06/07/2020
ROLLAND
Killian

Document de déploiement

TurtleBot3

Table des matières

I. Introduction

Ce document a pour but de permettre à toute personne possédant un TurtleBot3 de déployer les fonctionnalités suivantes :

- Le robot peut se déplacer de manière autonome de sa position vers un endroit voulu ;
- Le robot a une gestion autonome de sa batterie, lorsque celle-ci est trop faible, il va aller à un endroit défini comme sa « base de rechargement » et attendre d'être mis en charge ;
- Une interface Web permettant depuis un ordinateur ou un téléphone de donner des endroits que le robot doit rejoindre, d'avoir le niveau de batterie actuel, d'avoir la vue de la caméra du robot ainsi que des informations affichées en fonction des tâches actuelles du robot et aussi de pouvoir le piloter via un joystick.

Note : requiert l'usage d'un PC comme Master ROS et d'une connexion internet.

Notice d'assemblage du TurtleBot3 :
https://www.robotis.com/service/download.php?no=750

II. Configurations du TurtleBot3

Afin de configurer le TurtleBot3 et le PC, les fabricants (Robotis) ont mis en place un tutoriel sur leur site :
https://emanual.robotis.com/docs/en/platform/turtlebot3/setup/#setup
Nous allons ici voir quels éléments importants de ce tutoriel sont nécessaires au fonctionnement des fonctionnalités citées précédemment.

Sur votre PC, vous devez avoir comme système d'exploitation Ubuntu 16.04 LTS (Xenial Xerus). Ce PC va servir de master ROS et va permettre de contrôler le TurtleBot.

1. Installation de ROS sur le PC

Afin d'installer ROS Kinetic sur le PC, Robotis propose une installation automatisée avec les commandes suivantes :

$ sudo apt-get update

$ sudo apt-get upgrade

$ wget https://raw.githubusercontent.com/ROBOTIS-GIT/robotis_tools/master/install_ros_kinetic.sh && chmod 755 ./install_ros_kinetic.sh && bash ./install_ros_kinetic.sh

Ensuite, il faut installer les paquets ROS nécessaires au fonctionnement du robot :

$ sudo apt-get install ros-kinetic-joy ros-kinetic-teleop-twist-joy ros-kinetic-teleop-twist-keyboard ros-kinetic-laser-proc ros-kinetic-rgbd-launch ros-kinetic-depthimage-to-laserscan ros-kinetic-rosserial-arduino ros-kinetic-rosserial-python ros-kinetic-rosserial-server ros-kinetic-rosserial-client ros-kinetic-rosserial-msgs ros-kinetic-amcl ros-kinetic-map-server ros-kinetic-move-base ros-kinetic-urdf ros-kinetic-xacro ros-kinetic-compressed-image-transport ros-kinetic-rqt-image-view ros-kinetic-gmapping ros-kinetic-navigation ros-kinetic-interactive-markers ros-kinetic-rosbridge-suite ros-kinetic-web-video-server nodejs npm git ssh sshpass

$ sudo npm install -g live-server

Ensuite, redémarrez le PC. Dans un terminal, effectuez les commandes suivantes :

$ cd ~/catkin_ws/src/

$ git clone https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git

$ git clone -b kinetic-devel https://github.com/ROBOTIS-GIT/turtlebot3.git

$ mkdir ~/tempo $$ cd tempo~/

$ git clone https://github.com/KillianRolland/KillianRolland-Navigation-goals-and-battery-monitoring-TB3.git

$ cd KillianRolland-Navigation-goals-and-battery-monitoring-TB3/

$ cp -r battery_monitoring/ simple_navigation_goals/ ~/catkin_ws/src/

$ rm -rf ~/src/turtlebot3/turtlebot3_navigation/
~/catkin_ws/src/turtlebot3/turtlebot3_teleop/

