

第二章、Duckietown 電腦端之環境架設

林哲民

1 應用之工具安裝及網路架設

我們需要一個可以直接開發 Duckietown 之環境，因此我們選擇在電腦上使用一些工具作為 Duckietown 之開發環境，如此可以省去在機器之使用者介面上之額外花費。

1.1 虛擬電腦軟體：VirtualBox

Duckietown 之程式運作所使用的系統作業軟體是 Linux Ubuntu 14.04，考慮到現今大多數人都使用 Windows、IOS 系統，因此我們選擇以虛擬電腦軟體為主體來進行 Duckietown 之運作環境。

VirtualBox 是一套免費開放原始碼的虛擬電腦軟體，讓你在原有的系統架構下再建立出一台或是多台的新電腦。且可以在虛擬電腦裡安裝不同的作業系統，或是進行軟體測試，最重要的是在虛擬電腦裡所進行的動作皆不會影響或干擾到原有的電腦。接下來，我們會教學如何設定 VirtualBox。

步驟一、下載 VirtualBox

官方網站 (<https://www.virtualbox.org/wiki/Downloads>) 有提供載點根據你筆電的作業系統進行下載。



已知有支援的作業系統：Windows 7/10 64 bit; Ubuntu 14.04.3 32/64 bit; Mac-Book Pro (mid-2014)

