Communication Networks Lab
Topic IOT-Lab4 Android Studio

Q1:

Code append in **turtlebot3_teleop_key**:

Define angular velocity is positive while turning left, and angular velocity is negative while turning right. Simply check the angular velocity of our burger model will not out of bound, and control the simulation robot run in the turtle bound.

```
elif key == 'a' :
    # add turn left action
    target_angular_vel = checkAngularLimitVelocity(target_angular_vel + ANG_VEL_STEP_SIZE)
    status = status + 1
    print(vels(target_linear_vel,target_angular_vel))

elif key == 'd' :
    # add turn right action
    target_angular_vel = checkAngularLimitVelocity(target_angular_vel - ANG_VEL_STEP_SIZE)
    status = status + 1
    print(vels(target_linear_vel,target_angular_vel))
```

Inside **checkAngularLimitVelocity()** function simply check the updated angular velocity of our burger robot will not out of bound.

```
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
```

In **constrain()** function simply remain the velocity if the velocity is out of bound.

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

    return input
```

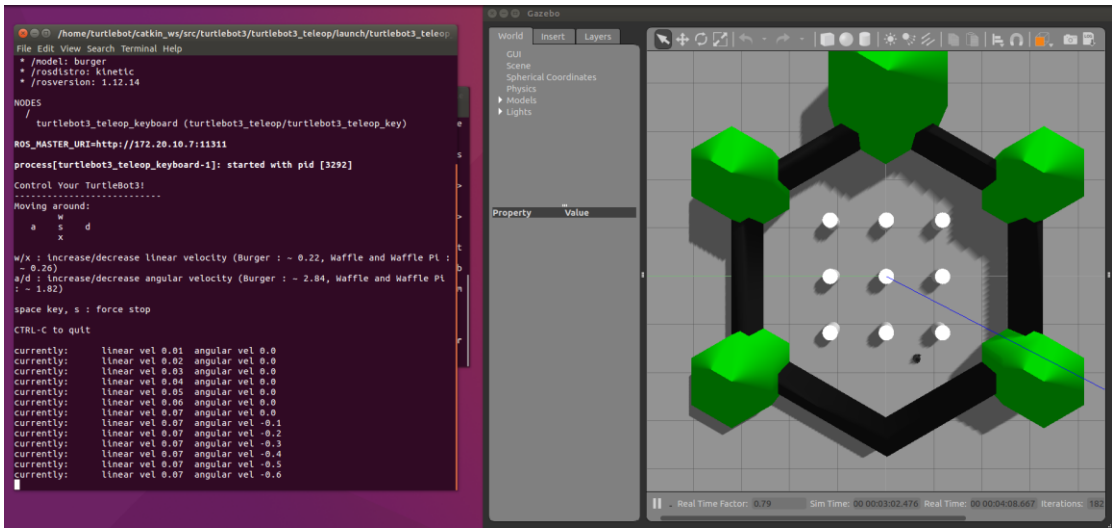
Since I see the Copyright (c) on the top of the code, therefore, I try to find whether the code is open sourced on their Github page, and I got the same code as below.

https://github.com/ROBOTIS-GIT/turtlebot3/blob/master/turtlebot3_teleop/nodes/turtlebot3_teleop_key

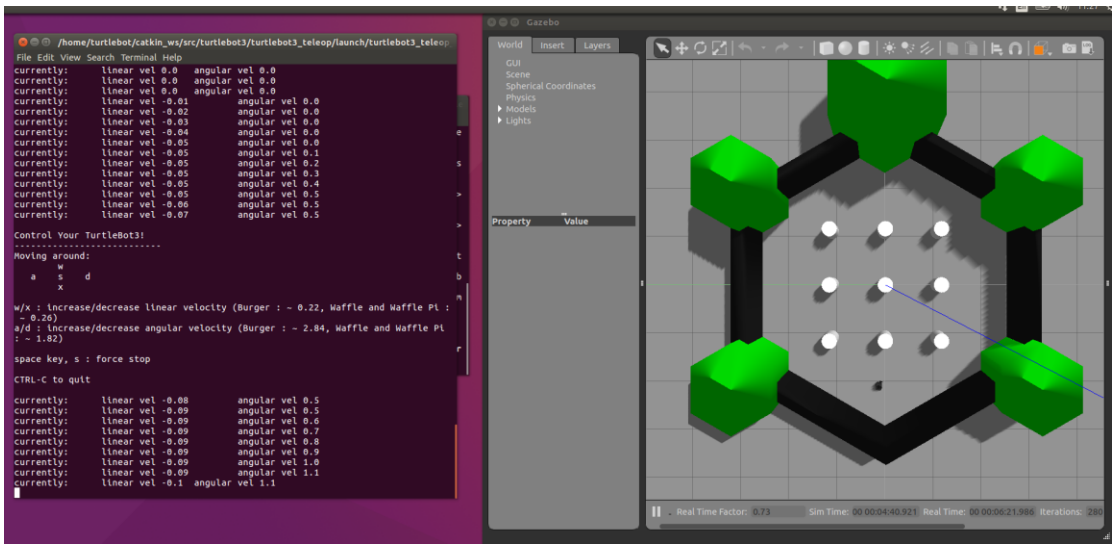
```
elif key == 'a' :
