

A blue and white robotic boat is shown on a body of water. The boat has a camera mounted on top and various sensors. The text "Rabboni Boat Control" is overlaid on the image. A small horizontal bar with green, white, and orange segments is located above the title.

# Rabboni Boat Control

Ubuntu & ROS

Developed in Python

# 器材清單

---


1. **Rpi4** (樹梅派第四代 - RAM需要8G 否則ROS可能無法編譯)
2. SD card (32G即可)
3. Windows 系統 (燒錄用)
4. 安裝 Linux 系統 (若要使用藍芽 Ubuntu 20.04, 若不需要 Ubuntu 18.04)
5. 相機 (具有usb接口)
6. 杜邦線
7. 無刷馬達
8. 電變
9. 絕緣膠帶
10. 三相金插
11. 端子臺
12. 降壓模組



# 大綱

---

- A. 燒錄 Rpi
- B. Ubuntu環境建置
- C. Linux 基本教學
- D. Vim 基本指令 (開發中常用到的編輯器)
- E. Python 基本語法
- F. Rabboni 程式解說
- G. 接線及安裝硬體



## A. 燒錄Rpi - 安裝作業系統

1. <https://opendarkbox.blogspot.com/2018/01/raspberry-pi-1.html>

>> 選擇 **Raspberry Pi OS (32-bit) with desktop and recommended software**

1. <https://opendarkbox.blogspot.com/2018/01/raspberry-pi-2.html>

>> 推薦用螢幕直接連



## A. 燒錄Rpi - 安裝 ROS (機器人控制系統)

1. 安裝 ros [ROSberryPi/Installing ROS Melodic on the Raspberry Pi - ROS Wiki](#)





## B. Ubuntu 環境建置

有兩種方案:

- a. 直接使用打包好的虛擬電腦 (virtualbox)
- b. 自行安裝
















## B. Ubuntu 環境建置 (a. virtualbox)

1. 安裝[virtualbox](#)
2. 下載打包好的[虛擬電腦](#)
3. 匯入 (**機器基礎資料夾** 請留**30GB**以上空間 如下頁)
4. 虛擬機名稱為 **vm**
5. 帳號名稱 密碼皆為 **ubuntu**

## 應用裝置設定

這些是包含在應用裝置的虛擬機器和匯入 VirtualBox 機器的建議設定。您可以透過按兩下項目來變更顯示的許多內容，並使用以下的核取方塊停用其它內容。

虛擬系統 1	
 名稱	vm
 客體作業系統類型	 Ubuntu (64-bit)
 CPU	2
 RAM	4096 MB
 DVD	<input checked="" type="checkbox"/>
 USB 控制器	<input checked="" type="checkbox"/>
 網路卡	<input checked="" type="checkbox"/> Intel PRO/1000 MT Server (82545EM)
 存放裝置控制器 (SATA)	AHCI
<input checked="" type="checkbox"/>  存放裝置控制器 (SCSI)	LsiLogic
 虛擬磁碟映像	Ubuntu-disk1.vmdk
 基礎資料夾	D:/virtualbox
 主要群組	/

機器基礎資料夾(M): D:/virtualbox

MAC 位址原則(P): 只包含 NAT 網路卡 MAC 位址

額外選項:  匯入硬碟磁碟機作為 VDI(I)

應用裝置未簽署

還原預設值

匯入

取消





## B. Ubuntu 環境建置 (b. 自行安裝)

在Ubuntu系統下執行:

1. 安裝 Rabboni [Hackmd](#)
2. 安裝 [ROS Melodic](#)
3. 自行跟著ROS官方教程做一個workspace
4. 將/Desktop (桌面上)的 /test 資料夾打開
5. 將 `auv_keelung package` 放入 workspace 裡面執行



## 編輯程式碼篇

C. Linux 基本教學

D. Vim 基本指令 ( 開發中常用到的編輯器

E. Python 基本語



## C. Linux 基本教學



## Linux 基本

1. **sudo** 代表是 super user 後面會用最高權限去執行指令
2. 按 **上** 可以跑出前面輸入過的指令
3. 按 **tab** 可以自動補齊
4. \* : 代表0到無窮多個任意字元 :
5. ? : 代表1個任意字元

