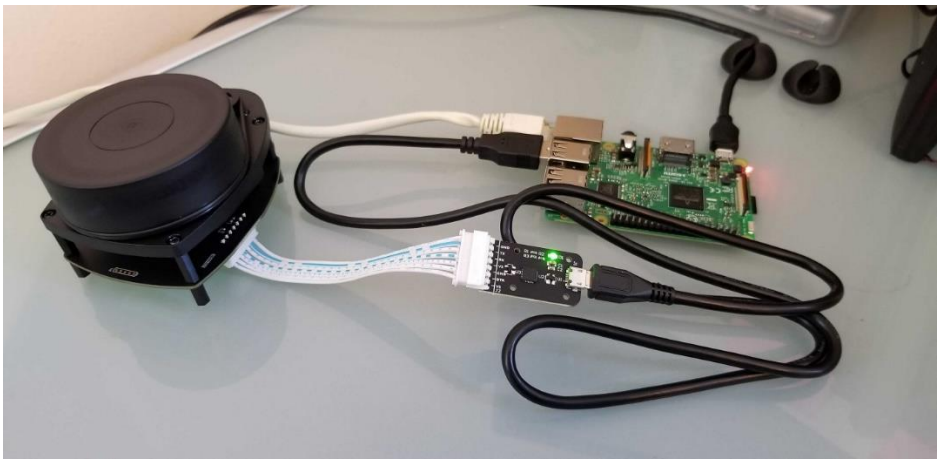
Eurobot : Documentation

Acquisition des données du lidar sur Raspberry avec ros :

Le lidar RPlidar est un lidar bon marché qui est assez facile à prendre en main avec ros.



Matériel requis :

- Un lidar RPlidar.
- Un câble micro USB.
- Une Raspberry.

Installation :

Il suffit de brancher le lidar à la Raspberry à l'aide d'un câble micro USB et d'entrer les commandes suivantes pour l'installer si ce n'est déjà fait (Attention, il faut s'assurer avant de brancher ce câble qu'aucun autre périphérique ne soit déjà branché afin que le lidar se trouve sur le port USB0 car le code appelé par roslaunch utilise par défaut ce dernier) :

```
cd ~
mkdir -p rplidar_ws/src
sudo git clone https://github.com/Slamtec/rplidar_ros.git
cd rplidar_ros
sudo mv * ~/rplidar_ws/src
cd ~/rplidar_ws/
catkin_make
echo "source ~/rplidar_ws/devel/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

Première utilisation :

Une fois l'installation terminée, il faudra vérifier que le lidar se trouve bien sur le port ttyUSB0 et rajouter les droits nécessaires comme ceci (Attention, suivant le câble micro USB utilisé, il est possible que le lidar ne fonctionne pas) :

```
sudo chmod 666 /dev/ttyUSB0
```

Une fois cela fait, il suffit d'entrer la commande suivante pour publier sur le topic /scan les données récoltées par le lidar en temps réel :

```
roslaunch rplidar_ros rplidar.launch
```

Une fois cette commande entrée, le lidar commence à émettre et il est possible d'accéder à ces données transmises sur le topic /scan soit via un scripts python par exemple, soit directement en utilisant les commandes basiques de ros. Dans un premier temps, nous allons utiliser ces

dernières. Pour ce faire, il suffit d'ouvrir une nouvelle ligne de commande et écouter sur le topic /scan comme ci-dessous.

```
rostopic echo /scan
```

Utilisation avec python :

Pour accéder aux données depuis python, il suffit d'utiliser le script subscribeLaserScan.py :

```
import rospy
from sensor_msgs.msg import LaserScan


def callback(data):
    angle_increment = (data.angle_max-data.angle_min)/(len(data.ranges)-1)

    # Angle without inf
    angle = [data.angle_min + angle_increment*i for i in
range(len(data.ranges)) if data.ranges[i] != float("Inf")]
    # Range without inf
    ranges = [data.ranges[i]*1000 for i in range(len(data.ranges)) if
data.ranges[i] != float("Inf")]

    print("theta = {}".format(angle))
    print("r = {}".format(ranges))


def listener():

    # In ROS, nodes are uniquely named. If two nodes with the same
    # name are launched, the previous one is kicked off. The
    # anonymous=True flag means that rospy will choose a unique
    # name for our 'listener' node so that multiple listeners can
    # run simultaneously.
    rospy.init_node('rplidarNode', anonymous=True)

    rospy.Subscriber('/scan', LaserScan, callback)

