通訊網路實驗 Lab5 Report

110511254 徐煜絨

一、實驗目的（第二部分實驗說明會有更詳細介紹）

設計並修改 turtlebot3_lab5 的 code，結合 lab1 所學做出防撞機制。Server 端設定在 VM，Client 端則是樹梅派與感測器，藉由在 client 端輸入 w, a, s, d, x 決定 turtle 的移動方向和速度，另外當感測器測到距離小於 10 時，turtle 會停止移動。

二、實驗過程（Code+說明）

Server

```
# Establish a TCP socket
HOST = '172.20.10.7'
PORT = 9324
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.bind((HOST,PORT))
sock.listen(5)
conn,addr = sock.accept
conn.settimeout(0.5)
```

建立 TCP 連線，HOST 是 VM 的 IP 位置。第四行 sock=socket.socket(…)代表資料傳送方式，socket.AF_INET 用於伺服器間的串接；第五行表示連線的 IP（server 的）和要使用的 PORT（必須和 client 一樣）；第六行 socket.listen(5)表示最多可以接受 5 個人連線；第七行用在伺服器端的接收，會接受連線並回傳對象 IP；第八行是若在 timeout 結束前沒有收到 data，會執行上一步驟。

```
try:
    print(text)
    conn,addr = sock.accept()
    print(addr)
    while(1):
        try:
            msg = conn.recv(1024)
            print(msg)
            # print(msg.decode())
            key = msg
        except socket.timeout:
            continue
```

msg = conn.recv(1024)中的 1024 代表可以接受 bit 數的最大值。

```
<launch>
  <arg name="model" default="$(env TURTLEBOT3_MODEL)" doc="model type [burger, waffle, waffle_pi]"/>
  <param name="model" value="$(arg model)"/>

  <!-- turtlebot3_teleop_key already has its own built in velocity smoother -->
  <node pkg="turtlebot3_teleop" type="turtlebot3_lab5" name="turtlebot3_teleop_keyboard"
output="screen">
  </node>
</launch>
```

Client

Client 端是用鍵盤輸入控制機器人移動，並加上距離感測器偵測距離，當偵測到距離小於 10，要讓機器人停止。

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


GPIO.setwarnings (False)
v=343
TRIG = 16
E = 18
LED_PIN = 12

HOST = '192.168.116.42'
PORT = 9324
```

1

```
GPIO.setmode (GPIO.BOARD)
GPIO.setup(LED_PIN, GPIO.OUT)
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup (E, GPIO.IN)
GPIO.output (TRIG, GPIO.LOW)

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print(1)
client.connect((HOST, PORT))
print(2)

def measure():
    GPIO.output (TRIG, GPIO.HIGH)
    time.sleep(0.00001)
```

2

```
while(1):
    d = measure()
    i, o, e = select.select([sys.stdin], [], [], 1)

    if d < 10:
        outdata = str("s")
        client.send(outdata)
        print(d)
        print(outdata)
    elif(i):
        command = sys.stdin.readline().strip()
        outdata = str(command)
        print(outdata)
        print(d)
        client.send(outdata)
    else:
        print("No command")
```

3

1: HOST 是 server 的 IP，port 要一樣。

2: 利用 socket 建立 TCP 連線。

3: 距離小於 10，則發送’s’，作為停止的輸入，此時 LED 會亮紅燈。

三、 心得

這是這個主題的最後一個實驗，結合了前面四次和電網導的課程內容，除了有趣也讓我覺得豐富而且有所收穫。雖然有趣，但 demo 過程並不容易，client 端嘗試和我連線時會發生 connection refused，有些組別的同學也是如此，不過他們在更換網路後就能執行成功，然而我多次嘗試 bun、我手機的網路、同學的手機網路，結果都失敗。

在瘋狂嘗試和切換 port 後，訊息終於成功從 client 端傳到我這個 server 端，之前會出現連線問題或是確認有連線但沒辦法傳訊息的原因還需要進一步查詢和討論。最後，很感謝協助捕 demo 的助教的細心和耐心，讓我能成功完成這次實驗。

附錄：

Server 端完整程式碼

```
#!/usr/bin/env python

# Copyright (c) 2011, Willow Garage, Inc.
# All rights reserved.
#
# Redistribution and use in source and binary forms, with or without
# modification, are permitted provided that the following conditions are met:
#
#    * Redistributions of source code must retain the above copyright
#      notice, this list of conditions and the following disclaimer.
#    * Redistributions in binary form must reproduce the above copyright
#      notice, this list of conditions and the following disclaimer in the
#      documentation and/or other materials provided with the distribution.
#    * Neither the name of the Willow Garage, Inc. nor the names of its
#      contributors may be used to endorse or promote products derived from
#       this software without specific prior written permission.
#
# THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
# AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
# IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
# ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
# LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
# CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
# SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
# INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
# CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
# ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
# POSSIBILITY OF SUCH DAMAGE.

import rospy
from geometry_msgs.msg import Twist
import sys, select, os
import socket
import time