2. ; : 代表1個任意字元



## ssh : 遠端連線工具 ( 可以遠端透過WIFI進入Rpi)

- `$ ssh <username>@<ip>`
  1. username 遠端的帳戶名稱
  2. ip 遠端電腦的ip
- 要連到同個網路下
- `ssh pi2@192.168.0.202`



## ls : list , 查看檔案及子目錄

● `$ ls [參數] [路徑]`

- 常用參數
  - a : all, 顯示所有檔案集目錄 , 包括隱藏檔案目錄
  - l : long, 顯示檔案的完整資訊
- example
  - `$ ls -al`
  - `$ ls -a`
  - `$ ls -l`



## cd : change directory , 移動進入資料夾

- `$ cd 路徑`
- `~` 代表home資料夾
- `.` 代表當前資料夾
- `../` 代表上層資料夾
- `/` 代表根目錄

### example

- `$ cd`
- `$ cd ~`
- `$ cd /`
- `$ cd Desktop`
- `$ cd .`
- `$ cd ../`



## cp : copy , 複製檔案或是資料夾

- `$ cp [參數] <要複製的檔案路徑>... <目標資料夾路徑>`
- 常用參數
  - `r` : 遞迴 , 複製整個資料夾的時候需要用
  - `v` : 顯示已複製的檔案
  - `i` : 若目標檔已經存在時 , 在覆蓋時會先詢問
  - `f` : 強制執行

### example

- `cp test.txt ../`
- `cp -r test_folder ../`





## scp : 傳輸遠端檔案

- `$ scp <File> <username>@<ip>:<path>`
- e.g. `$ scp test.py pi@192.168.0.102:~/Desktop/`
- e.g. `$ scp pi@192.168.0.102:~/Desktop/test.py ./Desktop/`
  1. 可以雙向互傳
  2. 利用SSH
  3. ip 後面要加:再接位置



## mv : move , 移動檔案或是重新命名檔案

- 移動：`$ mv [參數] <要複製的檔案路徑>... <目標資料夾路徑>`
- 重新命名：`$ mv [參數] <原檔名> <新檔名>`
- 常用參數
  - `v` : 顯示已複製的檔案
  - `i` : 若目標檔已經存在時，在覆蓋時會先詢問
- example
  - `mv ~/test.py ./src/`
  - `mv old.txt new.txt`



## man : manual , 說明書

- `$ man <指令名稱>`
- 按q離開

## cat : 將文件內容印在終端機上

- `$ cat <文件名稱>...`
- example
  - `cat .bashrc`



## **mkdir : make directory** , 創建新資料夾

- `$ mkdir <資料夾路徑>`
- example
  - `mkdir src`



**pwd : print work directory** , 印出目前工作目錄

- `$ pwd`

**ifconfig : 網路**

- 可用來查詢 ip



## apt-get : 套件管理工具

- 更新套件資料庫列表 : `sudo apt-get update`
- 升級套件並下載安裝套件 : `sudo apt-get upgrade`
- 安裝套件 : `$ sudo apt-get install <套件名>`
- 移除套件 : `$ sudo apt-get remove <套件名>`



## chmod指令：用來改變檔案權限

- 用3個2進位數字表示
  - `$ chmod 744 test.py`
- 用+, -配上r, w, x
  - `$ chmod +x test.py`



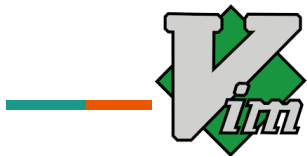
## D. Vim 基本指令 (開發中使用的編輯器)



## Vim （此專案大多使用vim去開發）

1. 純文字編輯模式軟體
2. 在Linux系統上被廣泛使用
3. 預設狀態是無法使用滑鼠，可使用插件去自由改變編輯模式
4. `vim test.py` （即可編輯test.py檔案）





最常使用的有**命令模式**（ 切換使用模式 ）**插入模式**（ 可編輯檔案 ）

- **i** : 進入編輯模式
- **a** : 在游標位置後進入編輯模式
- **Esc** : 取消指令或退出編輯模式
- **:<行數>** : 跳至第幾行
- **:wq** : 儲存並退出
- **:q!** : 不儲存強制退出
- **dNd** : 從游標開始刪除N行
- **V** : 選擇整行
- **v** : 選擇游標範圍
- **y** : 複製選取範圍
- **p** : 貼上剪貼簿內容
- **u** : 還原上一步



