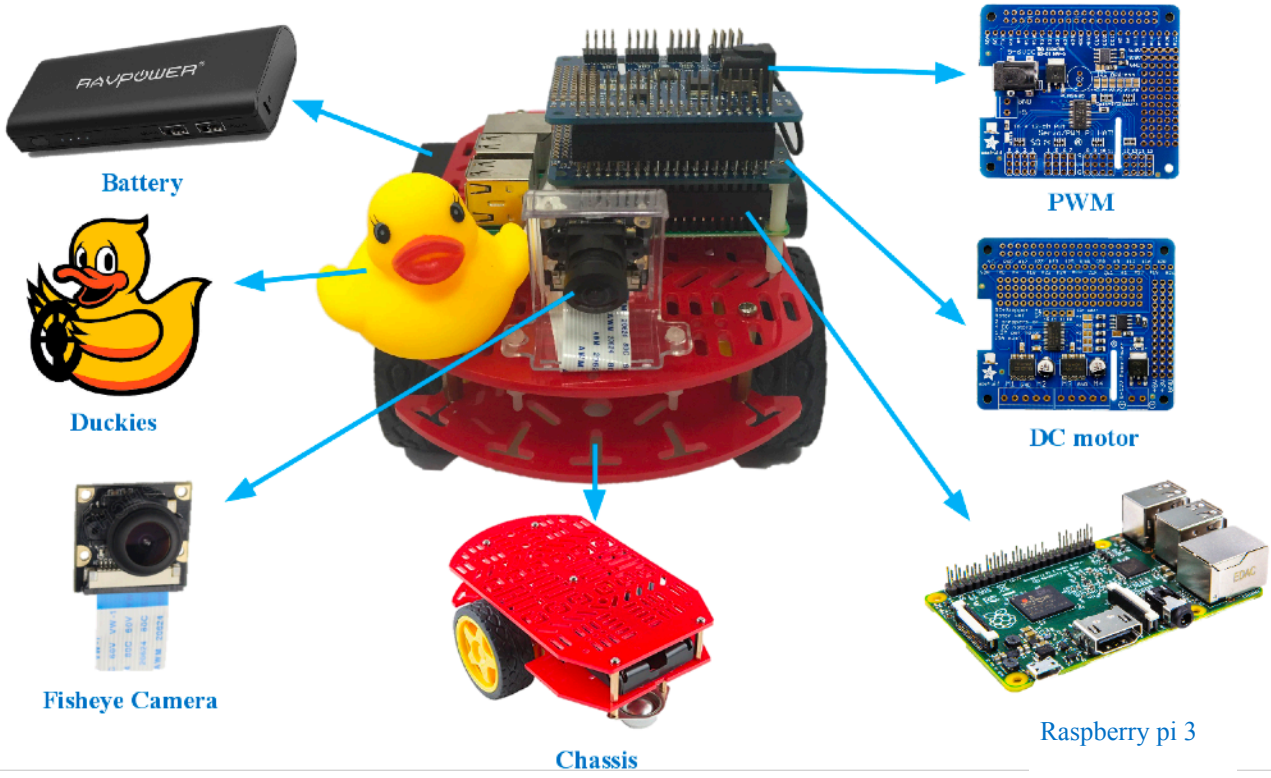
第一部分 Duckietown 鴨子城 與 Duckiebot 小鴨車

• 小鴨車介紹：

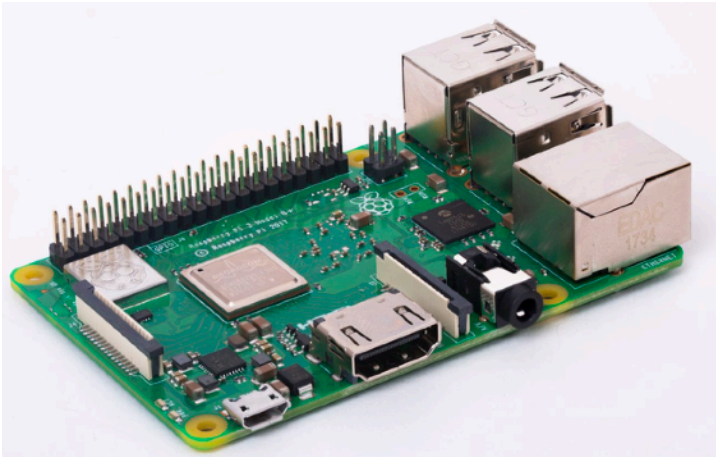
小鴨車是我們在鴨子城裡主要的機器人，我們會利用他做自動道路駕駛以及各式各樣的實驗。「機器人」是一個能以一定速度與精準度完成一個或若干任務的機器。而機器人主要由三個部分組成：感測、核心運算、運動

