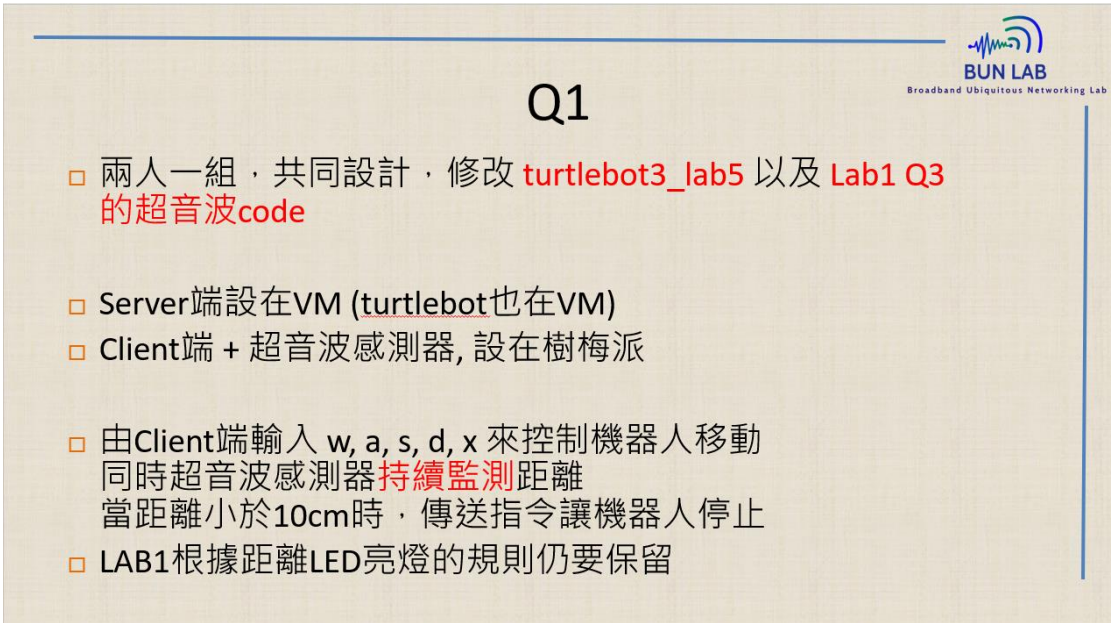
Communication Networks Lab

Topic IOT-Lab5 Turtlebot3



1. Establish socket on server (VM):

Using IPV4 and TCP protocol

```
# Establish a TCP socket
HOST = '172.20.10.7'
PORT = 8001

#IPV4 TCP protocol
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.bind((HOST,PORT))
sock.listen(5)    #at most 5 client can connect to server

#accept client connection and store the address
conn, addr = sock.accept()

#non-blocking
conn.settimeout(0.5)
```

2. Remove status variable in turtlebot3_lab5:

```
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 == '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
```

In lab4, the status variable is set to count how many status do we received from client. If the status is over 20, then we reset both linear velocity and angular velocity to 0. In this lab, we no longer need to use status variable. This variable is redundant in this program, and it keeps count down for no reason, so I remove the status to avoid confusing.

3. Deal with nothing received from client:

```
try:
    print(text)
    while(1):

        try:
            #receive message from client (at most 1024 bytes)
            msg = conn.recv(1024)
            print(addr)
            print(msg)
            key = msg

        except socket.timeout:
            pass

        if msg == 'w' :
            target_linear_vel = checkLinearLimitVelocity(target_linear_vel + LIN_VEL_STEP_SIZE)
            print(vels(target_linear_vel,target_angular_vel))
        elif msg == 'x' :
            target_linear_vel = checkLinearLimitVelocity(target_linear_vel - LIN_VEL_STEP_SIZE)
            print(vels(target_linear_vel,target_angular_vel))
        elif msg == 'a' :
            target_angular_vel = checkAngularLimitVelocity(target_angular_vel + ANG_VEL_STEP_SIZE)
            print(vels(target_linear_vel,target_angular_vel))
        elif msg == 'd' :
            target_angular_vel = checkAngularLimitVelocity(target_angular_vel - ANG_VEL_STEP_SIZE)
            print(vels(target_linear_vel,target_angular_vel))
        elif 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

        msg = ' '
```

After determine which message received from client, we set message to “ ”. By doing so, if no msg received from client, the linear velocity and angular velocity will remain the same, and it will not keep increasing or decreasing because of our msg didn’t clean up in next while loop.