## E. Python 基本語法



# Python 教學 (此篇只有講解程式碼遇到的語法)

1. 變數

2. Operator

3. list

4. Loop

5. 自定義函式

6. import 套件



# Variable

宣告變數 (Python 不需要宣告變數型態) :

```
i = 1
```

```
str = 'hello'
```

```
a = []
```

```
b = true
```



# Operator

運算子	C	python
加	+	+
減	-	-
乘	*	*
除	/	/ (//)
取餘數	%	%
乘冪	pow()	**
單行註解	//	#



**list** (類似於更強大陣列，可自由改變長度)

`a = []` : 空list

`a[index]` : 讀取成員數值

`len(a)` : 長度讀取

`a.append` : 尾端加入成員



## `range()` 產生一個等差數列

- `range(起始值, 中止值, 增值)`
- **起始值**預設為0
- **中止值**不會包含進去等差數列內
- **增值**預設為1





# While Loop

`while` (判斷句) :

    程式主體

```
while (1): # 這樣可製造無窮迴圈
```

```
    print("hello")
```



## def () 自定義函式

```
def function(變數):  
    函式主體 (return)
```

EX:

```
def f(x):  
    return x*2
```



## For Loop

常和 `range()` 一起做使用

```
for i in range(2,10,2):  
    print(i)
```

會得到 2 4 6 8 (不包含中止值10)



## `import` 匯入函式庫

```
import rabbit as rab
```

- 通常寫於程式開始 匯入套件且可以自行命名
- 上面範例將 `rabbit` 匯入 並將其在這個程式內命名為 `rab`  
(簡短方便)

# F. Rabboni相關控制



ROS

- Talker Node: 負責傳送rabboni資訊(加速度，角速度)
  - USB
  - BLE
- Listener Node: 負責接收rabboni資訊並控制馬達

# Talker Node -- USB

```
1  #!/usr/bin/env python3
2
3  import rospy
4  from std_msgs.msg import String
5  from std_msgs.msg import Float32
6  import traceback as tb
7  import math
8  import time
9
10 from rabboni import Rabboni
11
12 rab = Rabboni()
13 rab.connect()
14
15 rab.set_sensor_config(2, 500, 5, 100)
16 position_lr = 0
17 position_fb = 0
18 preW_fb = 0
19 preW_lr = 0
20 timer = time.time()
21
```

1至10行--引用一些我們所需的library可以保持不變

12至13行--讓rabboni可以用USB的介面傳資料給電腦

15行--第一個參數為三軸加速度正負值範圍(參數範圍:2,4,6,8)

第二個參數為三軸角加速度正負值範圍(參數範圍:

250,500,

1000,2000)

第三個參數為rabboni蒐集資料的速度(參數範圍:

1,5,10,20,

40,50,100,200,500,1000)

第四個參數為threshold(可自行填參數)

16至19行--position\_lr為rabboni左右轉姿態, preW\_lr紀錄上一個左右轉姿態

position\_fb為rabboni前後轉姿態, preW\_fb紀錄上一個前後轉姿態

20行--timer負責記錄兩筆資料之間的時間

# Talker Node -- USB

```
22 def usb_custom_callback(status):
23     global position_lr
24     global position_fb
25     global timer
26     global preW_fb
27     global preW_lr
28     value = 0
29     ratio = 0.7
30
31     if(abs(preW_fb-status['Gyr'][0]) > 0):
32         position_fb = (1 - ratio) *
33             (position_fb + (status['Gyr'][0] * math.pi/180) *
34             (time.time() - timer)) + ratio * status['Acc'][1]
35     if(abs(preW_lr-status['Gyr'][1]) > 0):
36         position_lr = (1 - ratio) *
37             (position_lr + (status['Gyr'][1] * math.pi/180) *
38             (time.time() - timer)) + ratio * status['Acc'][0]
39
40     #Left and Right
41     if position_lr < -0.35:
42         s = 'right'
43         value = status['Acc'][0]
44     elif position_lr > 0.35:
45         s = 'left'
46         value = status['Acc'][0]
```

22行--負責判斷前後左右轉的function,參數不用動  
status['Acc']分別存取的是x,y,z軸之加速度  
status['Gyr']分別存取的是x,y,z軸之角加速度