步驟二、取得 VirtualBox 映像檔

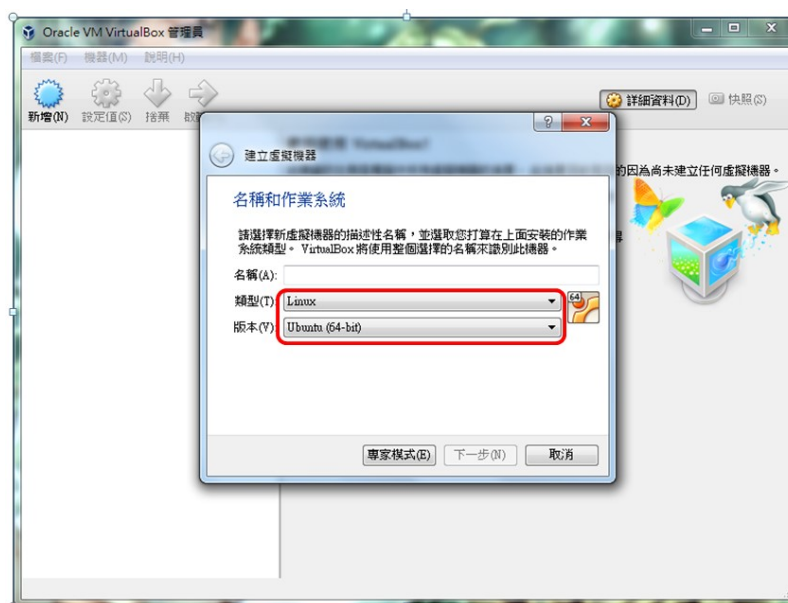
針對 Duckietown，我們已經準備相對應之映像檔供學生使用，連結如下：

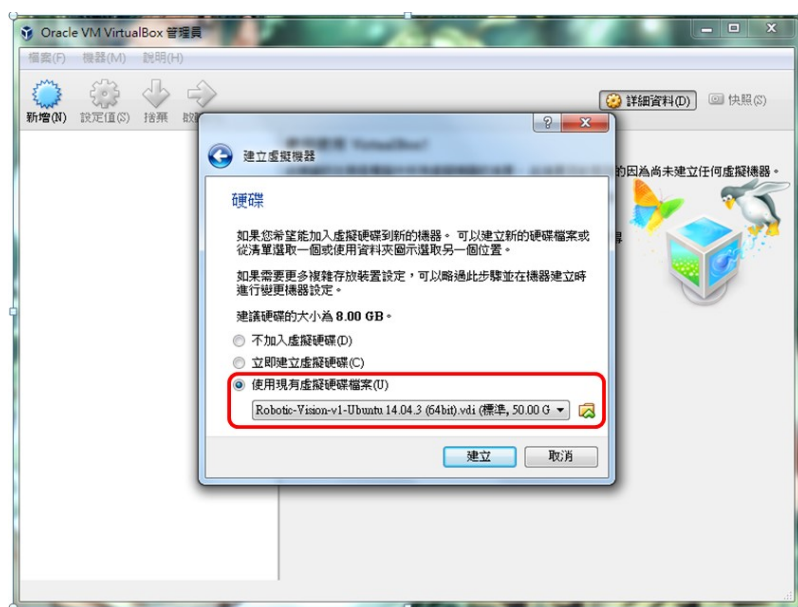
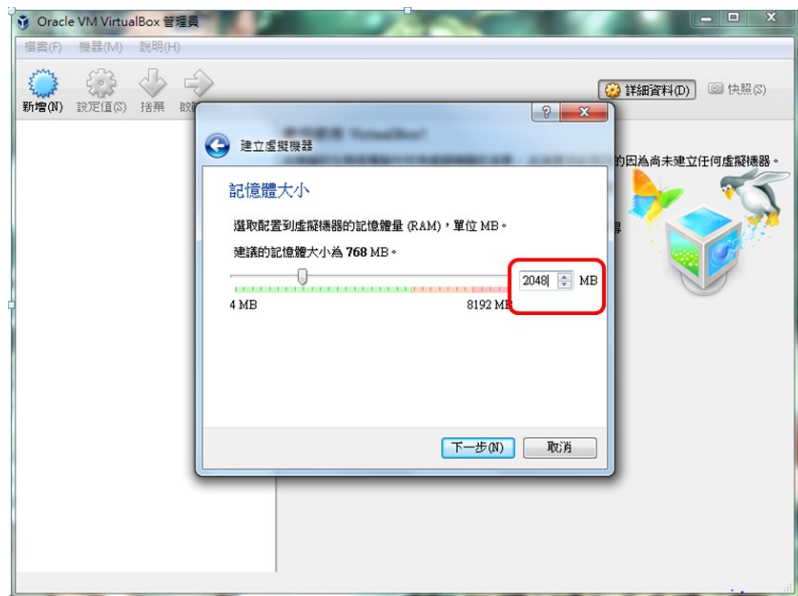
robotic-vision-v2 (in progress) : <https://drive.google.com/open?id=0B6vRmsV8B-ToQTJ3UmRzVHlyYXc>

此映像檔已包含必要之函式庫、LCM、ROS indigo、Anaconda (python 2.7) 及 Duckietown。

步驟三、VirtualBox 之設置

新增 -> Linux -> 64bit -> 記憶體大小 2G -> 選擇映像檔 (.vdi)





步驟四、VirtualBox 網路設定

為了讓虛擬機器能連線到樹莓派，你需要沿著以下步驟去設定網路: 設定值 -> 網路 -> 橋接介面卡，當你使用公共網路若是無法使用橋接介面卡時，切換成 NAT。

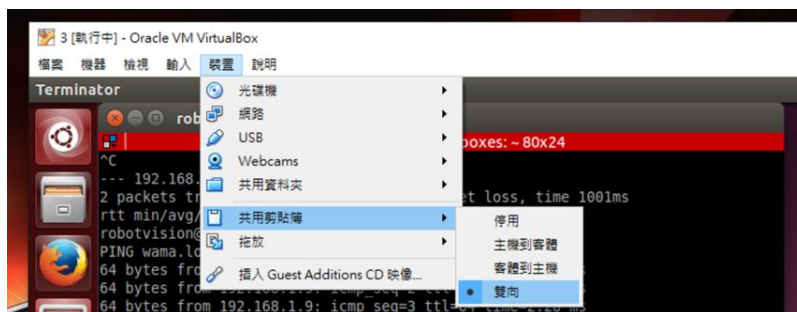


步驟五、啟用 Linux 及剪貼簿設定



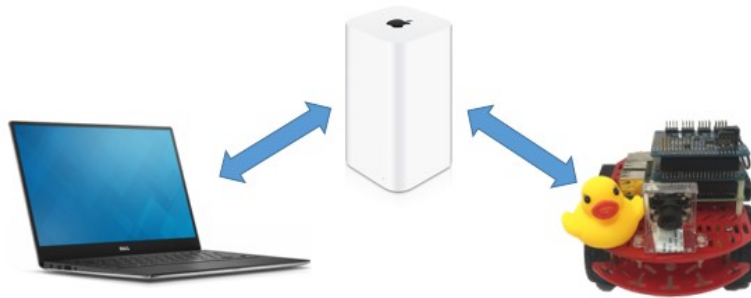
Username/password > robotvision / assistiverobotics

