29/09/2022

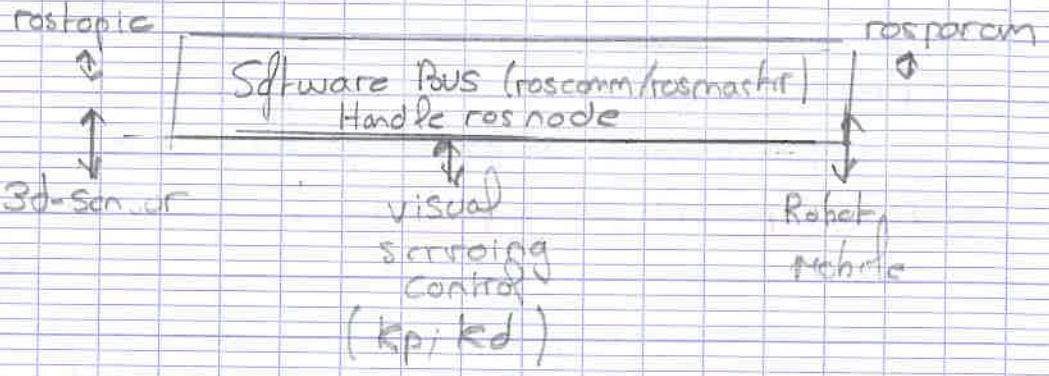
TP 2 : Robotique Mobile et Navigation

tlasguigne@laas.fr

PC 3-14-102

Introduction :

ROS = middleware (bus interlogiciel)
↳ portabilité du code (paramétrable)



↳ collaboration possible => standard
↳ communication entre les PC embarqués sur le robot
↳ publication/souscription des messages
↳ enregistrer rejouer les messages

⚠ ROS 2 n'assure pas le temps réel contrairement à ROS 2

ROS supporté par Ubuntu
↳ Melodic Morenia supporté par Ubuntu 2018

Astuce : terminator
ctrl+maj+O pour séparer horizontalement
ctrl+maj+E pour séparer verticalement
ctrl+D pour supprimer un terminal

source /opt/ros/melodic/setup.bash est à ajouter au fichier .bashrc (récupéré par le bash à chaque ouverture)

Initialiser un workspace de compilation CATKIN :

```
mkdir -p /DGC-catkin-ws/src
cd /DGC-catkin-ws/src
catkin_init_workspace dans le src
```

Créer une source dans les package

```
source $Home/catkin_ws/devel/setup.bash
```

Variables d'environement avec par env | grep ROS (enchaîner les commandes en cascade sur la sortie)

ROS_MASTER_URI = http://localhost:11311

⚠ pour travailler en local

Le master_URI est sous le format hostname : port_number

hostname : ordinateur où roscore est lancé

port_number : port où roscore attend les connexion

Création de paquet :

Un paquet contient un fichier package.xml au format catkin

Un seul paquet par répertoire ! et un CMakeLists.txt

catkin_create_pkg beginner_tutorials std_msgs rospy roscpp (depends 2 / depends 2)

crée un package beginner_tutorials et fait des dépendances avec std_msgs, rospy et roscpp

Dépendance : ce de quoi le package pour fonctionner par ordre d'importance de gauche à droite depends 1, depend 2, etc...

rospack depends beginner_tutorials affiche les dépendances du package beginner_tutorials (possible après le build des packages)

Compilation des paquets : catkin_make a faire dans la racine du workspace. Cela créer un répertoire build qui stocke les objets compilés intermédiaire et un répertoire devel qui stocke les exécutables et les librairies compilées

catkin_make install crée un répertoire install contenant des fichiers de setup et fichiers lib et sdare (comme dans le répertoire /usr du système)

work-space-folder
- src
  - CMakeLists.txt
  - Package-1
    - CMakeLists.txt
    - package.xml
    - include (contient les librairies)
    - src (dépend de l'implémentation C++ ou Python)

Définitions: Graphe d'application avec ROS

Node: processus qui utilise ROS pour communiquer avec d'autre noeud

Message: Type de données ROS utilisés pour souscrire ou publié sur un topic

Topics: Les nodes peuvent publier des messages sur un topic aussi bien que souscrire à un topic pour recevoir des messages

Master: Nom du service ROS (ie. aide les noeuds à se trouver mutuellement).

Parametres: Information très peu dynamique qui doivent être partagés dans l'application

rosout: Equivalent de stdout/stderr en C

roscore: Master + rosout + parameter server

Nod: fichier exécutable dans un paquet ROS
- ↳ donne un processus avec un nom
- ↳ utilise une librairie client pour communiquer avec d'autre noeuds (rospy pour Python ou roscpp pour C++)
- ↳ peut publier ou souscrire à des topics

↳ peut fournir ou utiliser un service (ex basique : on fournit 2 entier et il retourne l'addition)

Communication ROS :