23至27行--為了利用剛剛定義出來的變數

28至29行--value負責存當前rabboni測出來的值  
s負責存前後左右轉的字串

31至38行--運用complementary filter計算當前  
rabboni前後以及左右姿態

41至43行--當計算出的position\_lr小於-0.35代表現在

是向右轉，然後s為存right, value會存

當前

x軸的加速度

44至46行—類似41至43行



# Talker Node -- USB

```
47 #Forward and Backward
48 elif position_fb < -0.4:
49     s = 'forward'
50     value = status['Acc'][1]
51 elif position_fb > 0.4:
52     s = 'backward'
53     value = status['Acc'][1]
54 else:
55     s = 'stable'
56
57 pub1.publish(s)
58 pub2.publish(value)
59 print(s + ' ' + str(value))
60 timer = time.time()
61 preW_fb = position_fb
62 preW_lr = position_lr
```

47至54行--類似於41至43行，到要注意的是我們定義左右轉的優先序是大於前後轉，所以將position\_fb以elif的結構接在position\_lr後面，而當前後左右轉的條件都不符合時會將它定義成stable

57至58行--將s和value存下來的值publish到topic上

59行--把要publish的資訊print到螢幕上，以便debug

60行--重新計算時間

61至62行--紀錄上一筆資料以便下次計算

# Talker Node -- USB

```
64 if __name__ == '__main__':
65     rospy.init_node('talker', anonymous=True)
66     pub1 = rospy.Publisher('command', String, queue_size=10)
67     pub2 = rospy.Publisher('value', Float32, queue_size=10)
68
69     try:
70         rab.start_fetching_status(custom_callback=usb_custom_callback)
71         rab.polling_status()
72     except AssertionError: # 結束程式
73         print('Bye~!!')
74     except Exception:
75         tb.print_exc()
76     finally:
77         rab.disconnect()
```

64行--程式進入點，不用更改

65至67行--定義出node以及兩個publisher

69至71行--不斷的讀取rabboni測到的資料，並進入

22行的function

72至75行--例外處理，鍵盤按ctrl+c結束或拔掉USB線

76行--移除電腦與rabboni的連結

# Talker Node -- BLE

因為BLE只有在連線部分跟USB不同，所以只需要修改連線部分即可

```
12 rab = Rabboni(mode='BLE')
13 rab.scan()
14 rab.connect(mac_address='F6:20:ED:F3:5B:F2') #對應到Rabboni
15
16 rab.read_sensor_config()
17 rab.set_sensor_config(2, 500, 5, 100)
18 rab.read_sensor_config()
```

12行--因為rabboni預設是USB模式，所以要把mode  
調成BLE模式

13行--掃描附近rabboni裝置

14行--mac\_address後面要接rabboni背後的裝置代號

16至18行--固定寫法，只需要調整17行的參數

# Listener Node

```
1  #!/usr/bin/env python3
2
3  import rospy
4  from std_msgs.msg import String
5  from std_msgs.msg import Float32
6  import RPi.GPIO as GPIO
7  import time
8  import numpy as np
9
10 GPIO.setmode(GPIO.BOARD)
11 GPIO.setup(32,GPIO.OUT)
12 GPIO.setup(12,GPIO.OUT)
13
14 pR=GPIO.PWM(32,50)# 50hz frequency
15 pL=GPIO.PWM(12,50)
16
17 pR.start(7)
18 pL.start(7)
19
20 pR.ChangeDutyCycle(7)
21 pL.ChangeDutyCycle(7)
22
23 command = 'forward'
```

1至8行--引用一些我們所需的library可以保持不變

10至15行--設定腳位功能，我們使用的開發板為RaspberryPi4，所以pwm為12和32腳位，而馬達為50Hz，這些參數都依使用的器材作調整即可

17至21行--我們所使用的馬達以duty cycle為7不旋轉，這也依使用的器材調整即可

23行--command負責存取給馬達的命令

# Listener Node

```
25 def control(value):
26     global command
27     lr_strength = 1.4    #左右轉幅度，越大越強
28     fb_strength = 1.3   #前後強度，越大越強
29
30     if command == 'right':
31         pR.ChangeDutyCycle(7-lr_strength)
32         pL.ChangeDutyCycle(7+lr_strength)
33     elif command == 'left':
34         pR.ChangeDutyCycle(7+lr_strength)
35         pL.ChangeDutyCycle(7-lr_strength)
36     else:
37         pR.ChangeDutyCycle(fb_strength*value.data+7)
38         pL.ChangeDutyCycle(fb_strength*value.data+7)
39
40 def Command(data):
41     global command
42     command = data.data
43     print(command)
```

