Topic#3 Lab4 實驗結報

姓名：蔡佩蓉　　學號：109511286

一、 實驗目的

本次實驗旨在開發一個 IoT 應用，透過樹莓派（Client）遠端控制 Turtlebot3 機器人（Server）的移動，同時利用超音波感測器監測距離，實現防碰撞機制。具體目標包括：

- 建立 TCP Socket 通信連接
- 遠端控制 Turtlebot3 的移動（前進、後退、左轉、右轉、停止）
- 在 Client 端使用超音波感測器監測距離，當距離小於 10cm 時停止機器人運動

二、 實驗過程 (Code + 說明)

Q1 ：做出防撞機制，遠端 Client 操控機器人

Server 端（新增的）code :

```
# Establish a TCP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
HOST = '192.168.185.42'
PORT = 3000
sock.bind((HOST,PORT))
sock.listen(5)
conn, addr = sock.accept()
conn.settimeout(0.5)
```

建立連線

(1) sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)：
socket.socket([**family**], [**type**], [**protocol**])
family → socket.AF_UNIX → 於本機端進行串接 (IPv4 本機)
→ socket.AF_INET → 於伺服器和伺服器之間進行串接 (IPv4 網路)
→ socket.AF_INET6 → 於伺服器和伺服器之間進行串接 (IPv6 網路)
type → socket.SOCK_STREAM → TCP
→ socket.SOCK_DGRAM → UDP
protocol → 串接協定，通常預設為 0

(2) sock.listen(5)：最多可接受 5 個 socket 連接

(3) conn,addr = sock.accept()：

- accept()用於接收連接，並回傳串接對象與 IP 位址資訊給 conn 和 addr
- conn 存串接對象，addr 存連線資訊

(4) conn.settimeout(0.5)：若 timeout 結束時未收到 data，回到上一步驟執行

```
if __name__=="__main__":
    if os.name != 'nt':
        settings = termios.tcgetattr(sys.stdin)

    rospy.init_node('turtlebot3_teleop')
    pub = rospy.Publisher('cmd_vel', Twist, queue_size=10)

    turtlebot3_model = rospy.get_param("model", "burger")

    status = 0
    target_linear_vel   = 0.0
    target_angular_vel  = 0.0
    control_linear_vel  = 0.0
    control_angular_vel = 0.0
    status=0
    try:
        print(text)
        while(1):
            try:
                # Receive data normally
                msg = conn.recv(1024)
                print(addr)
                print(msg)
                key = msg
            except socket.timeout:
                # Timeout occurred, do something
                continue
```

接收訊息，最多 1024 bytes
若超過 timeout，則不執行任何動作，進入下一次迴圈

(1) 此處 conn 為建立連線的對象

Client 端（新增的）code :

```
import RPi.GPIO as GPIO
import time
import sys, select
import socket

GPIO.setwarnings(False)

v = 343
TRIG = 16
E = 18

print '1'
GPIO.setmode(GPIO.BOARD)
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(E, GPIO.IN)
GPIO.output(TRIG, GPIO.LOW)
HOST = '192.168.185.42'
PORT = 2000
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect((HOST, PORT))
```

建立連線

```
while(1) :
    i, o, e = select.select([sys.stdin], [], [], 3)
    if(i):
        a = sys.stdin.readline().strip()
        client.sendall(a)
        print(a, type(a))
    else:
        print("Nothing")
    distance = measure()
    print(distance)

    if distance < 10:
        print("Distance < 10")
        client.sendall('s')
```

timeout 為 3 秒，若超過 3 秒仍未收到 input 的訊息，則不傳送值且 print(“Nothing”)

距離小於 10 的時候傳送資料 ‘s’

三、 問題及解法

問題 1: Socket 通信問題

解法：確保 Raspberry PI 和 Turtlebot 之間的 Socket 通信正常運作。需要檢查 IP 地址、端口號等設定。記得先啟動 server，再啟動 client。

*注意: ‘.bashrc’ 中的 export ROS_MASTER_URI 和 export ROS_HOSTNAME 的 VM 的 IP 需與 Socket 中的 Host (IP) 一致。

四、 心得

通過 Socket 通信，我們實現了用 Raspberry PI 遠端控制 Turtlebot 的全方位移動和防碰撞機制的功能。這次實驗不僅提升了我們對 IoT 應用的理解，也加深了我們對 ROS 和 Raspberry PI 的應用能力。這次實驗也讓我更加熟悉了程式開發過程中常見的問題和解決方法，提高了我的解決問題能力和程式設計技能。