# Establish a TCP socket
```

```
HOST = '172.20.10.7'
PORT = 9324
#sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
#sock.bind((HOST,PORT))
#sock.listen(5)
#conn,addr = sock.accept
#conn.settimeout(0.5)



try:
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
except socket.error, msg:
    sys.stderr.write("[ERROR] %s\n" % msg[1])
    sys.exit(1)

try:
    sock.bind((HOST,PORT))
    sock.listen(5)
except socket.error, msg:
    sys.stderr.write("[ERROR] %s\n" % msg[1])
    exit(1)

if os.name == 'nt':
  import msvcrt
else:
  import tty, termios

#the max linear velocity of turtlebot
BURGER_MAX_LIN_VEL = 0.13
BURGER_MAX_ANG_VEL = 2.84

WAFFLE_MAX_LIN_VEL = 0.20
WAFFLE_MAX_ANG_VEL = 1.82

#the increased value of velocity when you press the bottom
LIN_VEL_STEP_SIZE = 0.01
ANG_VEL_STEP_SIZE = 0.1

text = """
```

```
Control Your TurtleBot3!
---------------------------
Moving around:
       w
   a   s   d
       x

w/x : increase/decrease linear velocity (Burger : ~ 0.22, Waffle and Waffle Pi : ~ 0.26)
a/d : increase/decrease angular velocity (Burger : ~ 2.84, Waffle and Waffle Pi : ~ 1.82)

space key, s : force stop

CTRL-C to quit
"""
msg = ''
e = """
Communications Failed
"""



def vels(target_linear_vel, target_angular_vel):
    return "currently:\tlinear vel %s\t angular vel %s " % (target_linear_vel,target_angular_vel)

def makeSimpleProfile(output, input, slop):
    if input > output:
        output = min( input, output + slop )
    elif input < output:
        output = max( input, output - slop )
    else:
        output = input

    return output

def constrain(input, low, high):
    if input < low:
      input = low
    elif input > high:
      input = high
    else:
```

```
      input = input

    return input

def checkLinearLimitVelocity(vel):
    if turtlebot3_model == "burger":
      vel = constrain(vel, -BURGER_MAX_LIN_VEL, BURGER_MAX_LIN_VEL)
    elif turtlebot3_model == "waffle" or turtlebot3_model == "waffle_pi":
      vel = constrain(vel, -WAFFLE_MAX_LIN_VEL, WAFFLE_MAX_LIN_VEL)
    else:
      vel = constrain(vel, -BURGER_MAX_LIN_VEL, BURGER_MAX_LIN_VEL)

    return vel

def checkAngularLimitVelocity(vel):
    if turtlebot3_model == "burger":
      vel = constrain(vel, -BURGER_MAX_ANG_VEL, BURGER_MAX_ANG_VEL)
    elif turtlebot3_model == "waffle" or turtlebot3_model == "waffle_pi":
      vel = constrain(vel, -WAFFLE_MAX_ANG_VEL, WAFFLE_MAX_ANG_VEL)
    else:
      vel = constrain(vel, -BURGER_MAX_ANG_VEL, BURGER_MAX_ANG_VEL)

    return vel

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)
    conn,addr = sock.accept()
    print(addr)
    while(1):
        try:
            msg = conn.recv(1024)
            print(msg)
            # print(msg.decode())
            key = msg
        except socket.timeout:
            continue
        if msg == 'w' :
            target_linear_vel=checkLinearLimitVelocity(target_linear_vel + LIN_VEL_STEP_SIZE)
            status = status - 1
            print(vels(target_linear_vel,target_angular_vel))
        elif msg == 'x' :
            target_linear_vel=checkLinearLimitVelocity(target_linear_vel - LIN_VEL_STEP_SIZE)
            status = status - 1
            print(vels(target_linear_vel,target_angular_vel))
        elif msg == 'a' :
            target_angular_vel=checkAngularLimitVelocity(target_angular_vel + ANG_VEL_STEP_SIZE)
            status = status - 1
            print(vels(target_linear_vel,target_angular_vel))
        elif msg == 'd' :
            target_angular_vel=checkAngularLimitVelocity(target_angular_vel - ANG_VEL_STEP_SIZE)
            status = status -1
            print(vels(target_linear_vel,target_angular_vel))
        elif msg == ' ' or msg == 's':
            target_linear_vel   = 0.0
            control_linear_vel  = 0.0
            target_angular_vel  = 0.0
            control_angular_vel = 0.0
            status = status -1
            print(vels(target_linear_vel, target_angular_vel))
        else:
            if (msg == '\x03'):
                sock.close()
                break
```

```
            twist = Twist()

control_linear_vel=makeSimpleProfile(control_linear_vel,target_linear_vel,(LIN_VEL_STEP_SIZE/2.0))
            twist.linear.x = control_linear_vel; twist.linear.y = 0.0; twist.linear.z = 0.0


control_angular_vel=makeSimpleProfile(control_angular_vel,target_angular_vel,(ANG_VEL_STEP_SIZE/2.0))
            twist.angular.x = 0.0; twist.angular.y = 0.0; twist.angular.z = control_angular_vel

            pub.publish(twist)

    except:
        print(e)

    finally:
        twist = Twist()
        twist.linear.x = 0.0; twist.linear.y = 0.0; twist.linear.z = 0.0
        twist.angular.x = 0.0; twist.angular.y = 0.0; twist.angular.z = 0.0
        pub.publish(twist)

    if os.name != 'nt':
        termios.tcsetattr(sys.stdin, termios.TCSADRAIN, settings)
```

Reference:

https://reurl.cc/2E55NO

https://nordvpn.com/zh-tw/blog/tcp-udp-bijiao/