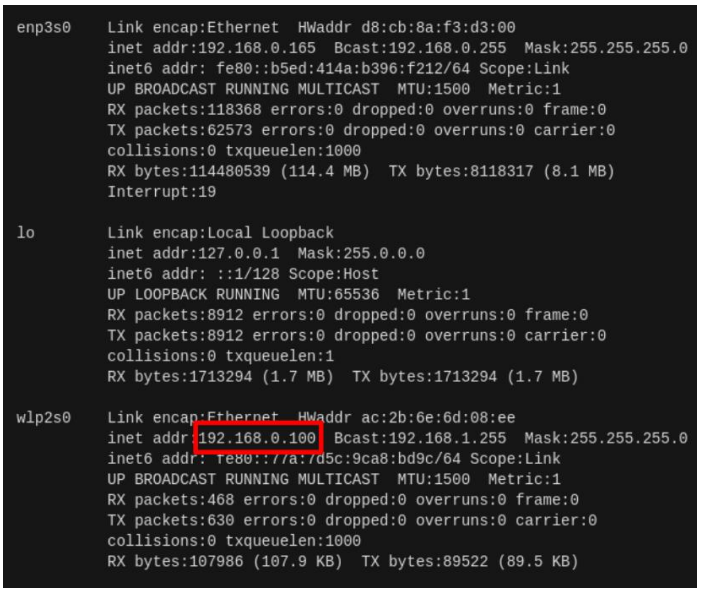
$ cp -r turtlebot3_teleop/ turtlebot3_navigation/ ~/catkin_ws/src/turtlebot3/

$ cp -r turtle.sh Web_App/ ~/ && rm -rf ~/tempo

$ cd ~/catkin_ws && catkin_make

Si la commande catkin_make s'est déroulée sans erreur, alors les installations sur PC sont terminées.

Il faut maintenant configurer le réseau pour que le PC puisse communiquer avec le TurtleBot. Pour cela, dans un terminal du PC entrez la commande suivante : $ ifconfig. Cela va vous afficher un résultat de la sorte :



Selon votre ordinateur/réseau, la configuration ne sera pas forcément la même, mais vous trouverez, comme dans la capture ci-dessus une « inet addr » de type 192.168.xxx.xxx.
Copiez alors cette adresse, puis, à l'aide d'un logiciel de modification de fichier, rendez-vous dans votre .bashrc (ex : $ nano ~/.bashrc). En bas vous devrez trouver deux lignes qu'il faut modifier de la sorte :

```
export ROS_MASTER_URI=http://192.168.1.46:11311
export ROS_HOSTNAME=192.168.1.46
export TURTLEBOT3_MODEL=waffle_pi
```

192.168.1.46 est l'adresse IP de mon PC sur mon réseau Wifi, remplacez cette valeur par l'adresse IP que vous avez copié dans votre ifconfig. Il faut ajouter la 3^{e} ligne ci-dessous afin d'indiquer votre modèle de robot : *export TURTLEBOT3_MODEL=waffle_pi*.

Enfin sourcez votre .bashrc avec la commande : $ source ~/.bashrc

2. Installation de ROS sur la Raspberry Pi du robot

Pour l'installation de ROS sur la Raspberry Pi, Robotis propose un OS préconfiguré (Raspbian Stretch) afin d'avoir tout ce dont il est nécessaire sur la Pi. Pour l'installer, télécharger l'image de l'OS sur le site suivant :

Ensuite insérez la carte SD de la raspberry dans votre PC et rendez celle-ci bootable avec l'image que vous avez téléchargée, via par exemple etcher.io ou Win32DiskImager (uniquement sous Windows).

Ensuite insérez la carte SD dans la Raspberry Pi, puis branchez la à un écran, un clavier et une souris. (Pour configurez le clavier en azerty, commande dans un terminal : $ setxkbmap fr).

Connectez-vous au même réseau que votre PC à l'aide des configurations de réseaux (en haut à droite de l'écran).

Dans un terminal effectuez les configurations suivantes :
$ sudo raspi-config
(Sélectionnez :
7 Advanced Options > A1 Expand Filesystem
4 Localisation Options > I2 Change Timezone > Europe > Paris
4 Localisation Options > I4 Change Wi-fi Country > FR France
5 Interfacing Options > P1 Camera > Yes
5 Interfacing Options > P2 SSH > Yes

Ensuite entrez les commandes suivantes dans un terminal :
$ sudo apt-get install ntpdate
$ sudo ntpdate ntp.ubuntu.com

De même que sur le PC, entrez la commande $ nano ~/.bashrc.
Comme le montre la capture suivante, veuillez dans un premier temps déplacer les éléments propres à ROS en haut du .bashrc (permet d'éviter un bug) puis modifiez les adresse IP. Ici 192.168.1.33 est l'adresse IP de mon PC et 192.168.1.41 est celle de ma Raspberry Pi (toutes deux déterminées avec ifconfig).

```
  GNU nano 2.7.4                                        File: /home/pi/.bashrc

# ~/.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)
# for examples

alias eb='nano ~/.bashrc'
alias sb='source ~/.bashrc'
alias gs='git status'
alias gp='git pull'
alias cw='cd ~/catkin_ws'
alias cs='cd ~/catkin_ws/src'
alias cm='cd ~/catkin_ws && catkin_make'

source /opt/ros/kinetic/setup.bash
source ~/catkin_ws/devel/setup.bash

export ROS_MASTER_URI=http://192.168.1.33:11311
export ROS_HOSTNAME=192.168.1.41
```

