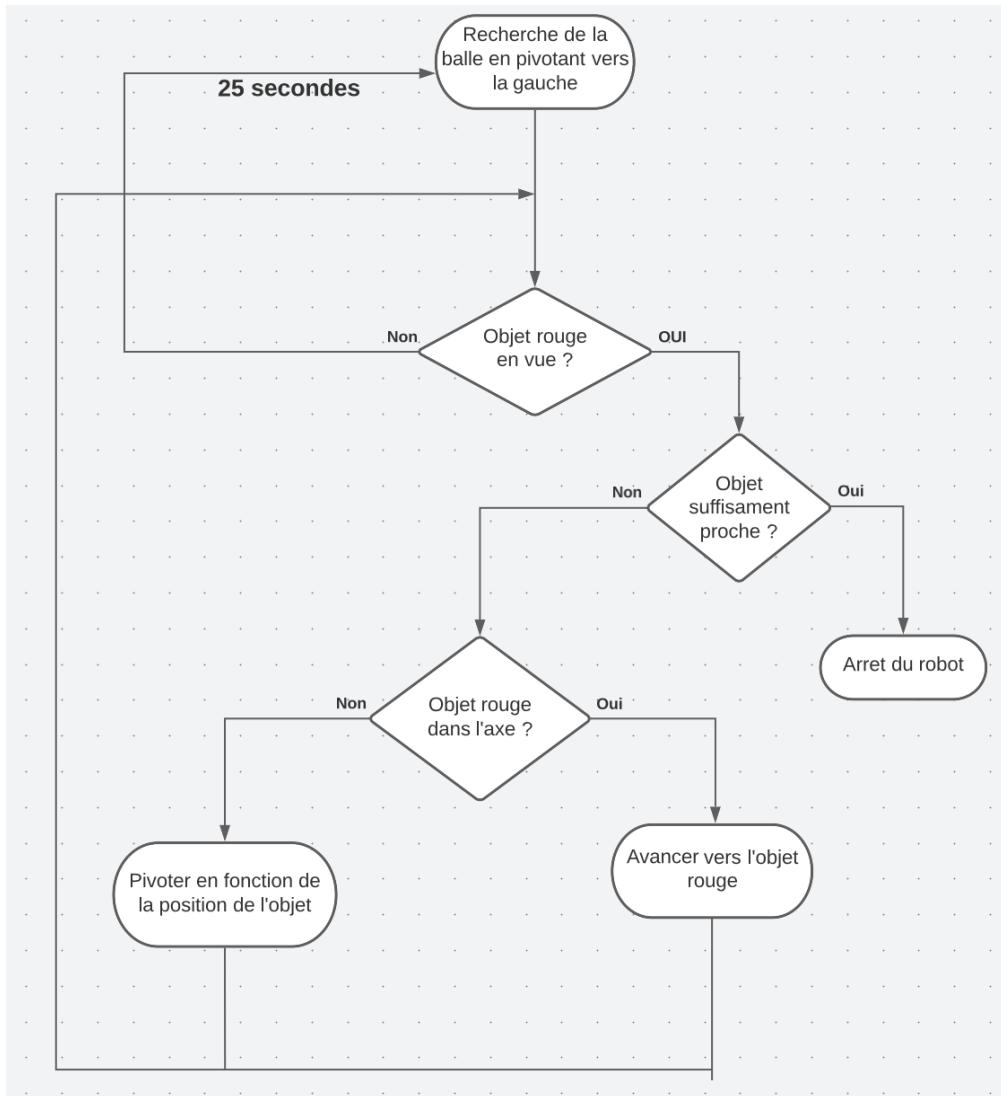
Image du script demarrerMonde.launch qui démarre le monde :

```

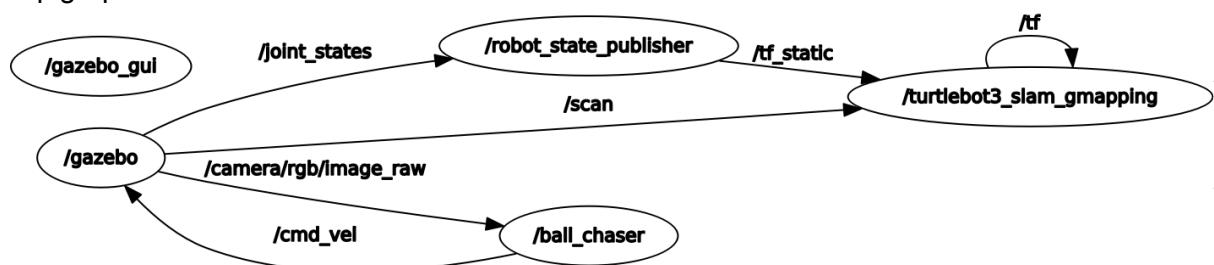
visual Studio Code
  basket.launch u  demarerMonde.launch u  chase_the_ball.py u
src > turtlebot3_simulations > turtlebot3_gazebo > launch > demarerMonde.launch
  1 <launch>
  2   <arg name="model" default="$(env TURTLEBOT3_MODEL)" doc="model type [burger, waffle, waffle_pi]" />
  3   <arg name="x_pos" default="-2.0" />
  4   <arg name="y_pos" default="0.5" />
  5   <arg name="z_pos" default="0.0" />
  6
  7   <include file="$(find gazebo_ros)/launch/empty_world.launch">
  8     <arg name="world_name" value="$(find turtlebot3_gazebo)/worlds/basket.world" />
  9     <arg name="paused" value="false" />
 10     <arg name="use_sim_time" value="true" />
 11     <arg name="gui" value="true" />
 12     <arg name="headless" value="false" />
 13     <arg name="debug" value="false" />
 14   </include>
 15
 16   <param name="robot_description" command="$(find xacro)/xacro --inorder $(find turtlebot3_description)/urdf/turtlebot3_$(arg model).urdf.xa
 17
 18   <node pkg="gazebo_ros" type="spawn_model" name="spawn_urdf" args="-urdf -model turtlebot3_$(arg model) -x $(arg x_pos) -y $(arg y_pos) -z
 19
 20 </launch>
 21

```

Algorithme du script nommée “chase\_ball.py”:



Rqt graph lors de la recherche du ballon :



Lien de la vidéo de l'objectif 1 : [lien](#)

Lien de la vidéo de l'objectif 2 : [lien](#)

## Conclusion :

Par manque de temps nous n'avons pas réussi à réalisé l'objectif 3 mais nous avons réalisé la plupart de nos objectifs.

	Challenge	Résultat
Objectif 1	Evitement d'obstacle	
Objectif 2	Détection de ballon de couleur et avancer vers lui	
Objectif 3	Prendre un ballon et l'emmener au joueur	

Les difficultés rencontrées :

- choix du robot expliqué plus haut
- Projet sous machine virtuelle donc bug, lenteur etc ..
- Nouvelle discipline, Logiciel à découvrir
- peu de temps pour un projet conséquent (2-3 mois)

Amélioration qu'on peut apporter :

- Faire un robot qui récupère une balle à l'aide d'un bras et le dépose à un endroit
- Rapporter la balle à un utilisateur défini
- Rassembler tous les objectifs pour avoir un robot avec toutes les fonctionnalité