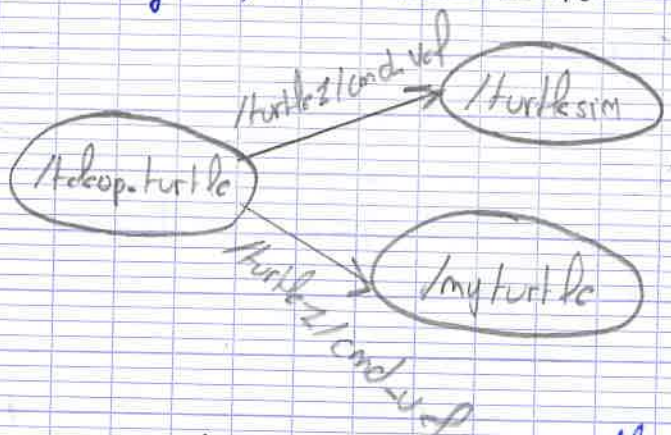
RosCore : annuaire pour localiser et interagir entre les nodes.
↳ on le lance toujours mais une fois.
↳ si roscore ne se lance pas
. vérifier la connection
. vérifier les droits en écriture du dossier .ros

roscore par lancer le roscore
rosnode list pour la liste des nodes actifs
rosnode info/ roscut pour obtenir des info sur un node
rosrun [package_name] [node_name] pour lancer un fichier exécutable / node d'un paquet
⚠ Passer à un autre terminal après avoir lancé le roscore
rosrun turtlesim turtlesim_node pour lancer turtle_sim (image de tortue)
si on lance sur le même terminal un même noeud alors ferme l'ancien et ouvre un nouveau
Solution
rosrun turtlesim turtle_sim_node __name := my_turtle
renommer le noeud

rosnode ping my_turtle pour vérifier s'il est actif

rosrun turtlesim teleop_key permet de contrôler la tortue au clavier
Si on avait lancer une autre tortue alors elle aurait suivi le même chemin car les nodes sont inscrits au même topic

Pour le vérifier, on trace rosrun rqt_graph rqt_graph



rosrun rqt_plot rqt_plot pour afficher l'évolution des topics

<u>rostopic</u> données publiées par les nœuds et auquel les nœuds souscrivent :

rostopic bw [nom_topic] bande passante
rostopic echo [nom_topic] affiche les messages à l'écran
rostopic hz [nom_topic] affiche la fréquence des messages
rostopic list affiche des information sur les topics actifs
rostopic pub [nom_topic] publie une donnée sur un topic
rotopic type [nom_topic] affiche le type d'un topic

<u>rqt :</u> interface d'affichage non 3D comprenant des pluggin.
Elle permet de construire une interface de contrôle incrémentale
rosrun rqt_console rqt_console

<u>roslaunch</u> : lit un fichier xml qui contient les paramètres pour lancer une application
=> on va créer un launch file notamment pour lancer plusieurs nodes en même temps dans un dossier launch au sein du package beginner_tutorial
touch turtlesim.launch

gedit turtlesim.launch

Le fichier contient

```
<?xml version="1.0"?>
<launch>
    <node name="turtle1" pkg="turtlesim" type="turtle
    sim_node" args="" />
</launch>
```

ros launch beginner_tutorials turtlesim.launch par lancer le fichier

Si plusieurs lignes dans le fichiers launch alors tous les noeuds sont lancés en même temps

⚠ Si un des nodes s'arrête alors il se peut qu'il kill les autres présents dans le fichier de launch.

rosbag : permet d'enregistrer et rejouer des messages

rosbag record -a enregitre tous les messages sous format .bag

ros bag record (-O subject) /turtle1/cmd_vel /turtle1/pose

(-O subject) : ajoute un préfixe au titre du fichier

rosbag play pour lire le fichier

rosservice : permet de lister et d'appeler les services d'un noeud

rosservice args affiche les arguments du service

rosservice call appelle le service avec les arguments (si tab)

rosservice find trouve les services par type de service

rosservice info affiche les info sur le service

rosservice list liste les service

rosparam : permet de gérer des données de configuration du fichier

rosparam set	fixe un paramètre
rosparam get	obtient un paramètre
rosparam load	charge un paramètre depuis un fichier
rosparam dump	mettre des paramètres dans un fichiers
rosparam delete	supprime un paramètre
rosparam list	liste le noms de paramètre

03/10/2022 La variable d'environement ROS_URI donne l'emplacement de l'annuaire objet dans lequel on indique au bus logiciel où se connecter.
ordinateur 112
On utilise un protocole http dans ROS 1 (non sécurisé)

N.B pas oublier de lancer le roscore dans un autre terminal (en pratique, il est lancé automatiquement par le robot)

Astuce: utiliser l'aide rostopic --help

rostopic echo /turtle1/pose affiche les message du topic

rostopic pub /turtle1/cmd_vel geometry
faire tab deux fois pour préremplir (remplir en unité SI)
ajouter -r 1 à la fin pour commander périodiquement
↑ ici toutes les seconde

Commande du robot:

1. Connexion en Wi-Fi au robot TIAGO_155 ou PAL_ROBOTICS
2. export ROS_MASTER_URI = http://10.68.01:11311
3. export ROS_IP = 10.68.132 ou dans tobin/ifconfig

rostopic list |grep cmd_vel affiche la liste des topic contenant cmd_vel

rostopic pub /mobile_base_controller/cmd_vel puis tab