Handling of Bluerov Robot

marine mechatronics course (mundus master MIR)

Short description

The Bluerov architecture includes a Raspberry Pi 3 and a pixhawk board connected by USB.

The OS of the Rasp is a linux Ubuntu 16.04.2 LTS with ros kinetic. It contains several ros packages for the device drivers (USB cam or raspicam, DVL, scanning sonar) and mavros/mavlink packages to deal with the communication with the pixhawk.

The pixhawk software has been crosscompiled from ardupilot, whose source code is available on the Internet. It has been slightly modified to avoid multiple checking (e.g. ground station id, etc.).

How to connect to the robot

* Connect the USB cable from the blue box FXTI interface to your computer. Check that the USB cable is connected to the blue box and that the yellow cable from the robot is also connected to the blue box.

* Create a network connection on your PC (usb-Ethernet interface) :

manual IP : 192.168.254.15

netmask : 255.255.255.0 (24)

check it with the *ifconfig* command on a console (terminal).

* Configure environment variables inside you .bashrc file (located in your working directory, e.g. /home/xxx, where xxx is your login) :

source /opt/ros/melodic/setup.bash # or source /opt/ros/neotic/setup.bash
source ~/catkin_ws/devel/setup.bash
export ROS_MASTER_URI=http://192.168.254.15:11311
export ROS_HOSTNAME=192.168.254.15
export ROS_IP=192.168.254.15

Once .bashrc file has been modified, you need to run new terminals or to execute a bash command in your current terminals to take into account the new variables. You can check them with, for instance, :
echo $ROS_MASTER_URI
=> should give 192.168.254.15
* run *roscore* on one of your consoles, and leave it open. Check that the master's IP is the IP of your PC (192.168.254.15) in the messages displayed on the screen.

Your PC will act as a master for ROS, and the robot is waiting for the master before running some device drivers (camera and mavros).

* ssh connection with the robot :

it can be useful to open a ssh connection with the robot :

ssh ubuntu@192.168.254.xxx

password is %ubuntu

xxx can be 10, 11, 12, 13, 14 or 16, depending on your robot.

Ping the IP addresses successively if you do not know the IP of your robot.

Checking topics

If everything has gone well, you must be able to have the list of topics on one of your PC's console thanks to the command :

rostopic list

check the messages of the pressure sensor :

rostopic echo /br5/mavros/imu/water_pressure

br5 is the namespace of the topics and services relative to your robot.

It can be br1, br2, br3,br4, or br5. Just check with the list of topics.

If there is no message scroll, it can be due to the activation of mavros intercommunication which has not been run properly (in case roscore was launched too late after the robot was started).

=> run the activation through the following command on one of the robot's console (open a console through ssh if not already done) – the command can be copy-pasted from the /etc/rc.local file that contains the commands that are run at robot's startup:

/home/ubuntu/catkin_ws/src/bluerov/launch/set_stream.sh

In the shell file, the stream rate of data can be modified (up to 50 Hz) (use nano editor to modify it if needed).

If nothing works,

* stop roscore on computer, relaunch it,

* open a ssh connnection on the robot's raspberry :

killall roslaunch

check there is no more ros nodes : *ps – A | grep ros*

re-launch mavros : *roslaunch mavros px4.launch*

* open a 2nd connection on the robots's raspberry :

/home/ubuntu/catkin_ws/src/bluerov/launch/set_stream.sh

or

roslaunch bluerov start_stream.launch

Leave these 2 sessions open.

Running the python progam

Install the package named autonomous_rov in catkin_ws/src.

This package contains a python file named *listener_MIR.py* (inside the *scripts* folder) that enables gamepad control and switch between 3 modes (manual control with gamepad, autonomous mode, and correction mode).

N.B. : if you use python2, rename *listener_MIR_python2.py* into *listener_MIR.py*

In catkin_ws, use *catkin build* to build everything.

* Connect the gamepad to your pc.

* check the gamepad with jstest or *jstest-gtk* (install it if not present).

* check the launch file *run_use_autonomous.launch* that is inside the launch directory of the package autonomous_rov, and modify the tag of the namespace group id if necessary, e g. if the topics are in the namespace br2, the tag must be modified as:

<group ns="br5">

=>

<group ns="br2">

* launch the file that runs the joystick node, the teleop node and the python program (*listener_MIR.py* file) :

roslaunch autonomous_rov run_use_autonomous.launch

* enable the thrusters thanks to the start button, disable with the stop button. Activate the manual mode wih the Y button. Test your robot with forward/backward, left/right rotating and up/down movements.

You will have to fill the correction mode with depth servo control lines and heading control lines of code according to the instructions of the practical work sessions.

Recording topic messages for data handling

* record the pressure measurements with :

rosbag record -O pressure.bag /br5/mavros/imu/water_pressure

* use bag2csv.py (or bag2csv_python3 if using python 3) file (inside *scripts* folder) to build a csv file from your bag file.

* print the data of the csv file with python or matlab.

It is also possible to plot the data with rqt_bag directly from the bag, but it less convenient.

* observe the data measurements to tune your control laws that you have to code inside the *listener_MIR.py* file.

Useful ros commands

rostopic list : give the list of topics

roscd name_package : change to package directory

rostopic echo name_topic : display the messages sent on the topic

rostopic hz name_topic : give the rate of topic messages

rostopic show type_message : give the fields of the message (use tab key to have the list of folders that contain message types)

rosnode list : give list of active nodes

rosbag record -O namefile.bag topic1 topic2 … : record topics in bag named namefile.bag

to get information on a command : *name_command -h*

rosrun name_package name_exec_or_python_file : execute an exec. or python file.

roslaunch name_package name_file.launch : run a launch file that can contain several nodes to be run.

Look at the documentation on the mooddle platform.

Camera testing :

raspicam

rosrun image_view image_view image:=/brx/raspicam_node/image _image_transport:=compressed

usb cam wide angle

rosrun image_view image_view image:=/brx/camera/image _image_transport:=compressed

→ replace x in brx by the right number.

Change parameters of the camera using : rosrun rqt_reconfigure rqt_reconfigure

Proper shutdown command

When finished with the robots, use a ssh connection to the Raspberry and execute :

sudo shutdown -h 0

Wait until the green light of the Raspberry has completely switched off before disconnecting the battery.