Ros : turtle double avec étoile

- roscore
- rosrun turtlesim turtlesim_node
- rosrun turtlesim turtlesim_node __name:=mytur tle

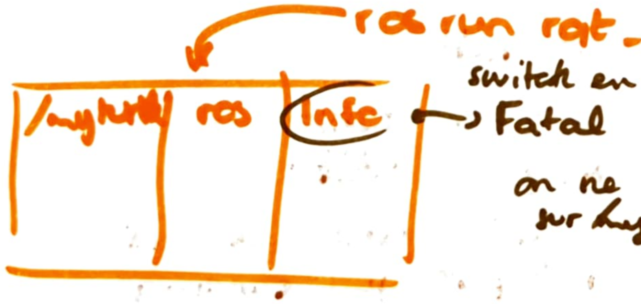
- rostopic pub /turtle1/cmd_vel geometry_msgs/Twist "linear :
  x : 1000
  y : 1000
  z : 0
  angular
  x : 0
  y : 0
  z : 1000" -r 4

linear	
x	1000
y	1000
z	~~1000~~ 0
angular	
x	0
y	0
z	1000

Pour modifier les info qu'a a sur un node

~~rqt~~ ~~ros rqt~~ rosrun rqt_console rqt_console

rosrun rqt_logger_level rqt_logger_level



on ne recevra que les message Fatal sur mytur tle

Se connecter au wifi

· lauch

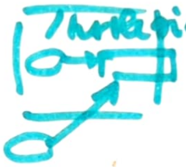
- creation d'un fichier.lauch dans ~~le~~ un répertoire lauch qu'on crée "mkdir lauch" dans le répertoire (workspace) upssitech_grd_ag_ws/src/begginer_tutorials/src/lauch)
- copier coller le xml
- lancer la cmd " $\underbrace{\text{roslauch}}_{\text{commande}}$ $\underbrace{\text{begginer\_tutorials}}_{\text{rep}}$ $\underbrace{\text{turtle mimic.lauch}}_{\text{fichier.lauch}}$"

<lauch>
<\lauch } balise lauch

<remap from="turtle1/cmd_vel" to="turtlesim1/turtle1/cmd_vel"/>

graph

rosrun rqt_graph rqt_graph



type

rostopic type /[illegible] turtlesim1/turtle1/pose
→ turtlesim/Pose

rostopic type / turtlesim1 / turtle1 / cmd_vel
→ geometry_msgs/Twist

(ros_topic info)

= JAVA

topic = classe
msg = attribut

le middleware converti n'importe quel language
// le crée pour nous
c'est de la traduction

ros msg

rosmsg show turtlesim/Pose
→ float32 x
→ float32 y
→ —
} = attributs

rosmsg show geometry_msg/Twist
→ geome. — msg/Vector3 linear
float64 x
— y
— z
→ ge— angular
— x
— y
— z

ros bag

rosbag -h ← "help"

↳ a trave ~~play~~ play
↳ record

Enregistre et on rejoue une séquence de mouvement

rostopic info

publisher

subscriber

- " rosbag record -a " ←
- " rosbag play * 2023 - ... .bag " ← enregistré a la racine par

ros service

"rosservice list" → nom de tous les service"

"rosservice type /turtlesim1/spawn" → regarde le type de "spawn

↳ les services

"rossrv show /turtlesim/spawn" → affiche les "attributs" de la "classe" spawn

↳ type des services

" rosservice call /turtlesim1/spawn [tab]

```
↳ x : 2.0
  y : 3.0
  theta : 0.0
  name : "ninja_turtle"
  name : " "
```

→ Fait spawn une nouvelle tortue a la position donnée et a l'orienta-tion donnée avec un nom défini

Si on veut la kill on fait : rosservice call /turtlesim/kill "name: turtle2 "

même nom

On souhaite changer la trace de la tortue en rouge

"rosservice call /turtlesim1/set_pen "{r: 255, g: 0, b: 0, width: 5, 'off': 0}" "

```
rosparam get /turtlesim1/sim/background_b → 255
rosparam set /turtlesim1/sim/background_b 50
rosservice call /turtlesim1/reset
```

→ change le background en vert foncé

ros service robot

" ros service call /controller_manager/list_controllers "

→ affiche les controleurs tel que avec tous les paramètres

→ gripper_current_limit
→ joint_state_controller → peut lire un URDF et positionne tous les corps rigides dans ROS
→ torso_controller
→ arm_independance_controller
→ arm_controller
→ mobile base_controller

catkin_make
pour compiler le package et le CMakeList.txt.xml

export ROS_IP

1C gazebo

ps aux | grep gz
kill -9 PID gz
(kill -9 PID ros)

(charge l'annuaire)
export ROS_MASTER_URI="https://localhost:11311"

env | grep ROS → pour voir la version des workspace ROS

roslauch