25行--傳送新的duty cycle給左馬達和右馬達的function，value即是talker傳出來的data

26行--為了讓此function使用先前定義出來的變數

27至28行--調整左右轉以及前後走的強度，數字越大越強

30至32行--接收向右轉的指令時，馬達要輸出的duty cycle

33至35行--類似於30至32行

36至38行--接收前後走的指令時，馬達要輸出的duty cycle

40至43行--更新要給馬達的指令

# Listener Node

```
45 def listener():
46     rospy.init_node('listener', anonymous=True)
47     rospy.Subscriber('command', String, Command)
48     rospy.Subscriber('value', Float32, control)
49     rospy.spin()
50
51 if __name__ == '__main__':
52     try:
53         listener()
54     except KeyboardInterrupt:
55         print("STOP")
56     finally:
57         GPIO.cleanup()
```

45至49行--定義出node以及兩個listener

51行--程式進入點，不用更改

52至53行--不斷讀取talker傳出來的資料

54至55行--例外處理，鍵盤按ctrl+c結束

56至57行--當先前定義的腳位清除

# G. 硬體接線和安裝

## G. 接線&組裝 注意：一定要檢查完所有接線無誤後，再接上電池

- 將所有零件接上端子臺
- 端子臺橫向兩個互通，垂直方向不通
- 紅接紅(正極)；黑接黑(接地)

註：可使用端子台的jumper使垂直互通





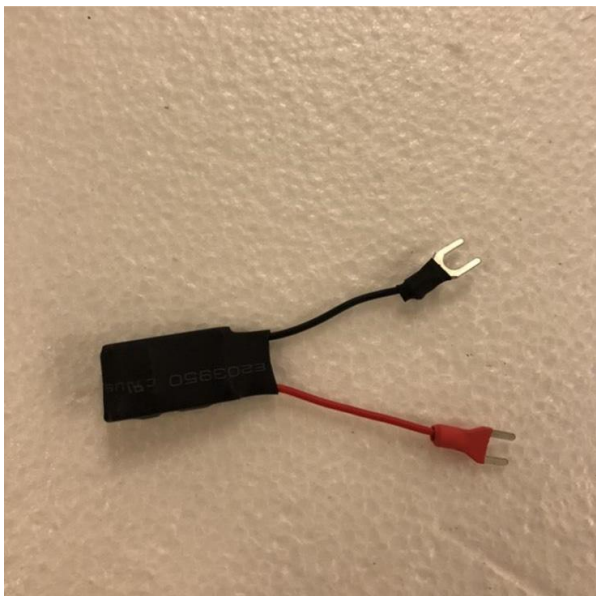
## G. 接線&組裝 **注意：一定要檢查完所有接線無誤後，再接上電池**

- 將電變的**黑、白線**分別接至Rpi上的**接地**及**訊號**腳位。
- 再將電變與馬達接上，需使用三相金插。
- 由端子台再接一降壓模組輸出**5V 3A**電源供Rpi使用。
- 確認馬達附近無東西碰到以及確認接線無誤後，接上電池，每次組裝前使用三用電錶確認端子台等等是否有短路狀況，避免危險（電變損毀也有可能造成短路）