首先我們來介紹鴨子車的硬體架構，並且也帶大家看這台機器人的「感測、核心運算、運動」分別在哪裡！



核心運算：樹莓派 Raspberry Pi 3 Model B +

樹莓派 是一塊嵌入式板子，你可以把它看作是一台小電腦，上面一樣有中央處理器 (CPU)、記憶體 (RAM) 等等。就像電腦一樣，必須運行作業系統才可以使用，我們會在上面運行 Linux Ubuntu16.04 的作業系統。



因為穩定的設計、強大的社群，樹莓派擁有非常多愛好者。如果上Google搜尋「樹莓派」的話，馬上就能看到許多應用與他人的分享，如果用英文搜尋就更不用說了，官方的官方網站、社群、大到你逛三天三夜的看不完！

機器人的核心運算就是機器人的大腦，我們會在裡面寫下程式碼，讓機器人可以照所想的進行感測、規劃、執行。這其中就包含非常豐富的知識，人類從二次大戰後開始的電腦科學都囊括在裡面，如果你有興趣，歡迎加入交大電機跟我們一起學習研究！

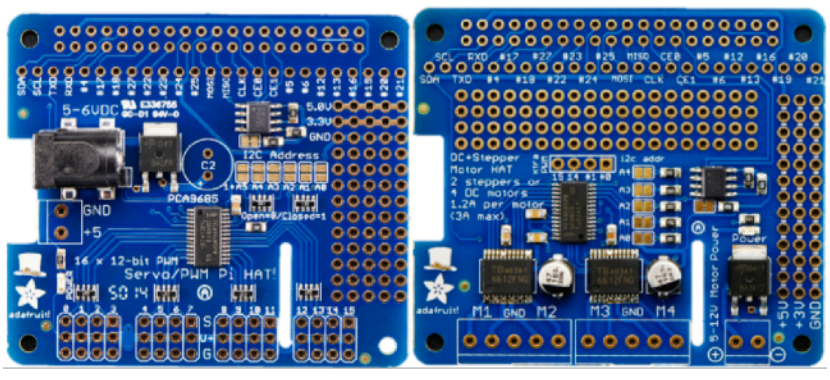
感測器：RPi 鏡頭 (G)

這款鏡頭可以支援所有版本的樹莓派，其魚眼鏡頭的設計可以讓視角寬度達 160 度，最高解析度達 2592x1944 或 1080p。一個感測器對機器人來說非常重要，他是決定機器人如何知道環境變化的關鍵，進而讓核心運算做出正確的決定。

機器人做感測的方法其實非常多種，從非常高階的雷達、光達、甚至較低階的顏色感測器、觸碰感測器、光感測器都可以是機器人感測的一部分。相機 Camera是一種非常常見、非常普遍、也是最直觀的感測器。他就像是我們人類的眼鏡、賦予機器人“視覺”。許多「電腦視覺」的技術就是從這裡開始。而小鴨車也用了非常多經典的電腦視覺技術達成自動駕駛的能力。

運動：PWM 板及馬達驅動板

PWM 板的目的是要用 PWM 的技術控制馬達驅動。PWM 是一種利用數位訊號模擬類比訊號的方式去控制 馬達轉速的技術。我們使用的是 Adafruit 16-Channel PWM / Servo HAT

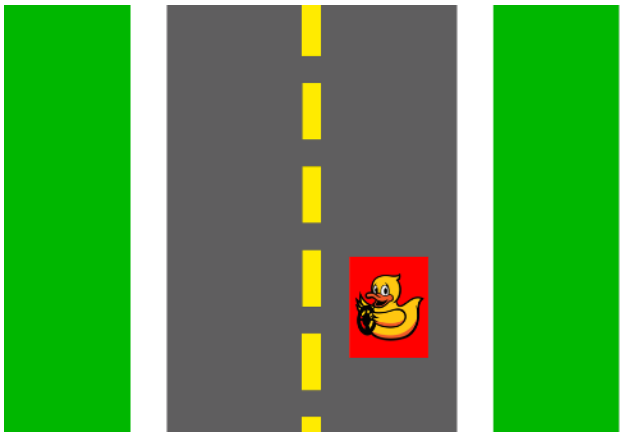


(左為 PWM 板/右為馬達驅動板) 。而馬達、馬達控制器，就是小鴨車「運動」的部分。當小鴨車從相機看到環境、核心運算決定處理資訊、作出決策、就換馬達做出相對應的運動。小鴨車的運動是以速度與角速度做描述。