Toujours sur la Raspberry Pi, veuillez mettre à jour la carte OpenCR en rentrant les commandes suivantes :

$ export OPENCR_PORT=/dev/ttyACM0
$ export OPENCR_MODEL=waffle
$ rm -rf ./opencr_update.tar.bz2
$ wget https://github.com/ROBOTIS-GIT/OpenCR-Binaries/raw/master/turtlebot3/ROS1/latest/opencr_update.tar.bz2 && tar -xvf opencr_update.tar.bz2 && cd ./opencr_update && ./update.sh $OPENCR_PORT $OPENCR_MODEL.opencr && cd ..

Si le texte [OK] jump_to_fw s'est affiché, c'est que la mise à jour est réussie.

Ensuite, effectuez la commande suivante :
$ cd ~/catkin_ws/src/turtlebot3/turtlebot3_bringup/launch
$ nano turtlebot3_rpicamera.launch

Modifiez la valeur du paramètre nommé 'framerate' à 15, modifiez le paramètre 'width' par la valeur 410 et 'height' par la avleur 308. Ajoutez la ligne suivante en dessous :
<param name= "/raspicam_node/quality" value= "5"/>

Cela donne normalement :

```
<launch>
 <node pkg="raspicam_node" type="raspicam_node" name="raspicam_node" output="s$
  <param name="camera_info_url" value="package://turtlebot3_bringup/camera_in$
  <param name="width" value="410"/>
  <param name="height" value="308"/>
  <param name="framerate" value="15"/>
  <param name="camera_frame_id" value="camera"/>
  <param name="/raspicam_node/quality"  value="5"/>
 </node>
</launch>
```

(Quittez avec CTRL+X et appuyez sur y).

III. Initialisations pour le premier démarrage du système

Eteignez votre TurtleBot, débranchez clavier, souris et écran puis branchez-lui sa batterie (précédemment chargée), puis réallumez le.

Sur votre PC, lancez le Master ROS avec la commande $ roscore.

Ensuite dans un autre terminal, connectez-vous à la Raspberry pi avec la commande :

$ ssh pi@192.168.x.x

où les x sont à remplacer en fonction de l'adresse IP de votre raspberry Pi. Le mot de passe qui vous ets demandé est 'turtlebot'.

(Alternative : $ ssh peut fonctionner).

Dans ce terminal connecté à la Pi, entrez la commande :

$ roslaunch turtlebot3_bringup turtlebot3_robot.launch

1. Utilisation du nœud SLAM pour créer la carte de l'environnement du robot

Il faut ensuite suivre les tutoriels du site et s'arrêter une fois que le partie Navigation est terminée (le SLAM a été fait afin de créer la carte de l'environnement du robot et celui-ci peut s'y déplacer via usage de Rviz).
Il faut ensuite imprimer sur une page le marqueur AR qui est un code qui va permettre au TurtleBot3 de rejoindre son dock de rechargement (fichier « *AR_marker_17.pdf* » et en annexe de ce document).
Il faut aussi faire la partie Automatic Parking Vision (faire le début de la page *Applications* jusqu'à *TurtleBot Follower Demo.* Ensuite faire la partie 15.4 afin de vérifier que le robot va bien rejoindre le marqueur comme prévu :
https://emanual.robotis.com/docs/en/platform/turtlebot3/applications/#applications

Puis dans un troisième terminal, lancez le SLAM :

$ roslaunch turtlebot3_slam turtlebot3_slam.launch slam_methods:=gmapping

Ensuite il faut emmenez le robot à tous les endroits de la/les pièce(s) que vous voulez qu'il enregistre dans sa carte. La carte est visible et se met à jour en temps réel dans la fenêtre rviz.

Pour déplacer le robot vous pouvez soit utiliser la manette forunie avec le robot, soit utiliser le clavier pour le déplacer (dans un autre terminal) :

$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch

Une fois que la carte créée vous satisfait, sauvegardez-la en utilisant la commande suivante dans un autre terminal :

$ rosrun map_server map_saver -f ~/map

(Vous pouvez modifier la dossier de destination, qui est ici le home : ~/ et le nom de la carte qui est ici « map »).