    target_angular_vel = checkAngularLimitVelocity(target_angular_vel + ANG_VEL_STEP_SIZE)
    status = status + 1
    print(vels(target_linear_vel,target_angular_vel))
elif key == 'd' :
    target_angular_vel = checkAngularLimitVelocity(target_angular_vel - ANG_VEL_STEP_SIZE)
    status = status + 1
    print(vels(target_linear_vel,target_angular_vel))
```

Result:

W+D move forward and turn right.



A+X move downward and turn left.



Discussion:

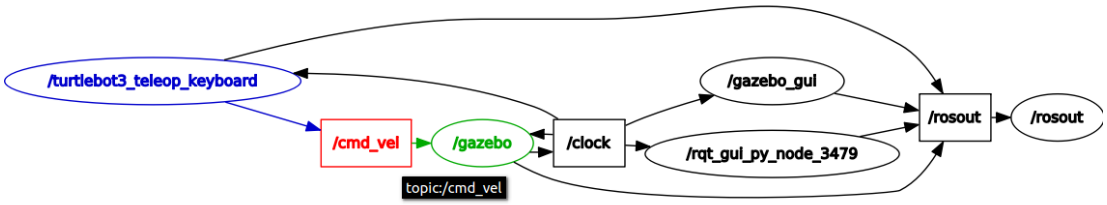
What actually happen in Q1?

Using `rosrun rqt_graph rqt_graph` to get more information in Q1.

Our program **turtlebot3_teleop_key** publish the topic :/cmd_vel to the subscriber Gazebo, which is the simulation panel of our experiment in Q1.

Turn on debug (hide common debug nodes)

Turn off debug (to see the hidden node)

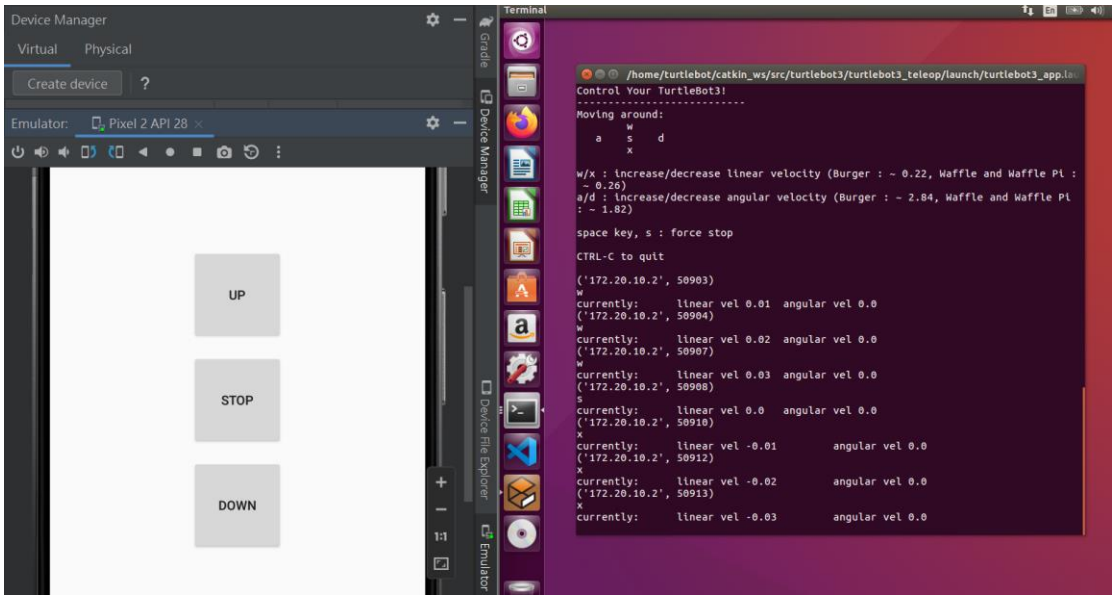


We can clearly see that node /turtlebot3_teleop_keyboard publish the topic:/cmd_vel to the subscriber /gazebo. We can observe that all of the nodes connect to the node rosout, which is the node to convey error messages. The communication between node and node is using socket this time.

In **turtlebot3_teleop_keyboard**, we can see that it publish the topic called 'cmd_vel':

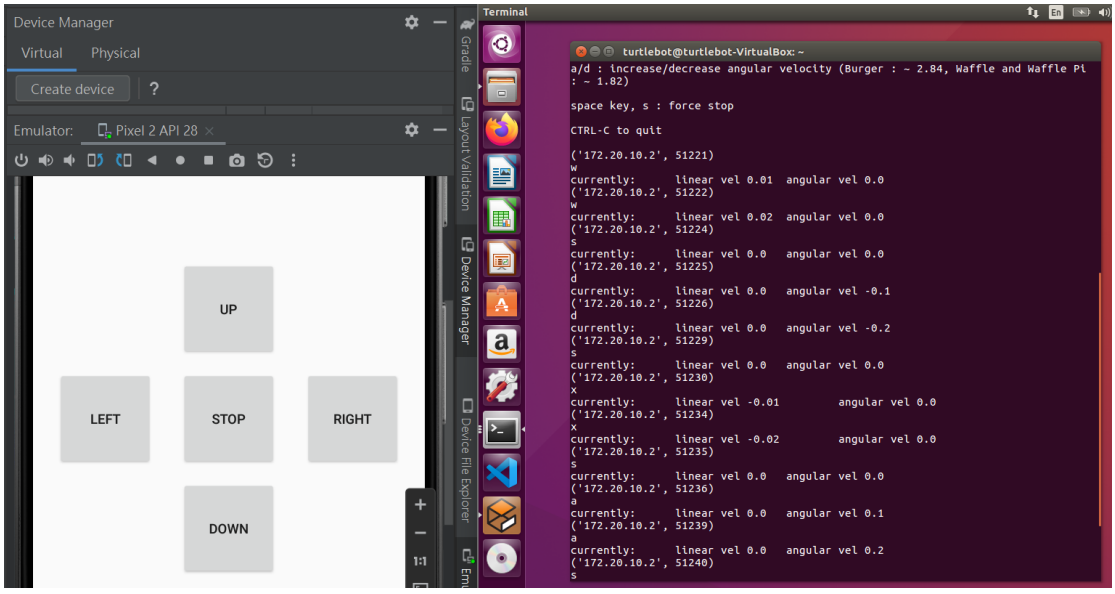
```
pub = rospy.Publisher('cmd_vel', Twist, queue_size=10)
```

Q2:


Device Manager
Virtual
Physical
Create device
Emulator: Pixel 2 API 28
UP
STOP
DOWN
Terminal
/home/turtlebot/catkin_ws/src/turtlebot3/turtlebot3_teleop/launch/turtlebot3_app.la
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
('172.20.10.2', 50903)
w
