Documentation TurtleBot

3iL ingénieurs

L'Institut d'ingénierie informatique de Limoges

DocumentationTurtleBot.docx

Table des matières

Documentation TurtleBot :

Inventaire Turtlebot

Lien vers le exel qui fait l'inventaire des Turtlebot
Inventaire_Turtlebot3 (2).xlsx

Sur PC UBUNTU :

- Installer Ubuntu
- Installer ROS
- Modifier le bashrc pour mettre le bon robot en master :

```
$ nano .bashrc
```

A la fin du fichier, modifier l'adresse IP du master par celle du robot auquel vous souhaitez vous connecter :

```
export ROS_MASTER_URI=http://(Ip turtlebot ):11311
export ROS_HOSTNAME=(Ip de la machine)
```

Attention : PAS METTRE LES () autour des IP

- Création du .sh :

On voudrait pouvoir avoir un exécutable qui lorsqu'il est cliqué permette de rentrer l'adresse ip du robot voulu puis d'appareillé le pc avec ce dernier. Ainsi que la visualisation des résultats du lidar grâce à RViz

On crée un fichier connect_and_launch.sh :

```
nano connect_and_launch.sh
```

On demande l'adresse IP de l'utilisateur :

```
read -p "Veuillez entrer l'adresse IP du TurtleBot : " TURTLEBOT_IP
```

On exporte la variable :

```
export TURTLEBOT_IP=$TURTLEBOT_IP
```

Puis on source les fichiers setup.bash de ROS et du TurtleBot :

```
source /opt/ros/noetic/setup.bash
source ~/catkin_ws/devel/setup.bash
```

On se connecte au TurtleBot :

```
export ROS_MASTER_URI=http://$TURTLEBOT_IP:11311
```

```
export ROS_HOSTNAME=(Ip du pc)
export TURTLEBOT3_MODEL=burger
```

On lance rviz :

```
roslaunch turtlebot3_gazebo turtlebot3_gazebo_rviz.launch &
```

On lance les commandes de déplacements :

```
roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

On enregistre :

```
Ctrl + x
```

Puis, on rend le script bash exécutable :

```
$ chmod +x connect_and_launch.sh
```

Pour réaliser la partie turtlebot, branchez votre robot à un écran et connectez-vous avec :

Identifiant : ubuntu

Mot de passe : turtlebot

Sur TURTLEBOT (version Burger) :

On retire le mot de passe et on autologin le robot afin qu'il puisse s'initialiser sans interface, directement lors de son allumage.

- Retirer le password :

```
$ sudo passwd –d ubuntu
```

- Autologin de l'identifiant :

```
$ sudo mkdir -p /etc/systemd/system/getty@tty1.service.d/
$ sudo nano /etc/systemd/system/getty@tty1.service.d/override.conf
```

Dans le fichier (vide), on place :

Attention : Sensible à la casse (copier pareil)

```
[Service]
```

```
ExecStart=
ExecStart=-/sbin/agetty --autologin ubuntu --noclear %I $TERM
```

Attention : espace entre (agetty et --autologin, ubuntu et --noclear, %I et $TERM)

On enregistre le fichier :

```
$ sudo systemctl daemon-reload
$ sudo systemctl restart getty@tty1
```

Puis si le redémarrage ne s'effectue pas. On peut relancer avec :

```
$ sudo reboot
```

- Mise en place de tous les packages dans le roslaunch (Roscore, Lidar, Omométrie, Rosmaster) :

On ouvre le fichier .profile avec :

```
$ nano ~/.profile
```

On ajoute le .launch

```
$ roslaunch turtlebot3_bringup turtlebot3_robot.launch
```

- Ajout d'un son lors du lancement :

Ensuite, créer un nouveau fichier "play_ros_sound" avec :

```
$ nano ~/play_ros_sound.sh
```

Ce fichier est donc vide, copier ces lignes dedans.

Attention copier exactement pareil, très sensible à la case

```
until rostopic list; do sleep 1; done
rostopic pub /sound turtlebot3_msgs/Sound « value: 1 »  -1
```

ATTENTION : bien mettre un espace entre : et 1 dans le <value: 1>

Puis, on enregistre le fichier :

Ctrl + x

On rend le script exécutable en début de programme