註：人體接觸超過10mA電流就會造成危險喔！

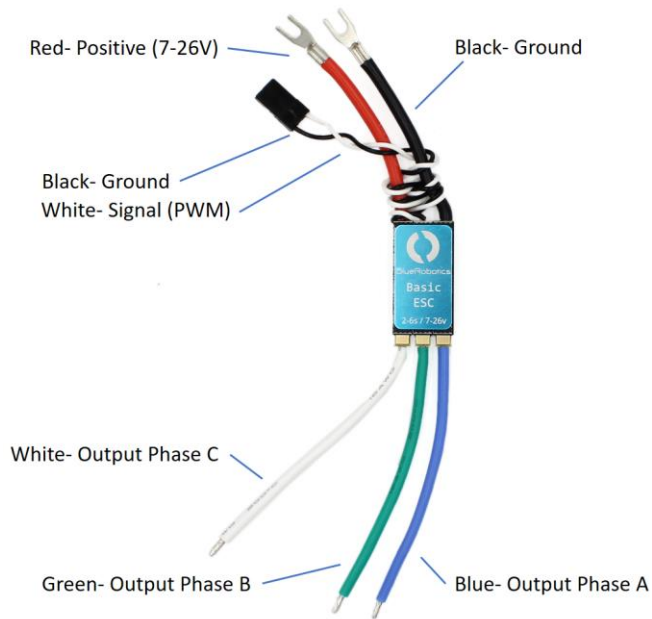


## 5V降壓模組



- rpi 的電供為5V, 所以我們須將鋰電池降壓後再輸出給rpi, 接線方法可以看第一張圖

# 電池、電變ESC、馬達

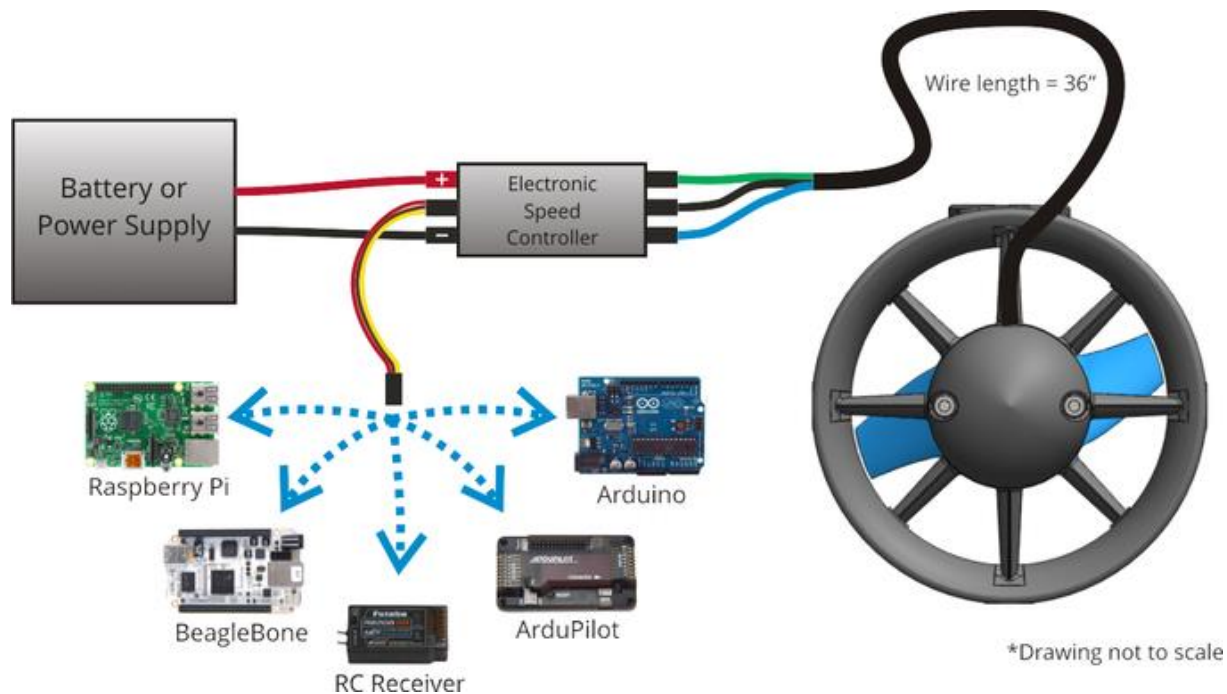


- 電變的選擇與馬達及電池需搭配好, 順序: 電池規格->電變->馬達
- 電池的C數指放電能力, 假如 2200mAh 20C, 電池的持續放電能力  $2.2 * 20 = 44A$ , 大於馬達電流即可使用, 若不足電池可能損毀 (過放膨脹)
- 馬達的最大電流也須比電變小

註: 馬達請選用防水的無刷馬達, 無刷馬達相較於其他種類的馬達有更高的防腐蝕及防水性

註: 電池容量請看mAh, 數字越大容量越大, 體積越大且續航力越強, 請應需求做選擇

## G. 接線&組裝 注意：一定要檢查完所有接線無誤後，再接上電池



# 此教程由 NCTU-AUV 團隊提供