啟動 Linux 後，開啟剪貼簿之雙向功能，以方便之後使用。



1.2 機器端網路設定

網路設定方面，由於我們再進程式編譯會以電腦進行遠端操作機器端上之程式撰寫、運作。主要以 Wi-Fi 路由器為溝通媒介，因此我們必須先確認機器端已經連線到 Wi-Fi 路由器上。



先將機器以 HDMI 連接到外接螢幕並開啟電源，每一次更換無線網卡，都必須重複此步驟。

先確認網卡是否正常運作：

duckiebot \$ lsusb

```
ubuntu@test:~$ lsusb
Bus 001 Device 007: ID 046d:c21f Logitech, Inc. F710 Wireless Gamepad [XInput Model]
Bus 001 Device 005: ID 148f:5370 Ralink Technology, Corp. RTS370 Wireless Adapter
Bus 001 Device 004: ID 046d:c52b Logitech, Inc. Unifying Receiver
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp. SMSC9512/9514 Fast Ethernet Adapter
Bus 001 Device 002: ID 0424:9514 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
ubuntu@test:~$
```

接著，設定無線網路所連線之 Wi-Fi 路由器：

duckiebot \$ sudo vim /etc/wpa_supplicant/wpa_supplicant.conf

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
    ssid="assistive-robotics"
    scan_ssid=1
    psk="boce1112015"
    priority=10
}
```

根據你的 Wi-Fi 路由器更改以下兩行對應你的分享器

ssid="Wi-Fi 帳號"

psk="Wi-Fi 密碼"

修改完畢後，依步驟刪除 70-persistent-net.rules 再進行重啟：

duckiebot \$ sudo rm /etc/udev/rules.d/70-persistent-net.rules

duckiebot \$ sudo reboot
重啟完畢後，進行網路測試：
duckiebot \$ ping google.com

```
ubuntu@test:~$ ping google.com
PING google.com (64.233.188.101) 56(84) bytes of data:
64 bytes from tk-in-f101.1e100.net (64.233.188.101): icmp_seq=2 ttl=47 time=8.00 ms
64 bytes from tk-in-f101.1e100.net (64.233.188.101): icmp_seq=3 ttl=47 time=8.57 ms
64 bytes from tk-in-f101.1e100.net (64.233.188.101): icmp_seq=4 ttl=47 time=6.63 ms
```

如果你連上 google，那就完成這個部份了。

1.3 電腦端網路設定

完成機器端之網路設定後，我們必須進行電腦端之網路設定。主要必須預留 IP 位址給予機器端，以確保每一次都能順利以電腦進行遠端操作。

步驟一、紀錄機器端之 MAC address

在機器端先進行 IP 設定之顯示：

duckiebot \$ ifconfig

```
ubuntu@duckiebot:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr b8:27:eb:bd:7c:b2
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:112 errors:0 dropped:0 overruns:0 frame:0
          TX packets:112 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:8840 (8.8 kB)  TX bytes:8840 (8.8 kB)

wlan0     Link encap:Ethernet  HWaddr 00:1d:43:10:1c:9f
          inet addr:10.0.1.78  Bcast:10.0.1.255  Mask:255.255.255.0
          inet6 addr: fe80::21d:43ff:fe10:1c9f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:110752 errors:0 dropped:15607 overruns:0 frame:0
          TX packets:72475 errors:0 dropped:4 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:168207680 (168.2 MB)  TX bytes:7671027 (7.6 MB)
```

此範例下，MAC address 是 “00:1d:43:10:1c:9f”

步驟二、進入 Wi-Fi 路由器設定介面

此範例中，我們使用的是 TP-Link 之路由器。不同之分享器有不同之 IP。

打開一個瀏覽器輸入網址 “192.168.0.254”，進行登入。

使用者名稱: admin 密碼: admin

需要驗證

http://192.168.0.1 要求提供使用者名稱和密碼。

您與這個網站建立了非私人連線。

使用者名稱：

密碼：

登入

取消

步驟三、進行 IP 預留之設定
“進階設定” -> “DHCP” -> “位址保留” -> “新增”

狀態

基本設定

快速設定

無線運作模式設定

網路

無線

進階設定

DHCP

DHCP 設定

DHCP 用戶端清單

位址保留

轉送

安全性

靜態路由

IP QoS

IP 和 MAC 繫結

動態 DNS

維護

系統工具

新增或修改位址保留項目

MAC 位址：

保留的 IP 位址：

狀態：

啟用

儲存

返回

MAC 位址請填步驟一你紀錄之 MAC address，IP 之位址則為 192.168.0.XXX，可依自己喜好設置，確保不要跟別其他裝置 IP 對沖。完成後儲存。

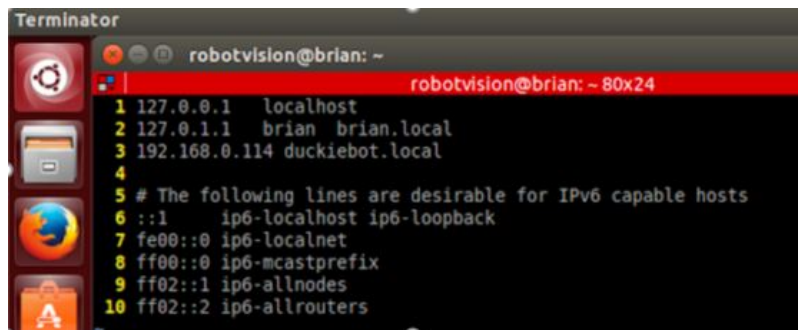
步驟四、重啟 Wi-Fi 路由器及機器端
請手動重啟你的路由器，並同時重啟機器端：
duckiebot \$ sudo reboot

2 程式環境架設

在正式開始軟體部分前，我們必須先確定能透過 Wi-Fi 分享器以電腦端去對機器端進行遠端操作。首先先把機器端電源開啟並將電腦端 Ubuntu 開啟。

2.1 遠端操作機器

啟用 Ubuntu 後，我們須先將前一部分預留之機器端 IP 加到 VirtualBox 中：
\$ sudo vim /etc/hosts