而事實上機器人的運動不只是馬達移動而已，包括手臂的運動、或是其他身體機構的運動都屬於機器人的運動。甚至以廣義來說，一台Google home、Amazon Alexa 用語音做表達、電腦螢幕顯示訊息、都可以算是「運動」，也就是機器人所做的反應。

• 小鴨車自動駕駛原理、流程圖

這裡我們要來解釋一下在 Duckietown 裡、小鴨車做自動駕駛的原理與流程圖。

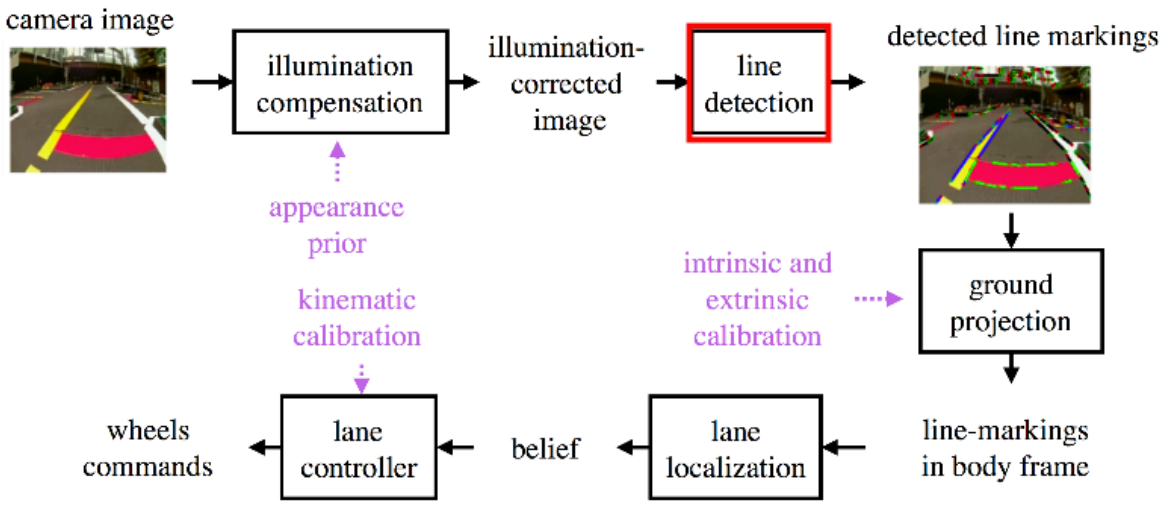


我們可以看到，一個鴨子城 Duckietown 是用許多黏了白線與黃線的巧拼組成，這個模組模仿真實道路的樣子。而小鴨車在上面行走時，我們有一個大前提：所有小鴨車都是「右駕」(靠右行駛)。

而在這樣的前提下，我們設計起小鴨車的策略就簡單多了，你會發現，只要我們讓白線在小鴨車的右邊、黃線在小鴨車左邊就可以了！另一方面來說，我們也可以用白線與黃線來"定位"小鴨車在道路中的位置。

所以整個流程就變成：用黃線白線定位小鴨車在馬路中的位置 -> 決定小鴨車要右轉、左轉或直走。

自動駕駛流程圖：



用上述的流程 (定位在馬路中的位置 -> 決定左右or直走)，我們用上面的流程圖，更詳盡解釋自動駕駛詳細的原理。

在鏡頭得到圖片 (camera image) 之後，這些圖會先利用程式調整光線對比 illumination compensation，讓小鴨車可以更容易看清、區分黃線、白線、紅線。