```
$ sudo chmod +x ~/play_ros_sound.sh
```

On ouvre le fichier .profile avec :

```
$ nano ~/.profile
```

On ajoute à la fin du fichier, **AVANT** le roslaunch qui a été mis en place précédemment, le fichier de lancement du son (Mise en place de tout les packages dans le roslaunch) :

```
~/play_ros_sound.sh &
```

Ainsi que la commande pour mettre le clavier en AZERTY :

```
sudo loadkeys fr
```

On enregistre le fichier :

```
Ctrl +x
```

– Mise en place du robot en tant que Master :

On va dans le bashrc :

```
$ nano .bashrc
```

On ajoute :

```
export TURTLEBOT3_MODEL=burger
export ROS_MASTER_URI=http://(Ip turtlebot) :11311
export ROS_HOSTNAME=(Ip de la machine)
```

- Caméra Raspberry :

On met à jour et on ajoute les packages nécessaire :

```
$ sudo apt upgrade
$ sudo apt install libraspberrypi-bin
$ sudo reboot
```

Accès au paramètre :

```
$ raspistill
```

Pour prendre une photo :

```
$ raspistill -o (nom_photo).jpeg
```

- ROS raspicam_node

git clone https://github.com/UbiquityRobotics/raspicam_node.git

yaml https://raw.githubusercontent.com/UbiquityRobotics/rosdep/master/raspberry-pi.yaml

Le noeud se lance automatiquement sur le robot et pour avoir le rendu sur l'ordinateur ouvrir un terminal et lancer rqt_image_view dans un terminal de commande

- Ajout de la LED:

Voir dans https://github.com/rapidoclash/Turtlebot/Led

Déploiement Robot

Pour éviter de refaire toutes les manipulations précédentes, il est possible de charger une copie d'un robot fonctionnel à partir d'un fichier.

Étapes à Suivre :

1. **Préparation de la Carte SD :**
- Retirez la carte SD d'un robot.
- Connectez-la à un ordinateur.
2. **Copie d'un Robot "Parfait" vers un PC :**
- Utilisez la commande suivante pour copier le disque du robot (ici nommé `sdb`) vers un fichier (`turtlebot3`) :

```
sudo dd if=/dev/sdb of=/home/ubuntu/Document/turtlebot3 bs=10M
```

ou telecharger robot_parfait sur https://github.com/rapidoclash/Turtlebot/

- Note : Cette opération prend environ 45 minutes.

3. **Copie d'un fichier "Parfait" vers un Robot :**
- Utilisez la commande suivante pour copier votre fichier de sauvegarde vers le disque du robot :

```
sudo dd if=/home/ubuntu/Document/turtlebot3 of=/dev/sdb bs=10M
```

- Note : Cette opération prend environ 45 minutes.
- Remettez ensuite la micro SD dans le robot.

4. **Configuration de l'Adresse IP :**

- Assurez-vous de configurer la bonne adresse IP pour le robot (notée sur le robot).(voir partie Configuration Réseau)
- Pour cela, suivez les instructions de configuration réseau.

5. **Modification du Fichier `.bashrc` :**

- Dans le répertoire utilisateur (`/home/ubuntu`), ouvrez le fichier `.bashrc` avec l'éditeur de texte `nano` :

```
nano ~/.bashrc
```

- Allez à la fin du fichier et modifiez les lignes suivantes en fonction de l'adresse IP du robot :

```
export ROS_MASTER_URI=http://(Ip turtlebot):11311
export ROS_HOSTNAME=(Ip turtlebot)
```

- Sauvegardez et fermez le fichier.

6. **Recharger les Modifications :**

- Exécutez la commande suivante pour appliquer les modifications :