    # spin() simply keeps python from exiting until this node is stopped
    rospy.spin()


if __name__ == '__main__':
    listener()
```

Reconnaissance de couleur :

Dans la plupart des concours Eurobot, il est nécessaire de pouvoir faire de la reconnaissance de couleur. C'est pourquoi nous avons créé un algorithme qui permet de faire ce traitement à partir d'une caméra.

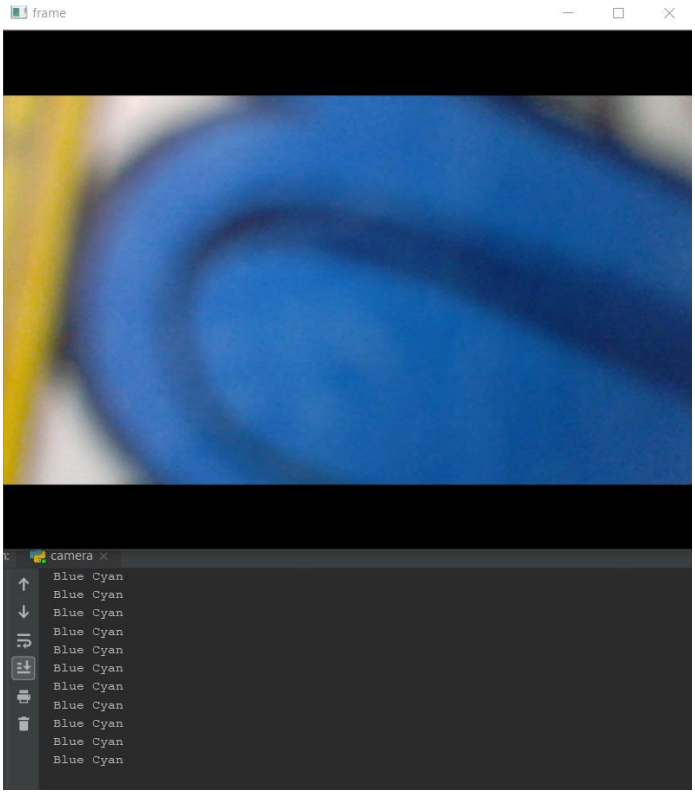
Matériel requis :

- Une Raspberry/un pc.
- Une caméra compatible Raspberry/une webcam.

Installation :

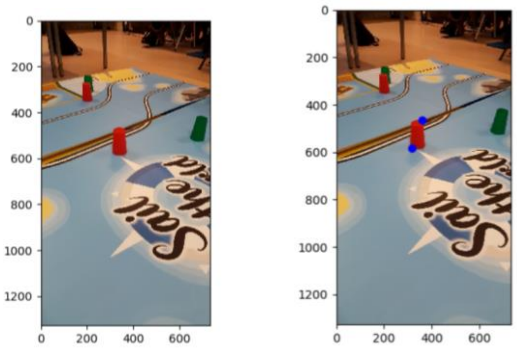
- Installer PIL
- Installer sklearn
- Installer colorsys

Reconnaissance de couleur en live :



Pour faire de la détection de couleurs en live, il suffit de lancer le script camera.py qui va analyser les images capturées depuis la caméra afin d'en déterminer la couleur moyenne en temps réel.

Reconnaissance de couleur d'une partie d'image :

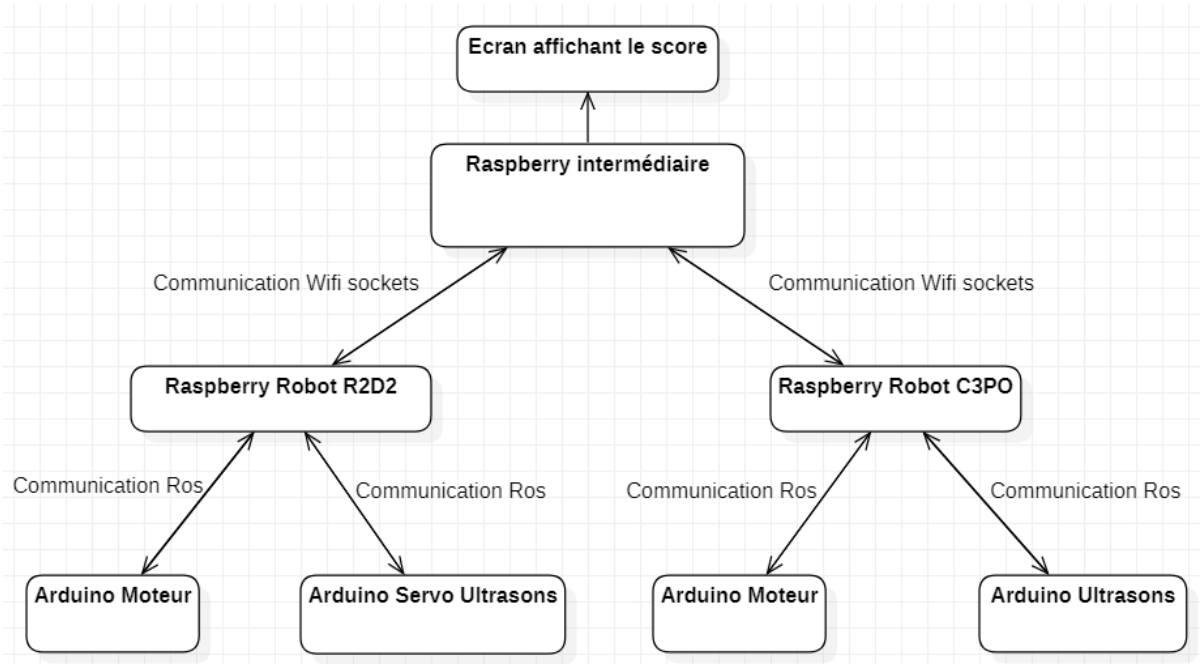


