Package

Joy_Controller

The following package was created by the team **Nantrobot** of the *École Centrale de Nantes* to provide an easier testing hardware-software tool for future competitions and developments. It requires

- ROS
- package Joy
- package Geometry_msgs
- Thrustmaster Controller or equivalent

1 Introduction

During a robot development there are many routines, programs, and sensors to test. Therefore if we want to test them we would have to change the code, load it into the robot and restart it. The purpose of the package is to control the robot's direction, publish the controller's buttons and sticks, if the controller is on and some states for testing. Then with the states it's possible to switch these routines in real-time to make testing faster for developers and with the buttons to specific tasks.

2 Controller conventions

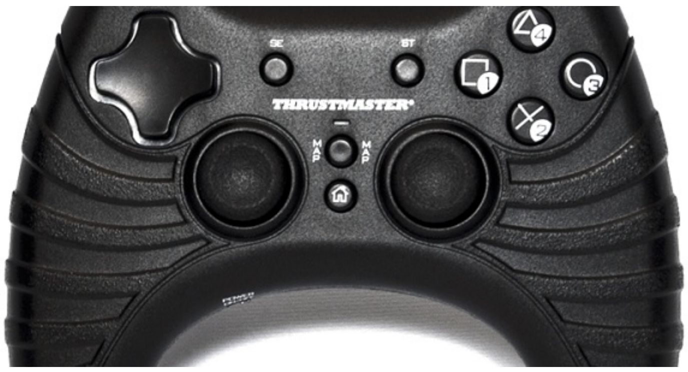

Figure 1: Controller front

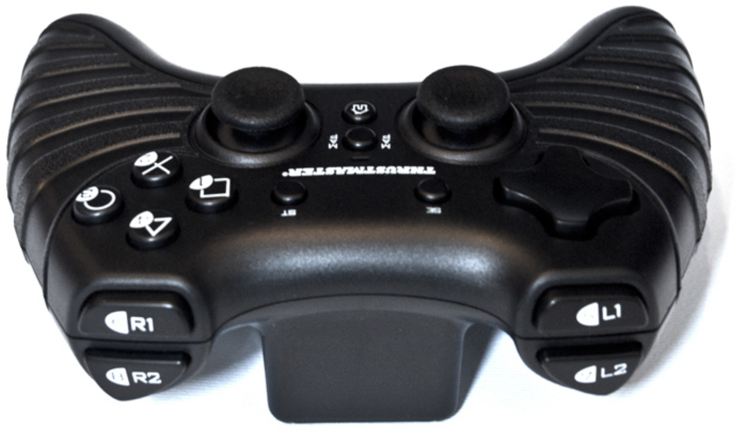

Figure 2: Controller back

We will name the controller components as follows:

1. square
2. cross
3. circle
4. triangle
5. L1
6. R1
7. L2
8. R2
9. SE
10. ST
11. Stick 1 pressed
12. Stick 2 pressed

The left Stick will be called **Stick 1** and the right stick **Stick 2**. The home button will serve as **on_state** for the controller and the pad will be the third stick but only with int values (**IntStick**).

3 Message managment

We'll have two messages published:

geometry_msgs/Point Message
As **PointCons**
This will be used to command the robot using the axes X and Y as for *Vertical* with the Stick 1 and *Horizontal* with the Stick 2.

joy_controller/joystick Message
As **JoyStick**
This will be used for publishing the states and the values of each component.

4 Using the controller

Now that we have our messages defined let's make some remarks:

- If the on_state is Off (0) then no change will be reflected on the values. Then if you press the cross and switch the controller off the cross will remain pressed. The States will remain the same as well. But for security reasons the *PointCons* values will always be reset so that the robot doesn't collide.
- The Stick 1 can only control the vertical axis (y) of *PointCons*.

- The Stick 2 can only control the horizontal axis (x) of *PointCons*.
- There two states:

State 1	State 2	Value
Square	L1	1
Cross	R1	2
Circle	L2	3
Triangle	R2	4

  And their prupose is to serve testing.

5 Final Remarks

The *launch File* is configured to use the sensor "$/dev/input/js1$" and it can be changed easily if required. The meaning of the states are completely up to the developer.

6 Contact

Any problem, improvements, suggestions to
dpalmac@dcc.uchile.cl
and
alexis.martin.2@eleves.ec-nantes.fr.