接著我們就會去偵測圖裡面的線段 (圖中藍色、綠色的部分)，也就是道路黃線白線“邊邊”的線段。這些線段能幫助我們“定位”小鴨車。

接著偵測出來的線段，會利用相機校正後得到的參數，將線段在照片中的“像素”位置，轉換成在真實世界中，以相機為原點的三維座標位置(x, y, z)。

接著利用這些線段回推車子的位置 (專業用語是 Pose”姿態”)。我們就能知道小鴨車在實際道路的位置。

最後經過運算，決定左轉、右轉、直走後，再給馬達做相對應的動力。

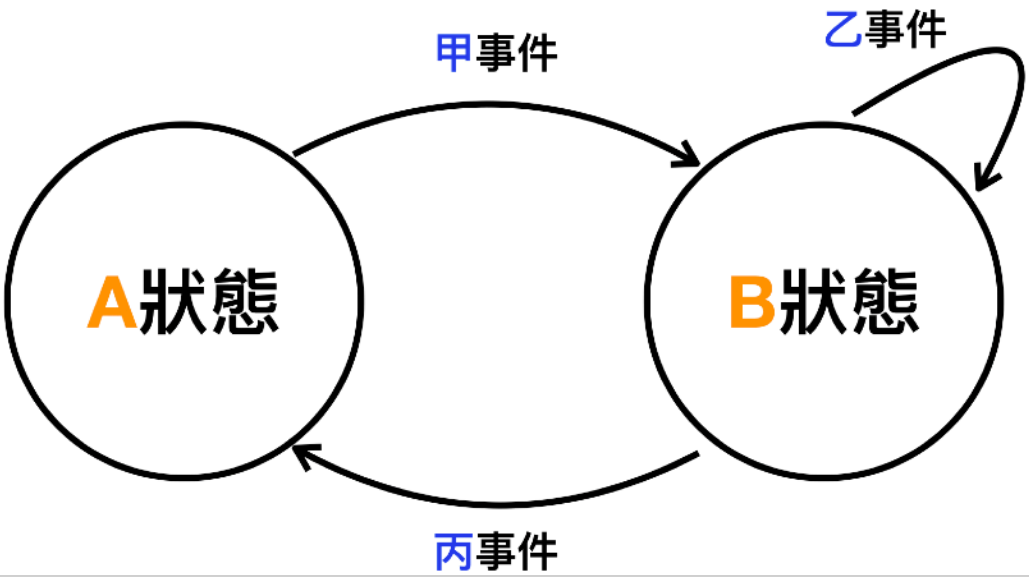
進階資料：

- **機器人的有限狀態機 Finite State Machine(FSM)**

什麼是有限狀態機 Finite State Machine？：

有線狀態機就是一個“有限” (不是無限、可以數、你心中想的那個有限)、“狀態” (有很多不同的狀態、情況、情景，看你想怎麼形容)、機。以實際例子來解釋有限狀態機是最好理解他的方法