Dans le fichier turtle.sh il faut modifier l'adresse de la Rapsberry Pi (la mienne étant pi@192.168.1.41, il faudra mettre l'IP correspondant à votre Raspberry Pi sur le réseau).
Il vous faudra aussi remplacer le nom de la carte ainsi que l'endroit où est stocké le fichier yaml correspondant (En bleu : à adapter) :
$ gnome-terminal -x roslaunch turtlebot3_navigation turtlebot3_navigation.launch map_file:=$HOME/map.yaml &
Pour la dernière commande, il faut aussi changer le répertoire du dossier Web_App/ importé depuis GitHub :
--working-directory=Web_App/ -x live-server . --port=5500 --no-browser &
Afin de rendre ce fichier exécutable, il faut aller dans le répertoire où il se trouve (ou alors le déplacer vers un autre répertoire voulu) puis rentrer dans le terminal :
$ chmod +x ./turtle.sh.

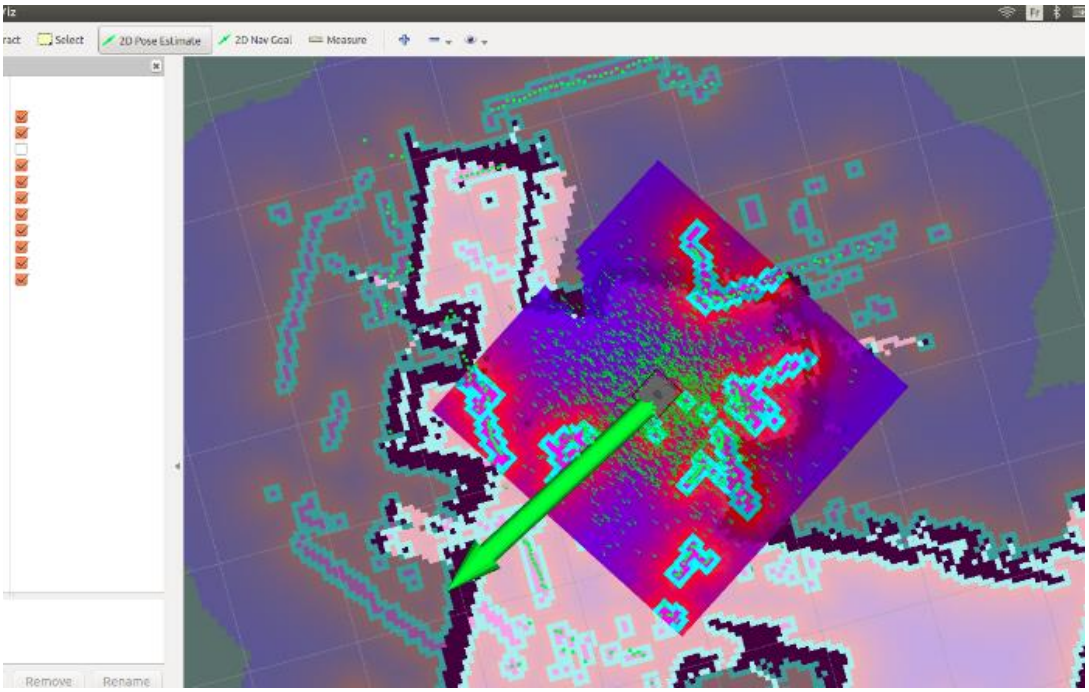
Maintenant que vous avez votre carte d'enregistrée, nous allons utiliser le nœud de navigation du robot pour enregistrer les porsitions que l'on veut sur la carte ainsi que l'origine du robot.

Comme pour le SLAM, il faut lancer roscore, lancer turtlebot3_robot.launch en ssh sur la Pi, puis dans un autre terminal utilisez :

$ roslaunch turtlebot3_navigation turtlebot3_navigation.launch
map_file:=$HOME/map.yaml

où $HOME représente le chemin qui mène au dossier où se trouve la carte que vous avez enregistrée et « map » le nom de cette carte.