```
source ~/.bashrc
```

Configuration Réseau:

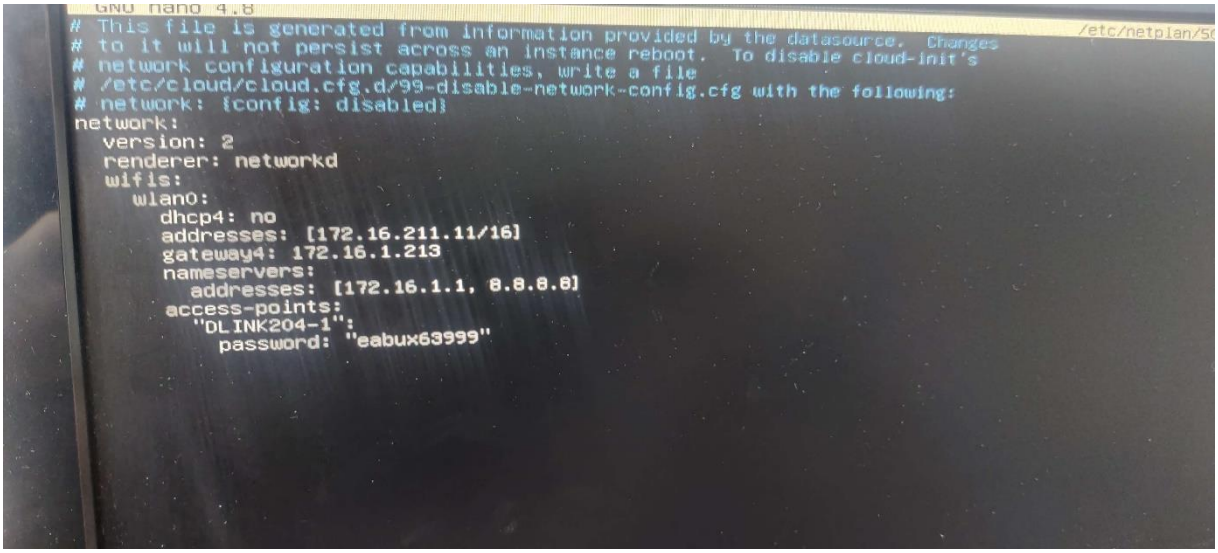
```
sudo nano /etc/netplan/50-cloud-init.yaml
```

ou

```
sudo nano /etc/netplan/50-network-manager-all.yaml
```

remplacer sur la ligne « addresses : » par [ip_du_robot/16]

Sauvegarder le fichier et le fermer