currently: linear vel 0.01 angular vel 0.0
('172.20.10.2', 50904)
w
currently: linear vel 0.02 angular vel 0.0
('172.20.10.2', 50907)
w
currently: linear vel 0.03 angular vel 0.0
('172.20.10.2', 50908)
s
currently: linear vel 0.0 angular vel 0.0
('172.20.10.2', 50910)
x
currently: linear vel -0.01 angular vel 0.0
('172.20.10.2', 50912)
x
currently: linear vel -0.02 angular vel 0.0
('172.20.10.2', 50913)
x
currently: linear vel -0.03 angular vel 0.0

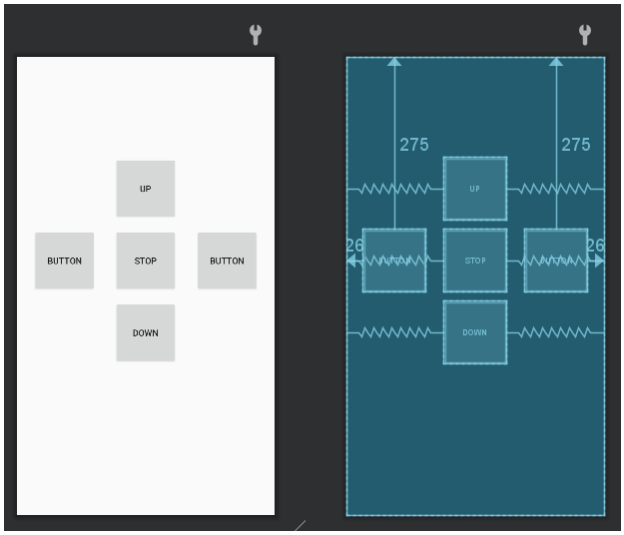
Q3:


Device Manager
Virtual
Physical
Create device
Emulator: Pixel 2 API 28
UP
LEFT
STOP
RIGHT
DOWN
Terminal
turtlebot@turtlebot-VirtualBox: ~
a/d : increase/decrease angular velocity (Burger : ~ 2.84, Waffle and Waffle Pi : ~ 1.82)
space key, s : force stop
CTRL-C to quit
('172.20.10.2', 51221)
w
currently: linear vel 0.01 angular vel 0.0
('172.20.10.2', 51222)
w
currently: linear vel 0.02 angular vel 0.0
('172.20.10.2', 51224)
s
currently: linear vel 0.0 angular vel 0.0
('172.20.10.2', 51225)
d
currently: linear vel 0.0 angular vel -0.1
('172.20.10.2', 51226)
d
currently: linear vel 0.0 angular vel -0.2
('172.20.10.2', 51229)
s
currently: linear vel 0.0 angular vel 0.0
('172.20.10.2', 51230)
x
currently: linear vel -0.01 angular vel 0.0
('172.20.10.2', 51234)
x
currently: linear vel -0.02 angular vel 0.0
('172.20.10.2', 51235)
s
currently: linear vel 0.0 angular vel 0.0
('172.20.10.2', 51236)
a
currently: linear vel 0.0 angular vel 0.1
('172.20.10.2', 51239)
a
currently: linear vel 0.0 angular vel 0.2
('172.20.10.2', 51240)
s

Button Layout on **activity_main.xml**