舉例：試想以下狀況，有兩個狀態：A狀態、B狀態。以及三種事件：甲事件、乙事件、丙事件。

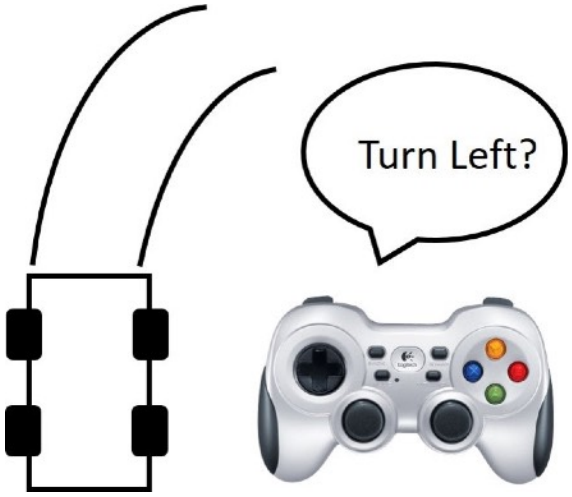


當機器人在A狀態時，若發生甲事件，機器人就會轉換到B狀態。在B狀態時若發生丙事件，就會再回到A狀態。如果在B狀態時發生乙事件，就會“再”到B狀態。

這就是一個最簡單的有限狀態機了！

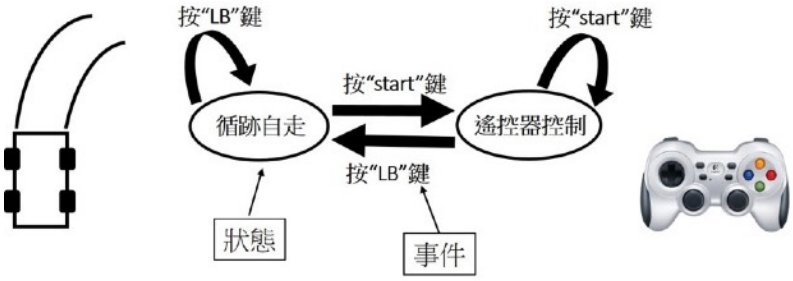
為何需要有限狀態機？：

首先設著想一下下圖的情況，如果現在有一台車子放在軌道上，照道理它會循跡自走，所以他應該要右轉， 但是這時我們又用遙控器叫它向左轉，踏它到底該選擇循跡自走向右轉還是選擇聽從遙控器的指令向右呢? 於是這時候我們就需要有限狀態機來幫我們，有限狀態機



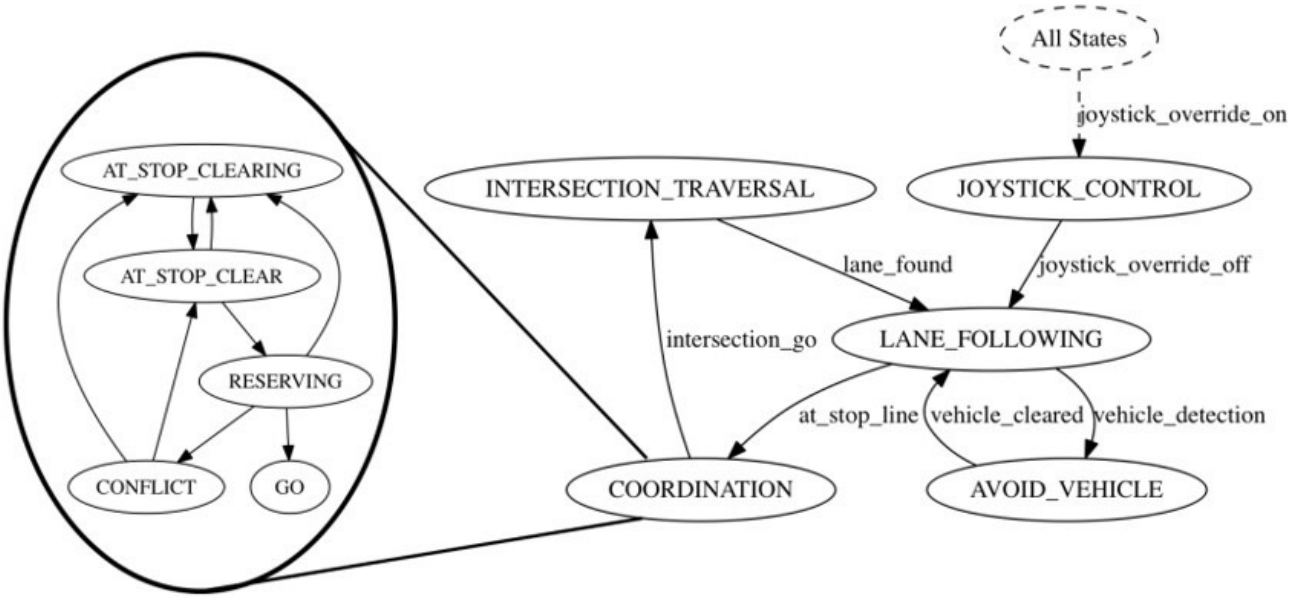
主要由”事件”和”狀態”組成，也就是當我們遇 到一個特定的事件，它就會跳到特定的狀態，藉此車子就會知道他這時到底該循跡還是該聽從遙控器。

以 duckietown 為例，我們只要”按下 LB 鍵”這個事件發生，車子就會跳到”循跡自走”這個狀態，而當我 們有”按下 start 鍵”這個事件發生，車子就會跳到”搖控器控制”這個狀態，因此藉由有限狀態機，車子就 不會感到困惑說到底該向左還是向右，因為它會遵循事件而跳到特定的狀態。



小鴨車自動駕駛 完整的有限狀態機示意圖：

當然，有限狀態幾可不是每次都這麼簡單。小鴨車也是利用有限狀態機才得以完成自動駕駛的任務。以下是小鴨車做自動駕駛時，他的有線狀態機。有興趣想理解更深入知識的同學可以參考下圖，有問題也都可以跟老師、助教詢問喔！