```
sudo netplan apply
```

Individualisation des robots :

Après plusieurs idées, nous avons choisi l'utilisation de pastilles de couleurs afin de les différencier. Cette solution est simple et peu couteuse.

Composition du Turtlebot3 :

Le TurtleBot3 est composé de :

- Châssis

- Roues

- Moteurs

Électronique :

- Carte Raspberry Pi

- ARM Cortex-M7

Alimentation :

- Batterie Lipo

- Chargeur de Batterie

Capteurs :

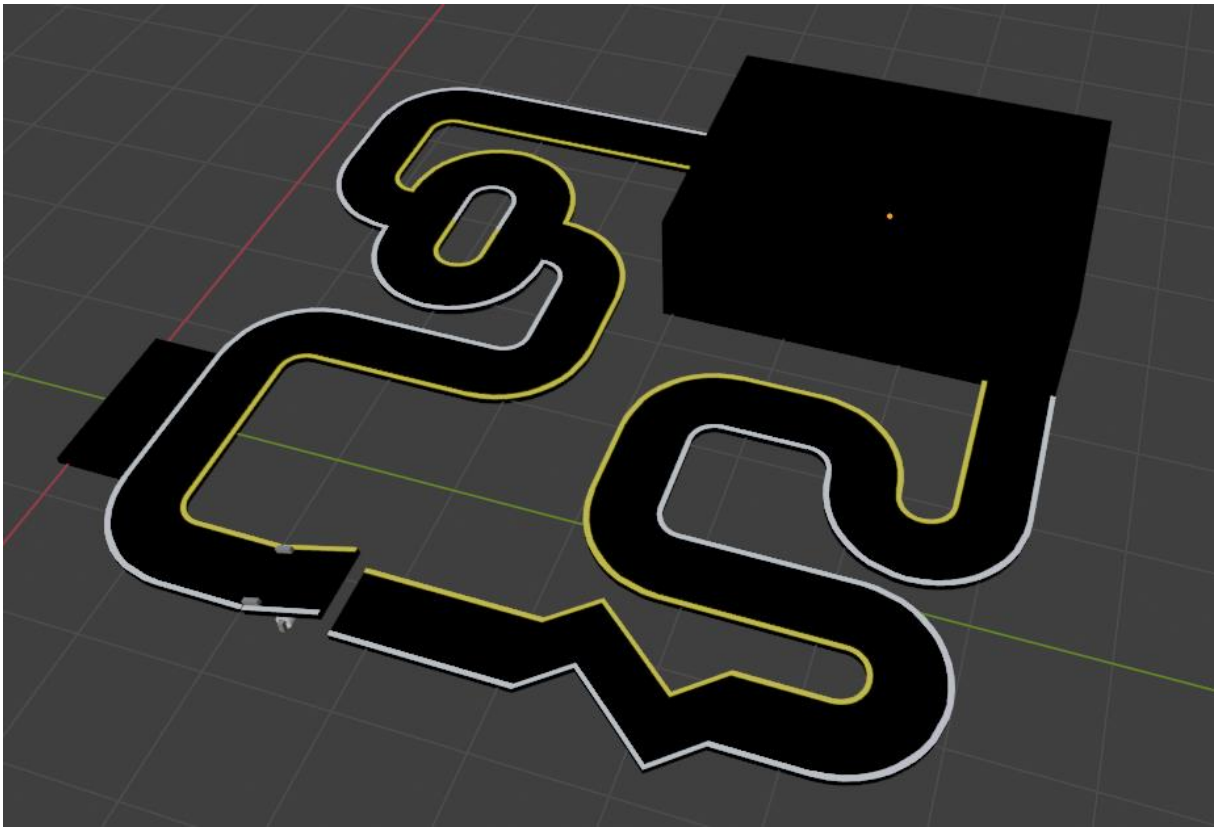
- LIDAR : Utilisé pour la détection d'obstacles et la cartographie.
- Caméra

Logiciel :

- ROS (Robot Operating System)

Circuit Turtlebot :

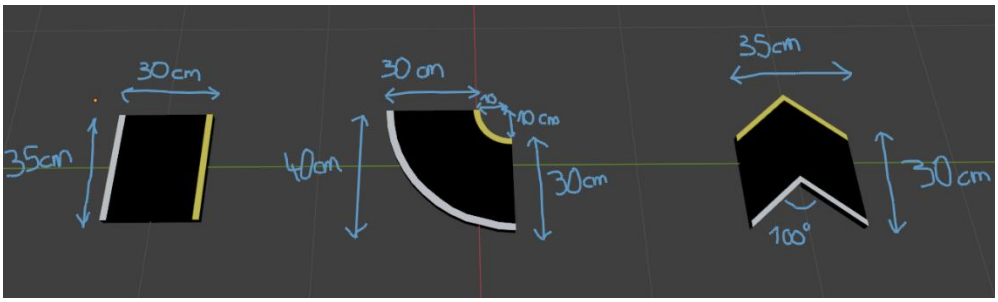
Modélisation 3D du circuit avec Blender

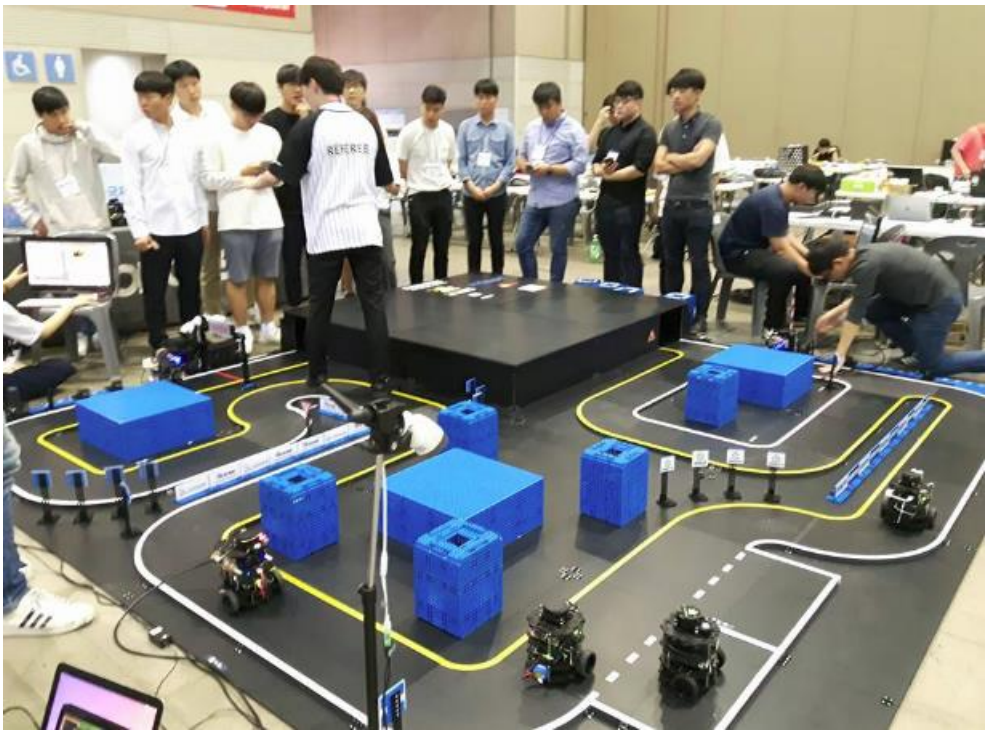


Besoin pour la conception du circuit :

- 25 rectangles : 35x30 cm

- 17 courbes : 40x40 cm
- 2 pièces en zigzag : 35x30 cm

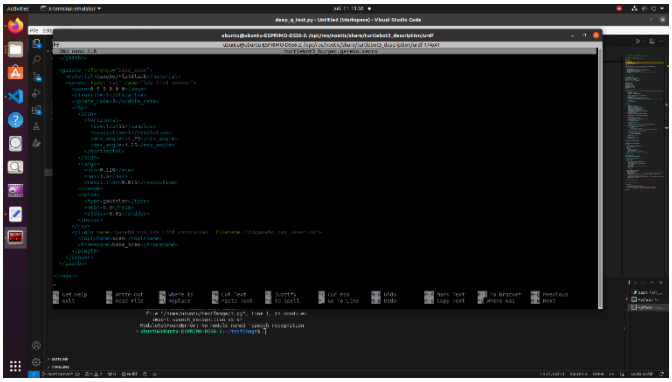




Machine learning :

Stage 0 et 1 opérationnel
https://github.com/rapidoclash/Turtlebot/tree/main/turtlebot3_machine_learning

update de la nape laser



Nous avons reduit le nombre d'entrer lidar pris en compte pour le modifier il faut aller dans le fichier

Pour cette partie le turtlebot possède 4 environnements d'apprentissage nommé « stage »

Pour lancer ces stages, on lance un terminal :