```
<PIL.Image.Image image mode=RGB size=43x119 at 0x15980299
Mid Red
```

Pour détecter la couleur d'un morceau d'image, il suffit de mettre dans le dossier courant une image nommé test.jpg et de lancer le script DetectColorFromCoordinates.py qui, une fois que l'on aura cliqué 2 fois sur l'image affichée, renverra la couleur du morceau d'image sélectionné.

Protocole général mis en place :

Le protocole que nous avons mis en place est illustré ci-dessous.



Communication Wifi et affichage du score :

Matériel requis :

- 3 Raspberry.
- Un écran.

Installation :

Il faut installer Kivy qui est un Framework open source multiplateforme qui permet de créer des interfaces graphiques facilement en python sur la Raspberry intermédiaire. Pour ce faire, il faut installer python 3.6 sur cette Raspberry puis entrer les commandes ci-dessous.

```
pip3.6 install docutils pygments pypiwin32 kivy.deps.sdl2 kivy.deps.glew
pip3.6 install kivy.deps.gstreamer
pip3.6 install kivy.deps.angle
pip3.6 install kivy
```

Utilisation :

Chaque robot possède sa propre Raspberry qui lui permet de communiquer avec la Raspberry intermédiaire via Wifi en utilisant des sockets pour faire soit incrémenter positivement ou négativement le score affiché par la Raspberry intermédiaire, soit pour recevoir ou envoyer un message à la Raspberry de l'autre robot.

Pour afficher le score initial et initialiser la communication Wifi, il suffit de lancer le script server.py sur la Raspberry intermédiaire et pour chaque Raspberry de chaque robot lancer le script client.py en oubliant pas d'utiliser l'adresse IP de la Raspberry intermédiaire qui joue le rôle de serveur dans ce dernier.

Une fois cela fait, il suffit d'entrer l'une des commandes ci-dessous sur une des Raspberry client pour incrémenter positivement ou négativement le score.

```
score +8
score -5
```

Pour envoyer un message à la Raspberry de l'autre robot, il suffit d'utiliser la commande « msg » comme ci-dessous.