Commencez par initialiser la position du robot. Pour se faire, indiquez au robot où il se trouve etd ans quel sens en cliquant sur 2D Pose Estimate et cliquer sur la carte (relâcher lorsque l'orientation du robot est la même que celle de la flèche) :



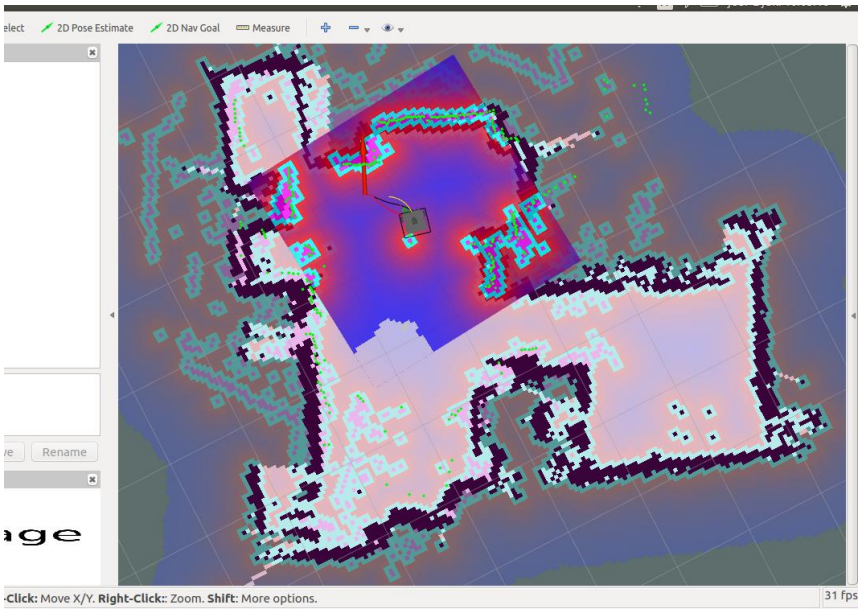
Ensuite, vous aller pouvoir décider des positions que le robot doit enregistrer.

Dans un autre terminal entrez la commande suivante :

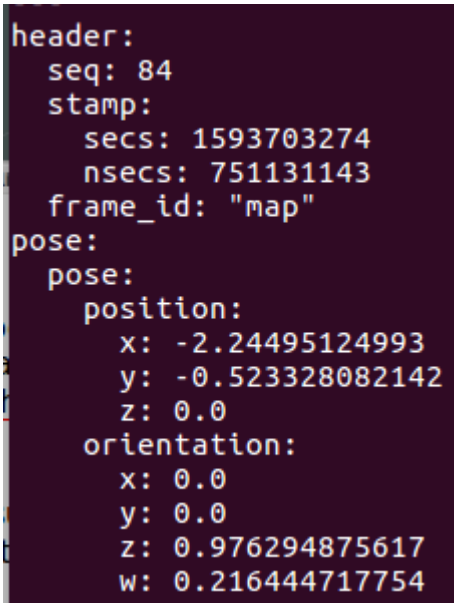
$ rostopic echo /amcl_pose

Envoyez le robot aux positions que vous voulez à l'aide du bouton 2D Nav Goal.

Commencez par son dock, envoyez une position au robot pour que celui-ci se déplace à son dock, c'est-à-dire en face de l'AR marker que vous avez imprimé précédemment (à 1 dizaine de cm). Exemple :



Ouvrez un document de traitement de texte et tapez « dock : » et copiez-en dessous la partie pose (position + orientation) que vous obtenez dans le « rostopic echo /amcl_pose » :



Procédez ensuite de la même manière pour les lieux que vous voulez enregistrer et que le robot pourra alors rejoindre par la suite.

Pensez à les nommer par des noms explicite type « chambre », « salon », etc.

Il faut aussi effectuer une sauvegarde d'un point qui se situe à 1 mètre du dock (peu importe l'orientation), pour que le robot y aille lorsqu'il est déchargé.

2. Configurations des points de la carte

Position initiale :

Pour le point de départ du robot, c'est-à-dire celui où le robot est à son dock (face au marqueur à une dizaine de cm), il faut effectuer un calcul.

Les informations d'orientation du robot correspondent à :

$$w = cos\left(\frac{alpha}{2}\right) et\ z = sin\left(\frac{alpha}{2}\right)$$

Où alpha est l'angle entre z et l'orientation du robot en radians.

Pour obtenir alpha on peut donc calculer :

$$alpha = 2cos^{-1}(w) = 2sin^{-1}(z)$$

Sur l'exemple précèdent, on a donc alpha = 2*arccos(0,2164) = 2,7 rad.