```
<Button
    android:id="@+id/right"
    android:layout_width="100dp"
    android:layout_height="100dp"
    android:layout_alignParentTop="true"
    android:layout_alignParentEnd="true"
    android:layout_alignParentRight="true"
    android:layout_marginTop="275dp"
    android:layout_marginEnd="26dp"
    android:layout_marginRight="27dp"
    android:text="Right" />

<Button
    android:id="@+id/left"
    android:layout_width="100dp"
    android:layout_height="100dp"
    android:layout_alignParentStart="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:layout_marginStart="26dp"
    android:layout_marginLeft="26dp"
    android:layout_marginTop="275dp"
    android:text="Left" />
```



MainActivity.java part to activate the button

```
button_left.setOnClickListener(new Button.OnClickListener(){

    @Override

    public void onClick(View v) {
        System.out.println("left");
        new SendData().execute("a");

    }

});

button_right.setOnClickListener(new Button.OnClickListener(){

    @Override

    public void onClick(View v) {
        System.out.println("right");
        new SendData().execute("d");

    }

});
```

Since there are multiple buttons need to use **setOnClickListener()** function, we need to add Override key word first.

Feedback:

之前就有使用過 Android Studio 來試著做一些 app，大家 Android Studio 的第一個範例一定都是 BMI 計算機，可惜在寫背後的運算式使用的是 JAVA 程式語言，因此當時的我早早就放棄了。到了這次實驗才大致上對 Android Studio 有更進一步的認識，透過拖拉把 app 版面做好，再 Main code 把背後的邏輯寫出來。在這次實驗中更在裡面加入了 Socket Programming，讓他可以跟其他軟體溝通，非常酷。這次實驗其實應該不用分組做，我覺得做的步驟反倒還比上一個 MQTT 的實驗步驟還要少，只是下載東西會下載很久而已，所以如果可以先在家先全部下載好，應該可以減少很多時間。

這次實驗介紹的是 ROS 是機器人的作業系統，剛好跟我前陣子在逛 github 上找到的 RIOT 有點類似: https://github.com/RIOT-OS/RIOT，是一個給 IOT 用的高效率低功耗的作業系統。一直都很期待自己也能夠參與這種大型 code 的開發，看到別人使用自己所開發的軟體，一定是非常興奮呢!