All States
joystick_override_on
AT_STOP_CLEARING
AT_STOP_CLEAR
RESERVING
CONFLICT
GO
INTERSECTION_TRAVERSAL
JOYSTICK_CONTROL
lane_found
joystick_override_off
intersection_go
LANE_FOLLOWING
at_stop_line
vehicle_cleared
vehicle_detection
COORDINATION
AVOID_VEHICLE

第二部分 用Python與Jupyter Notebook控制小鴨車

- **Python 與 Jupyter Notebook 介紹：**

Python:

Python 是現在非常廣用的程式語言之一，不管是在資料科學、機器學習、機器人領域都能看見他得蹤跡。

相對於其他程式語言而言，它具有較易上手且較易閱讀的特性。 Python 語言特別強調其可讀性，讓程式碼可以輕易地被他人理解。

另外，Python 與 C、C++、Java 等語言很不同的一點是，Python 是一種直譯式的語言，意思是不需要先編譯 (Compile) 在執行，而是一邊執行一邊編譯。目前，Python 也被廣泛地使用在資料科學裡面。

附註：大家常用的Arduino是使用一種跟C很像的語言。

附註二：Python是蟒蛇的意思

Jupyter Notebook：

Jupyter Notebook 是一個網路應用程式，有一點點像是Google doc。但最特別的是，他能讓使用者在裡面書寫程式碼，並且執行。

對於我們而言，他最方便的地方在於能將程式碼分成一個一個的區塊 (Cell)，並且分別觀看每個區塊的輸出結果。因為這樣的分辨性，讓我們可以分別檢查每段的程式，是否有期望中的輸出抑或是挑出其中的 Bug。

不過要注意的是，Jupyter Notebook 只能使用直譯式語言，例如 Python 等等。在使用上要特別注意。

• Jupyter Notebook 基本操作：

以下是Jupyter Notebook快捷鍵，各位同學可以自己玩玩看喔！

上下方向鍵：選擇不同的Cell
Enter：編輯該Cell
Shift+Enter：執行該Cell
a：往上加一個Cell
b：往下加一個Cell
dd：刪除該Cell

• Python 基本語法：

當Jupyter Notebook開啟後，你會看到這個畫面。我們來解釋接下來的語法。

印出一個句子：(shift+Enter 執行)

1. Python Basic

print a sentence

```
In [1]: print "Hallo world and Duckietown!"
        Hallo world and Duckietown!
```

使用print的話可以印出你想要得句子，使用Shift+Enter執行。你可以發現，Jupyter notebook會直接把結果印出來。

變數與加法：(shift+Enter 執行)

Initialize a variable and addition

```
In [2]: duckiebot = 0
        a = 1
        total = duckiebot + a

        print "The total is:", total
        The total is: 1
```

在程式語言中，我們可以定義一個新的“變數”，你可以任意為這些變數取名字。圖中的“duckiebot”、”a”、“total”都是一個變數。在使用變數時，我們都會寫成 “變數 = ”，代表我們將這個變數“變成“某個數字。

另外，用加號的話可以把兩個數字或變數相加。再下面的格子試著複製貼上剛剛的程式碼，把加法變成減法試試看！

A small task - 1

```
In [3]: ### copy and paste the above cell and make it to subtraction

In [ ]:
```

乘法與除法：(shift+Enter 執行)

Multiplication and division

```
In [4]: duckies = 5
        students = 10
        total_amount = duckies*students

        print "How many duckies do we need?", total_amount

        duckies = 60
        students = 12
        duckie_per_student = duckies/students

        print "A student can get this amount of duckies.", duckie_per_student

        How many duckies do we need? 50
        A student can get this amount of duckies. 5
```

乘法與除法也非常簡單，乘法就是星號 * ，除法就是斜線 /。

定義函數：(shift+Enter 執行)

Function in Python

```
In [5]: ### A function is a part of code that can be called and reused whenever you want

In [6]: def addition(a, b):
            result = a + b
            return result

In [7]: ### How do we used a defined function?

In [8]: duckies = 5
        chickens = 10
        total_birds = addition(duckies, chickens)

        print "Here are the total birds we have.", total_birds

        Here are the total birds we have. 15
```

