F : Wheel Calibration

The objective of the lab is to check the duckiebot motor is working, and make the car could go straight well.

本次實驗目的是讓我們小鴨車的馬達能夠正常運作，並且可以控制車子走“直線”。

Hardware and Software Setup

Hardware :

Duckiebot with joystick dongle.

Software :

Make sure your laptop can connect to your duckiebot.

Overview

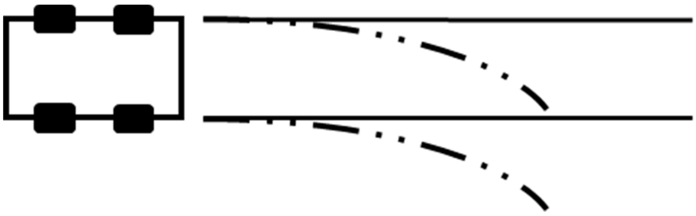
After completing this lab you will

- understand how to control the duckiebot by joystick.
- be able to use simple ROS command such as roslaunch, rosservice.
- learn how to perform wheel calibration

完成這次實驗後你將

- 了解如何使用搖桿控制小鴨車
- 學會使用簡單的 ROS 指令，例如：roslaunch、rosservice.
- 學會如何校正車輪的路徑

Why do we need wheel calibration?

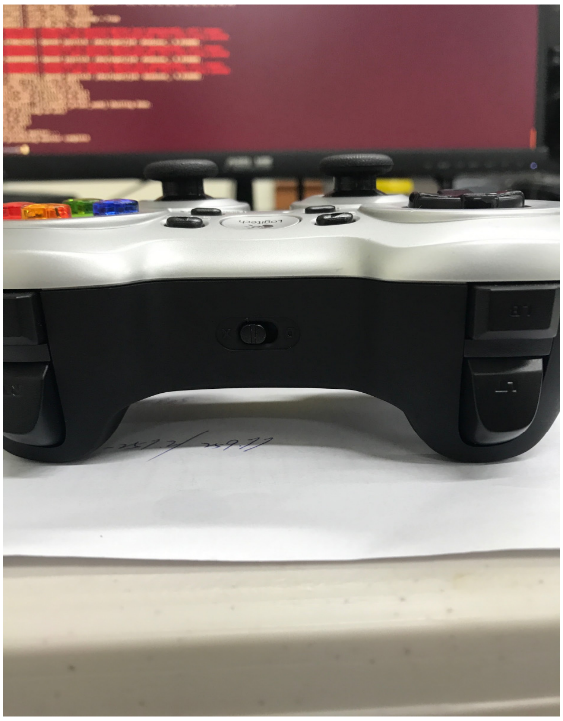


There are some situations in which duckiebot need to go straightly; for instance, performing joystick control or lane following.

Topics and Activities

Topic/Activity 1 Run the duckiebot by joystick controlling

First, Switch the switch behind the joystick to “X”, like the figure below



On your **duckiebot**, run the command below,

設定 ROS 環境並啟動 joystick.launch (請在**小鴨車**上執行)

```
duckiebot $ cd duckietown
```

Set the ROS environment

```
duckiebot $ source environment.sh
```

Set the ROS Master

```
duckiebot $ source set_ros_master.sh robotname
```

Launch the Joystick

```
duckiebot $ roslaunch duckietown joystick.launch veh:=robotname
```

At this point you can control the car by **joystick.**

Check point :

joystick control should follow this pattern:

- push left joystick forward : duckiebot go forward
- push left joystick backward : duckiebot go backward
- push right joystick leftward : duckiebot turn counter-clockwise
- push right joystick fightward : duckiebot turn clockwise

If your duckiebot don’t follow this pattern, check the connection of motor’s wire to motor hat is correct.

執行 **joystick.launch** 檔案可以讓我們使用搖桿去控制車子，輸入指令後請確認以下幾點：

- 推左搖桿向上：小鴨車前進
- 推左搖桿向下：小鴨車後退
- 推右搖桿向左：小鴨車向左旋轉
- 推右搖桿向右：小鴨車向右旋轉

Leave this terminal opened and open another terminal to do the next task.

別關掉剛剛操作的終端機，請緊接著完成接下來的實驗。

Topic/Activity 2 Perform wheel calibration

On your **duckietop**, communicate to your duckiebot as master:
(make sure your laptop is connect to your duckiebot wifi)

設定 ROS 環境並將 master 設定在小鴨車上。

(請在**電腦**上執行，並且注意電腦是否已經連上小鴨車的 wifi)

Set the ROS environment and ROS Master

```
laptop $ cd duckietown
laptop $ source environment.sh
laptop $ source set_ros_master.sh robotname
```

Still on your **laptop**, these two setting below are to set your inverse kinematics:
Set the trim, which can let duckiebot go straightly.

```
laptop $ rosservice call /robotname/inverse_kinematics_node/set_trim -- trim_value
```

Set the gain, which can let duckiebot go faster or slower.