```
msg hello other robot
msg end of the tuto
```

Algorithme d'évitement :

Introduction :

L'algorithme d'évitement a pour but de permettre au robot R2D2 de se déplacer d'un point A vers un point B en contournant les potentiels obstacles qu'il pourrait rencontrer au travers de son chemin. Pour ce faire, à chaque instant, on connait les informations suivantes :

- La position absolue du robot sur la table.
- La position absolue de la destination.
- La position absolue des bords de table.
- La position absolue de chaque obstacle autour du robot détecté par le lidar et potentiellement aussi par des ultrasons en temps réel.

Durant la conception de cet algorithme, il a été nécessaire de créer un simulateur pour faciliter le développement. Pour ce faire, nous avons utilisé Kivy qui est un Framework open source multiplateforme qui permet de créer des interfaces graphiques facilement en python.

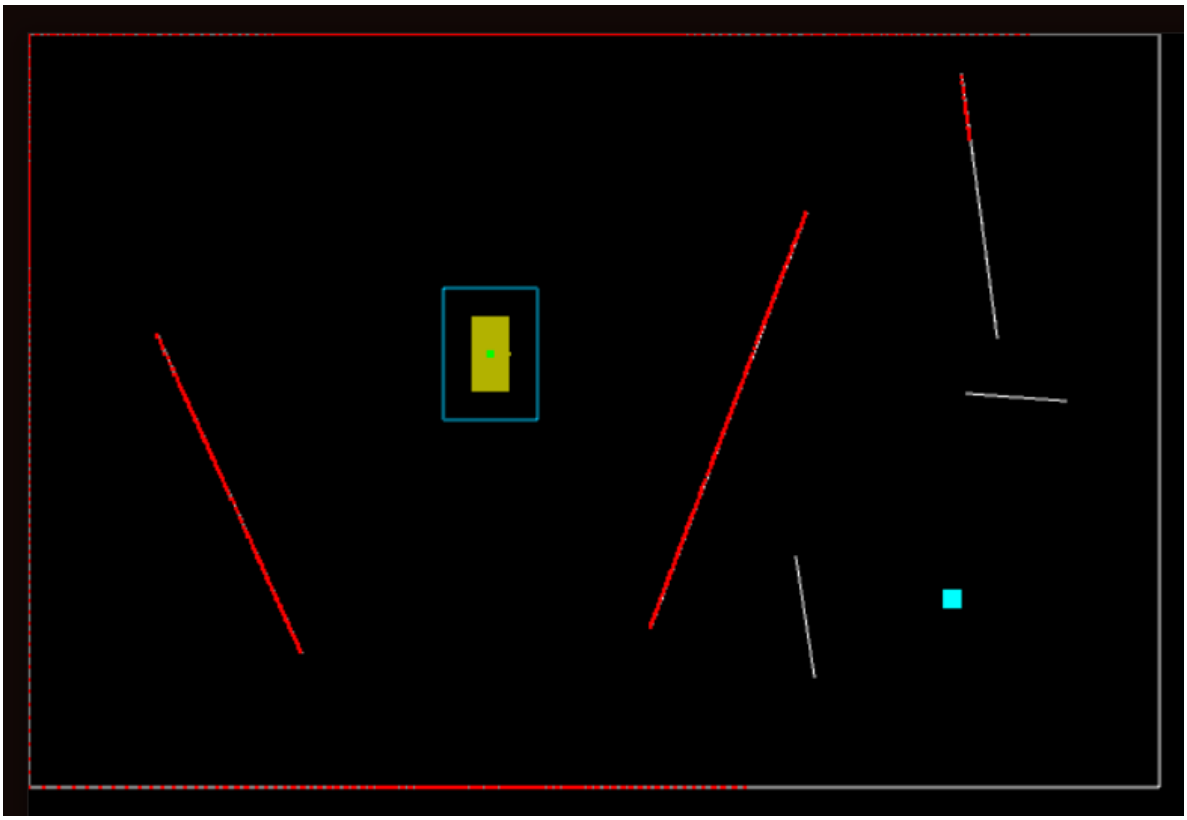
Installation :

Pour installer Kivy, il suffit d'entrer les commandes suivantes :
(Attention, une version de python plus grande que python3.6 pourrait potentiellement causer des soucis)

```
pip3.6 install docutils pygments pypiwin32 kivy.deps.sdl2 kivy.deps.glew
pip3.6 install kivy.deps.gstreamer
pip3.6 install kivy.deps.angle
pip3.6 install kivy
```

Utilisation :

Une fois Kivy installé et le programme démarré, il y a plusieurs éléments que nous pouvons distinguer dans l'interface :



- En jaune, on peut voir le robot physique avec son centre en vert.
- En bleu foncé autour du robot, on peut voir sa zone de sécurité, zone dans laquelle aucun obstacle ne pourra pénétrer.
- En bleu clair, on peut voir la destination du robot.
- En gris, on peut voir les différents obstacles qui se trouvent sur le terrain.
- En rouge, on voit ce que le lidar voit et donc ce que le robot voit en temps réel (avec un tampon sur les 20 derniers scans)

Il est possible, avant de faire une simulation, d'ajouter des droites qui représenterons des obstacles. Pour ce faire, il suffit de faire un clic gauche sur le 1er point par lequel doit passer cette droite en question et un deuxième clic gauche sur le 2ème point par lequel celle-ci doit passer.

Il est aussi possible de modifier la destination en faisant un clic droit sur la destination désirée.

Une fois le terrain complètement configuré, il suffit d'appuyer sur enter 1 fois pour lancer la simulation. (Il sera possible d'ajouter des droites et de modifier la destination une fois la simulation terminée c'est-à-dire une fois que le robot sera arrivé à sa destination)

Principe de fonctionnement général :

https://www.youtube.com/watch?v=BbewRdceUCM