幾乎所有程式語言都可以定義函數 (function)，他是程式語言非常重要的一環，可以讓我們重複使用某一段程式碼。

在Python當中，你必須使用 “def” (代表definition) 來開始定義你的函數。你可以任意命名你的函數，並且決定有幾個值要被傳過來到 a、b。最重要的是，你可以定義函數的內容並且回傳你想回傳的值。

值得注意的是，在這個函數內的部分，必須在一行的開頭隔出「四個空白」，並且在“def”該行結束時，補一個冒號：。

for 迴圈：(shift+Enter 執行)

For loop

```
In [9]: ### A for loop is a "loop" which you can defined how many times you want to run it

In [10]: for duckie in range(5): # i = 0, 1, 2, 3, 4 in each loop
             print "Now the 'duckie' is", duckie

Now the 'duckie' is 0
Now the 'duckie' is 1
Now the 'duckie' is 2
Now the 'duckie' is 3
Now the 'duckie' is 4

In [11]: for duckie in range(2, 5): # i = 2, 3, 4 in each loop
             print "Now the 'duckie' is", duckie

Now the 'duckie' is 2
Now the 'duckie' is 3
Now the 'duckie' is 4
```

迴圈是另一個程式語言重要的概念。迴圈主要有兩種：for 迴圈 跟 while 迴圈。我們在這裡先介紹for loop, for 迴圈。

在 for 迴圈中，你可以定義一個”range”，那一部分的程式就會跑你指定的次數，並且其中的“變數”也會隨之增加。

值得注意的是，在這個迴圈內的部分，必須在一行的開頭隔出「四個空白」，並且在“for”該行結束時，補一個冒號：。

if-else 條件：(shift+Enter 執行)

if-else

```
In [12]: duckies = 5

         if duckies > 0:
             print "There are duckies."
         else:
             print "There are no duckies."

There are duckies.
```

if else條件是第三個程式語言非常常見的部分，我們可以設定當什麼樣的條件產生時就做什麼事，否則的話就發生另一件事。

值得注意的是，在這個if else條件內的部分，必須在一行的開頭隔出「四個空白」，並且在“if”該行結束時，補一個冒號：。

• 用Python讓你的小鴨車動起來！：

現在，我們要用剛剛學到的程式語言，讓你的小鴨車動起來！！！

因為這邊的程式碼部分與環境設置有關係，我們將會跳過那些部分並放在補充資料裡。

如何使用car_command 函數：

Examples of Using car_command

Ex1: Forward velocity = 1 for 0.75 seconds

In [ ]:
```
#car_command(1, 0, 0.75)
```

EX2: Turn with angular velocity = 4 for 1.0 seconds

In [ ]:
```
#car_command(0, 4, 1.0)
```

car_command函數，是我們驅動小鴨車的函數，使用者必須分別填入「速度 (velocity)」、「角速度 (angular velocity)」與「時間」，在執行時，小鴨車就會根據你的定義坐相對應的動作。

在這裡有兩個使用範例供大家參考，第一個是使用速度為1 (正數往前、負數向後)、時間為0.75秒的指令。第二個為角速度為4 (正數逆時針轉、複數順時針轉)、時間為1.0秒的指令。

定義自己的按鈕動作：

Define Joystick reaction

Here is where all magic happens, you can define your own buttons!!

In [ ]:
```
# button A
def button_A():
    # forward
    car_command(1, 0, 1)
```

In [ ]:
```
# button B
def button_B():
    # turn around
    car_command(1, 10, 1)
```

In [ ]:
```
# button X
def button_X():
    # for loop
    for i in range(4):
        car_command(1, 0, 0.5)
        car_command(0, 0.8, 0.5)
```

In [ ]:
```
#button Y
def button_Y():
    # put your creation!!
```

在這四個函數中，分別定義自己按該按鈕後想要做的動作，在執行後即可按按鈕、執行對應的動作！

按鈕A：用速度=1，往前1秒鐘。

按鈕B：用速度=1、角速度=10，往前1秒鐘。

按鈕X：用速度=1，往前0.5秒鐘，再用角速度為0.8，往前0.5秒鐘，連續四次。

按鈕Y：給你自己定義！！

定義完之後，請執行整份Jupter notebook檔案，並且開啟小鴨車 (可請助教幫忙)，即可控制你的車子囉！

Demo影片參考：https://youtu.be/XlGfy3l0X5s