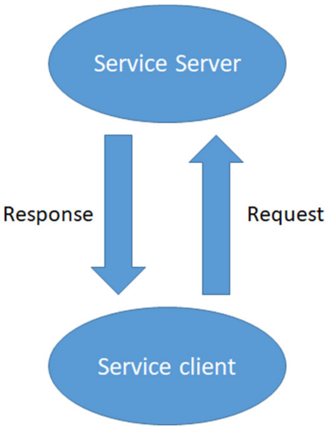
```
laptop $ rosservice call /robotname/inverse_kinematics_node/set_gain -- gain_value
```

Note : Trim and Gain value could be various from negative (below 0) to positive.
For example : -0.05 or 0.5

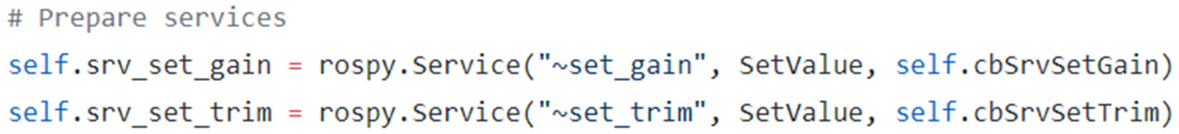
在這裡主要是為了校正車輪路徑，為了小鴨車以穩定的速度且準確的方向行駛，而參數 **trim** 可以調整左右輪子馬達的輸出差，使車子以直線行駛；參數 **gain** 可以調整左右輪的馬達輸出，也就是小鴨車的行駛速度。 請注意：兩個參數皆可以為負數。

What is Service in ROS?

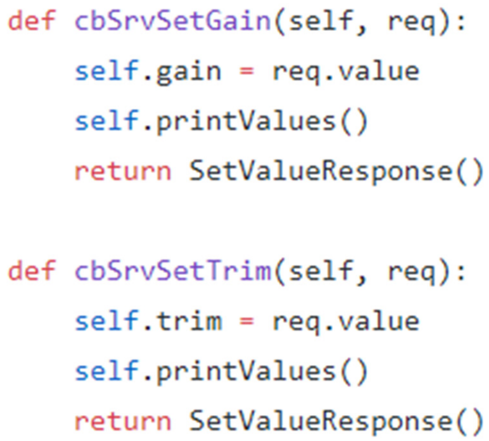
As its name is, it's just like a service which can get the information from client, and then do specific function in the server.



Example : (In inverse_kinematics_node.py)



```
# Prepare services
self.srv_set_gain = rospy.Service("~set_gain", SetValue, self.cbSrvSetGain)
self.srv_set_trim = rospy.Service("~set_trim", SetValue, self.cbSrvSetTrim)
```



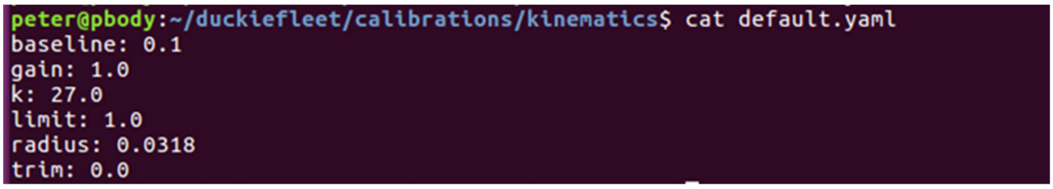
```
def cbSrvSetGain(self, req):
    self.gain = req.value
    self.printValues()
    return SetValueResponse()

def cbSrvSetTrim(self, req):
    self.trim = req.value
    self.printValues()
    return SetValueResponse()
```

you can use **rosservice list** to show all of service node offered in system now.

After your duckiebot could go straight well, save the calibration value:
laptop $ rosservice call /robotname/inverse_kinematics_node/save_calibration

Check point:

You should see a yaml file called robotname.yaml under the
duckiebot $ cd ~/duckiefleet/calibrations/kinematics/ folder in your duckiebot.
If not, you didn't do finish topic2.



```
peter@pbody:~/duckiefleet/calibrations/kinematics$ cat default.yaml
baseline: 0.1
gain: 1.0
k: 27.0
limit: 1.0
radius: 0.0318
trim: 0.0
```

最後可以透過指定路徑位置的檔案來確認是否完成車輪馬達的校正。

Assignment Tasks

(Please test the following tasks on **Jigsaw Mats**)

Task 1 Adjust parameter to match specification

- Spec.1 - Velocity : Approximately **30 cm/s**
- Spec.2 - Go straight : Please control duckiebot only with left joystick (i.e., you can only let duckiebot go forward or backward), and move your Duckiebot **straightly** for two meters.

Reference

Duckiebook: https://docs.duckietown.org/DT19/index.html