```
Terminator
robotvision@brian: ~
robotvision@brian: ~ 80x24
1 127.0.0.1 localhost
2 127.0.1.1 brian brian.local
3 192.168.0.114 duckiebot.local
4
5 # The following lines are desirable for IPv6 capable hosts
6 ::1 ip6-localhost ip6-loopback
7 fe00::0 ip6-localnet
8 ff00::0 ip6-mcastprefix
9 ff02::1 ip6-allnodes
10 ff02::2 ip6-allrouters
```

在/etc/hosts 中加上一行：192.168.0.XXX 你的車名.local

接下來，透過 ssh 來進行遠端操作：

\$ ssh ubuntu@your_duckiebot.local (此處 your_duckiebot 為你的車名)

ssh 是一個可以讓我們使用筆電遠端登入機器，並去操控機器執行指令

2.2 程式環境掛載及彙編

3 ROS 架構介紹及應用

ROS 又稱作為 RobotOS，是專門為了機器人軟體開發所設計出的一套作業系統。他是一個開源的系統，任何人都可以在 ROS 上機器人相關之功能、系統。且其在系統間之溝通，是一個極為優秀之跨平台軟體通訊機制，藉由不同節點間之溝通去達到最佳效果。此外，在 ROS 本身就蘊含龐大的 Library，其中不論是感測器資料、2 或 3D 模擬介面、特定演算法資料庫等等，都廣為開放在 ROS 平台中。因此，使用者可以輕鬆地去使用其中之工具於研究之機器人平台上，並加以延伸應用。

3.1 ROS 架構解析

在這一章節，將針對 ROS 中主要的各個運作項目作解釋。讓學生了解整體 ROS 是如何運作，各節點間如何溝通以及各個節點中之內容包含了哪些東西。以下針對 ROS 架構中之各個項目進行分析，包含了 Package、Node、Topic、Master、Messages、Parameter、Launch。

ROS Package :

ROS Package 的概念就是一個概括所有程式驅動、溝通橋樑及程式碼的集合體，是一個完整的程式組織。舉例來說，若今天要做一個人臉辨識之專題，所有的跑動程式碼、參數及相關之檔案都會放在一個名為 Face Detection 的 Package 裡。因此，在架構一個完整的系統時，必須建立一個屬於該系統之 Package。

ROS 在建立 Package 上很容易，只需陸續在添入一些功能就好。

舉例來說，以下資料夾均為 Duckietown 中 (catkin_ws/src) Package

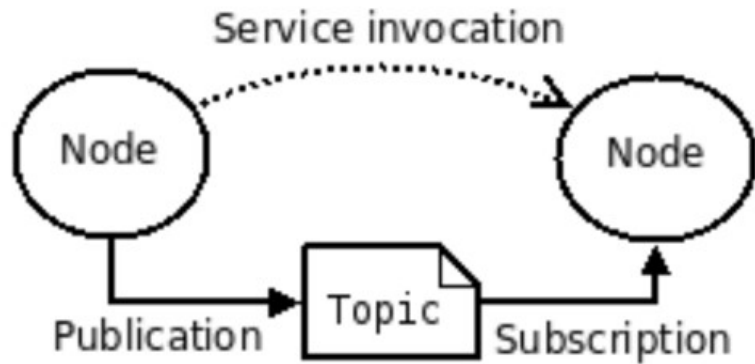
```