4. Connect socket to server in client.py:

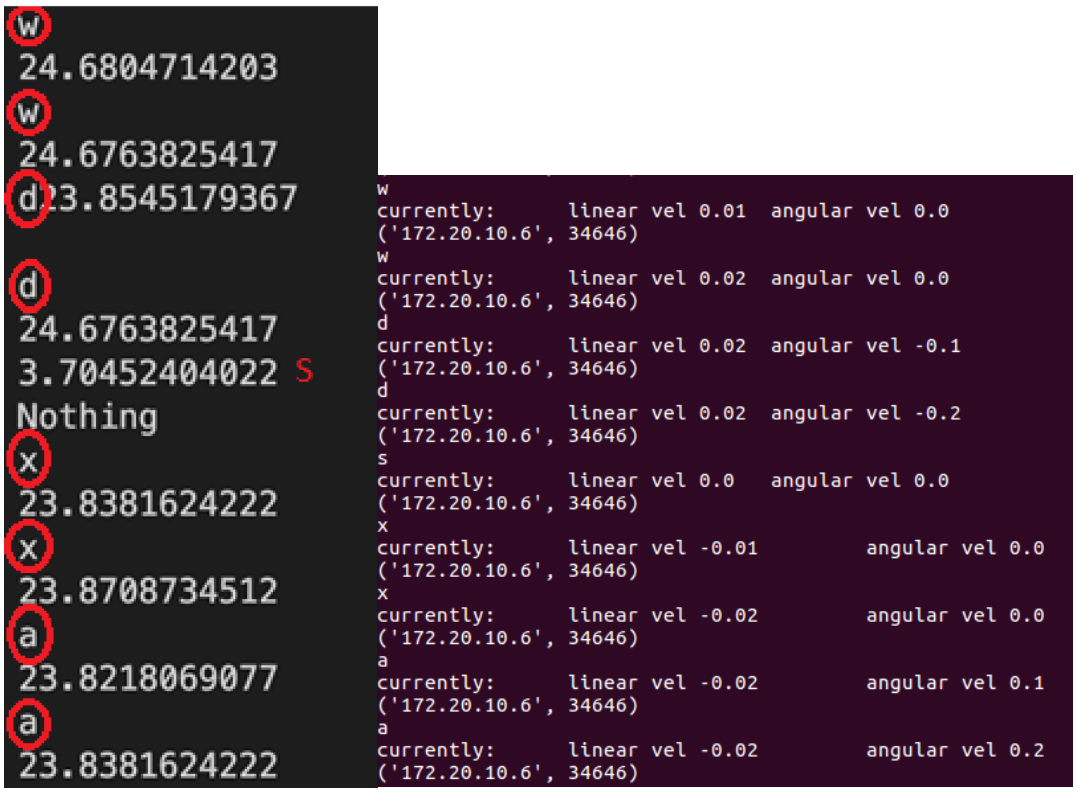
```python
#connect TCP socket to server
HOST = '172.20.10.7'
PORT = 8001

#IPV4, TCP
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect((HOST, PORT))
```

5. Client main code with timeout input:

```python
try:
    while(1):
        distance = measure()
        print(distance)

        i, o, e = select.select([sys.stdin], [], [], 3)
        if(i):
            a = sys.stdin.readline().strip()
            client.sendall(a)
        else:
            print('Nothing')


        if distance < 10:
            client.sendall('s')

        #keep the led off
        if distance > 20:
            pass

        #LED shine
        elif distance ⩽ 20 and distance ⩾ 10:
            shine()

        #LED turn on
        else:
            turn_on()
        time.sleep(3)

except KeyboardInterrupt:
    print('stop')

finally:
    GPIO.cleanup()

```

Using timeout input if user type some string on the standard input stream, and send it to server to control out robot. Also, use ultrasonic sensor to measure the distance. If distance is less than 5 cm, send stop signal to our robot, and reset both linear velocity and angular velocity to 0.

6. Result:



Send w twice to make robot move forward.
Send d twice to make robot turn right.

When distance is less than 5 cm, send s to reset the robot to 0 velocity.

Send x twice to make robot move downward.
Send a twice to make robot turn left.

7. Feedback:

這次 lab 比較好玩，透過超音波感測器去測量距離，當距離太小時，傳送停止的訊號給我們的模擬機器人，使他停下。這次還蠻順利的跟我的夥伴一起把 code 填好，一次就成功了!但是我覺得有一點小麻煩的是，感覺 turtlebot3 這個軟體對於終止軟體的部分沒有好好做好，可能有些子程序沒有 kill 掉，導致 address already in use，必須更換 port 再重來，不然就是要重新開機，這點算是小小可惜。除此之外，謝謝助教這五週的教學，獲益良多!!