Ensuite il faut aller modifier la position initiale du robot, entrez la commande :

$ cd ~/catkin_ws/src/turtlebot3/turtlebot3_navigation/launch

$ nano amcl.launch

```
1 <launch>
2   <!-- Arguments -->
3   <arg name="scan_topic"     default="scan"/>
4   <arg name="initial_pose_x" default="-0.902469525337"/>
5   <arg name="initial_pose_y" default="-0.568361103535"/>
6   <arg name="initial_pose_a" default="-0.4"/>
7
8   <!-- AMCL -->
9   <node pkg="amcl" type="amcl" name="amcl">
```

Comme sur la capture ci-dessus, changez les valeurs misent à 0.0 par défaut à l'aide du fichier texte que vous avez écrit plus tôt, avec :

- initial_pose_x est la valeur de x dans position,
- initial_pose_y est la valeur de y dans position,
- initial_pose_a est la valeur de alpha que vous venez de calculer.

Positions dans la carte :

Maintenant, vous aller modifier les positions que vous voulez que le robot puisse rejoindre.

Dans le fichier battery_monitoring.cpp (trouvable dans le répertoire : ~/catkin_ws/src/battery_monitoring/src/), modifier les lignes 85 à 91 avec les valeurs que vous avez enregistrées précédemment pour une position à 1 mètre du dock.

```
// Si la batterie est inférieure à 5% (= 11.071 V) :
if (result < 5){
        // On empeche la personne d'envoyer de nouveaux objectifs au robot :
        system("rosnode kill /simple_navigation_goals");

        // Position proche du dock
        goal.target_pose.pose.position.x = -1.64440190792;
        goal.target_pose.pose.position.y = -0.205766946077;
        goal.target_pose.pose.position.z = 0;
        goal.target_pose.pose.orientation.x = 0;
        goal.target_pose.pose.orientation.y = 0;
        goal.target_pose.pose.orientation.z = -0.560109994579;
        goal.target_pose.pose.orientation.w = 0.828418248213;
```

Ainsi modifiez la position en x et y ainsi que l'orientation en z et w.

Dans le fichier dans le fichier simple_navigation_goals.cpp (trouvable dans le répertoire ~/catkin_ws/src/simple_navigation_goals/src/), modifier les lignes 117 à 123, 148 à 154 et 174 à 181.

Pour le dernier emplacement que je viens de citer, vous pouvez recopier ce que vous venez de rentrer dans battery_monitoring.cpp.

Pour les autres, vous mettrez les positions que vous avez enregistré, (sachant que pour moi elles sont nommées salon et cuisine).

Lorsque tout est terminé, effectuez la commande :

$ cd ~/catkin_ws && catkin_make

3. Configuration de la page Web

La dernière étape de nos configurations : la page Web.

Rendez-vous dans le dossier Web_App téléchargé sur le GitHub, puis ouvrez le fichier webui.js.

Dans ce fichier modifiez la ligne 10, changez l'adresse IP actuelle par celle de votre PC.

Si vous voulez changer le nom et la quantité de boutons, vous pouvez vous inspirer de ceux déjà existants et les modifier :

- pour la création des boutons, référez-vous lignes 48 à 54 dans Web_App/index.html ;
- pour les fonctions derrière ces boutons (avec la correspondance de l'entier pour chaque bouton), référez-vous lignes 58 à 76 dans Web_App/webui.js ;
- pour les modifications associées aux choix de l'utilisateur, veuillez mettre une concordance entre les valeurs de my_message.data dans Web_App/webui.js avec la valeur de id prise dans les cases du switch dans le fichier simple_navigation_goals.cpp.

Vous pouvez maintenant utiliser pleinement les fonctionnalités.

IV. Mise en marche

Dans un premier temps il faut allumer le robot, lui laisser le temps de démarrer (environ 30s), puis il faut exécuter le bash turtle.sh (commande : $ ./turtle.sh) et attendre que tout soit lancé (environ 50s). Ensuite si vous voulez vous connecter à l'interface Web depuis le même PC il faut rechercher « http://127.0.0.1:5500/index.html » dans un navigateur Web (testé avec Chrome et Firefox). Si vous voulez vous connecter avec un autre appareil comme par exemple un smartphone, il faut que ce dernier soit connecté au même réseau Wi-Fi que votre PC (et robot) et rechercher dans un navigateur Web :
« http://*adresse_IP_du_PC*:5500/index.html ».
Par exemple : http://192.168.1.33:5500/index.html ».

V. Annexe