```
cd catkin_ws
source devel/setup.bash
roslaunch turtlebot3_gazebo turtlebot3_stage_(numéro_du_stage).launch
```

Puis dans un nouveau terminal :

```
cd catkin_ws
source devel/setup.bash
roslaunch turtlebot3_dqn turtlebot3_dqn_stage_(numéro_du_stage).launch
```

Il y a une erreur sur la librairie de Keras

Pour régler ça. Ouvrez dans turtlebot3_dqn/nodes/

Le fichier python du stage voulu et modifier la fonction TRAINMODEL :

La méthode predict a une très grosse fuite de mémoire il vaut mieux utiliser la fonction équivalente predict_on_batch

nous avons eu des problèmes lors de la deuxième partie du lancement du stage 2 et 4.

Pour le stage 2 :

Vérifiez bien que keras et tensorflow sont installés :

```
pip show keras tensorflow
```

Puis vérifiez dans vos fichiers :

```
catkin_ws/src/turtlebot3_machine_learning/turtlebot3_dqn/nodes/turtlebot3_dqn_stage_2
```

Ouvrez le et vérifiez ligne 33 que le .core n'est pas ajouté

Normalement vous devez avoir :

from keras.layers import Dense, Dropout, Activation

Enfin vérifiez le state_size ligne 151 qui doit être mis à 364 :

```
state_size = 364
```

Pour le stage 4 :

Ouvrez vos fichiers :

catkin_ws/src/turtlebot3_machine_learning/turtlebot3_dqn/nodes/turtlebot3_dqn_stage_4

Ouvrez le et ajoutez après la ligne 47 :

```
self.action_size = actions_size
```