_attic      duckietown_demos      ground_projection      mocap_2016      spring2016
adafruit_drivers  duckietown_description  indefinite_navigation  navigation_util_1  spring2016_nctu
adafruit_imu      duckietown_logs      access_intersection_control  parallel_autonomy  stop_line_filter
apriltags_ros      duckietown_msgs      access_joy_mapper_or_menu  pi_camera      veh_coordinator
arg_nctu      duckietown_unit_test  lane_control_3  pkg_name      vehicle_detection
car_supervisor      f1      Processing      lane_filter_menu  rostopic_example  visual_odometry_line
dagu_car      f23-LED      joint8dof      line_detector      scene_segmentation
duckiebot_visualizer  f4-devel      joint8dof      localization_its_failed  simcity_pipe
duckietown      fsm      robot_pose_ekf      mdoap      slam

```

ROS Node Topic :

ROS Node 在 ROS 中就是扮演著主要運行之核心程式碼的角色，每一個 Node 都是獨立運算之程式。Node 一般可由 C++ 或 python 構成，主要程式內容極為整個程式所使用到之功能、演算法等。而在不同的 Node 之間，Topic 擔任資訊傳遞之角色。最主要是以 Publish/Subscribe 進行多個 Node 之間的訊息傳遞。在一個大系統中會概括許多個 Node，而 Topic 即支援多對多之間的通訊。



舉例來說，Duckietown 中每個 Package 也都概括許多 Node：

```

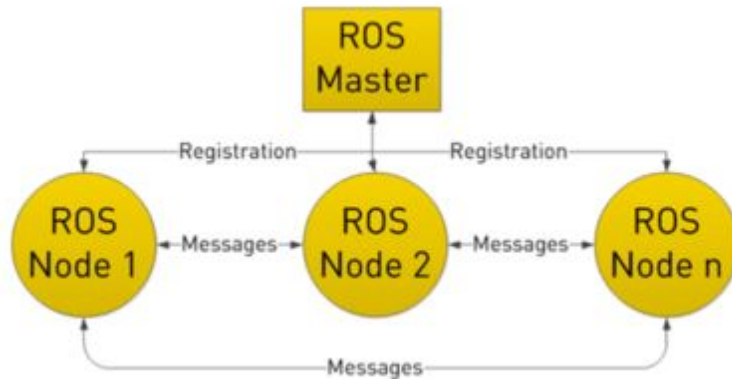
hchengwang:/Volumes/NICK-DATA/duckietown/catkin_ws/src/lane_filter/src$ ls
lane_filter_node.py      lane_filter_node_inverse.py      lane_pose_visualizer_node.py
lane_filter_node_Yinverse.py

hchengwang:/Volumes/NICK-DATA/duckietown/catkin_ws/src/apriltags_ros/apriltags/src$ ls
Edge.cc      GLineSegment2D.cc      etl.cc      Homography33.cc      Segment.cc      TagFamily.cc
FloatImage.cc      Gaussian.cc      joint8dof      MathUtil.cc      rite failed      TagDetection.cc      UnionFindSimple.cc
GLine2D.cc      GrayModel.cc      Quad.cc      TagDetector.cc

```

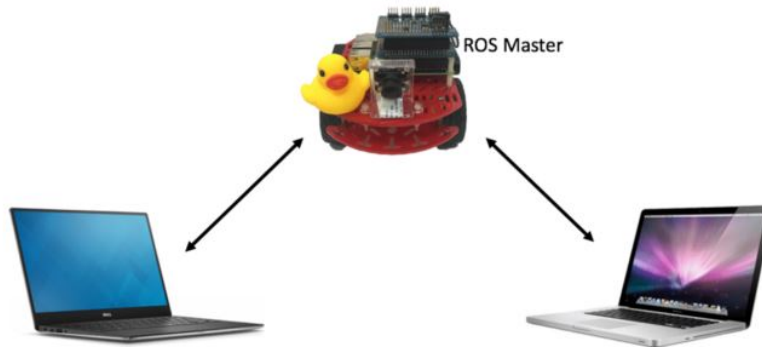
ROS Master :

在前面提到 Node 之間皆藉由 Topic 去進行訊息之傳遞，而 Master 則是負責管理所有 Node 之間的溝通。Master 最大的功能就是將不同 Node 之間的 Publisher 與 Subscriber 以 Topic 進行連接，以確保他們的訊息能夠確實的連接起來。因此可知，Master 主要是管理 Node 間該如何做連接。



而在多台 Machine 同時運作時，我們也可透過 Master 來進行多台 Machine 間之溝通。舉例來說，我們把機器與電腦都設成同樣的 ROS Master，如此我們就可以進行機器與電腦之間的資訊傳遞。因此，在 Duckietown 運作前我們都會先把 Master 設定好，確保電腦與機器間能互相傳遞訊息。

(舉例：`$ source set_ros_master.sh`)



ROS Message :

傳遞所用 Topic 之內容即為 Message 所構成及定義的，而不同的訊息傳遞也對應不同的 type，像是 bool, string, float32 等等。Message 就是溝通及傳遞之最基礎的訊息定義，Duckietown 中之 Message 都以 .msg 檔案定義。

```

AntiInstagramHealth.msg
AntiInstagramTransform.msg
AprilTagDetection.msg
AprilTagDetectionArray.msg
AprilTagsWithInfos.msg
BoolStamped.msg
CarControl.msg
CoordinationClearance.msg
CoordinationSignal.msg
DuckieSensor.msg
FSMState.msg
KinematicsParameters.msg
KinematicsWeights.msg
LEDDetection.msg
LEDDetectionDebugInfo.msg
LEDInterpreter.msg
LanePose.msg
ObstacleImageDetection.msg
ObstacleImageDetectionList.msg
ObstacleProjectedDetectionList.msg
ObstacleType.msg
Pixel.msg
Pose2DStamped.msg
Rect.msg
Rects.msg
SceneSegments.msg
Segment.msg
SignalsDetection.msg
SourceTargetNodes.msg
StopLineReading.msg
StreetNameDetection.msg
StreetNames.msg
TagInfo.msg
ThetaDotSample.msg
Trajectory.msg
Twist2DStamped.msg
Vector2D.msg
VehicleCorners.msg
VehiclePose.msg
Vsample.msg
WheelsCmd.msg

```

ROS Parameter :

每一個 Node 中之程式演算都一定會使用到特地功能之參數，譬如鏡頭畫素、馬達轉速等等。而在 Duckietown 中所有的參數檔都會統一歸類並放置 duckietown/catkin_ws/src/duckietown/config，以方便隨時替換參數值。

ROS Launch :

Launch 檔為 XML 檔，其主要功能是将 Node 組織化並定義好每個 Node 間之通訊及 Parameter(參數)。當你啟動一個 Launch 時，他會啟用所有該檔中所定義的 Node。是一個用來運作所有程式碼之工具，與驅動類似。而 Duckietown 中之 Node 都是依靠 Launch 來進行定義並運作的。

```

apriltags.launch
avoid_obstacles.launch
calibrate_turn.launch
camera.launch
closed_loop_navigation.launch
f4_demo.launch
ground_projection.launch
indefinite_nav_calibration.launch
indefinite_navigation.launch
intrinsic_calibration.launch
joystick.launch
joystick_camera.launch
joystick_camera_led.launch
joystick_dagu.launch
joystick_direct.launch
joystick_direct_camera.launch
joystick_old.launch
joystick_plus_led.launch
joystick_state.launch
kinematics.launch
kinematics_sync.launch
lane_controller.launch
lane_filter.launch
lane_filter_2csv.launch
led_emitter_detector_interpreter_joy.launch
led_emitter_detector_joy.launch
line_detector.launch
localization.launch
navigation.launch
navigation_graph.launch
odometry_training_pairs.launch
open_loop_intersection_controller.launch
parallel_autonomy.launch
rostopic_echo.launch
static_obstacle_detector.launch
stop_obstacles.launch
test_camcalib.launch
vehicle_camera_detection_filter_control.launch
vehicle_detection.launch
vehicle_detection_filter.launch
vehicle_detection_filter_control.launch
vehicle_detection_navigation.launch
vicon_learning.launch
visual_odometry_april_tags.launch
visual_odometry_line.launch
wheels_driver.launch

```

3.2 機器端之 ROS 應用