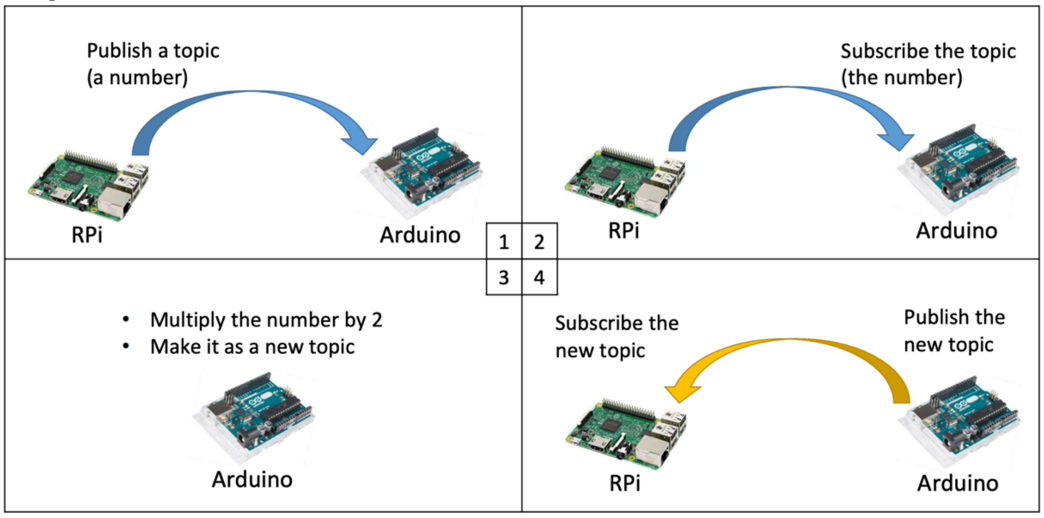
Checkpoint #1　Report

[EECN30169] Mobile Robot 2022

Student ID:310611008　　**Name:** 張祐誠　　**Date: Oct 12, 2022**

1. Purpose:

The purpose of this checkpoint requires us to:
1. Install the ROS platform on raspberry pi and set up the environment for robotics, including the connection with Arduino board.
2. Write project to publish an integer to Arduino from raspberry pi then receive the double of the number by the platform set up previously. As shown in the following figure.



(Figure credit to TA's slide)

2. Description of Design:

1. We first initialized a node on raspberry pi named 'publisher' and a topic '/numbers' where we published the user's input.

```
def publisher():
    pub = rospy.Publisher('numbers', Int32, queue_size=10)
    rospy.init_node("publisher")
```

2. Then we take user's input as a massage (noted the massage class was already declared in the previous step). Publish the massage to the '/numbers' topic.

```
while not rospy.is_shutdown():
    num_to_pub =int(input("user's input is "))
    pub.publish(num_to_pub)
```

3. Then the Arduino done the work. Similarly, Arduino first initialized a serial node, a subscriber to subscribe the '/numbers' topic, a publisher, a topic '/output' to publish the answer after multiplied the input.

```
ros::NodeHandle  nh;
long ansDouble;
std_msgs::Int32 output_msg;
ros::Publisher answer("/output", &output_msg);
```

```
ros::Subscriber<std_msgs::Int32> sub("/numbers", messageCb );
void setup()
{
  nh.initNode();
  nh.advertise(answer);
  nh.subscribe(sub);
}
```

4. The subscriber will subscribe the input and multiply the inputs by 2, then the massage will store the answer and publish to the topic ‘/output’.

```
void messageCb( const std_msgs::Int32& input_msg){
  long input = input_msg.data;
  ansDouble = input*2;
  output_msg.data = ansDouble;
  answer.publish( &output_msg);
}
```

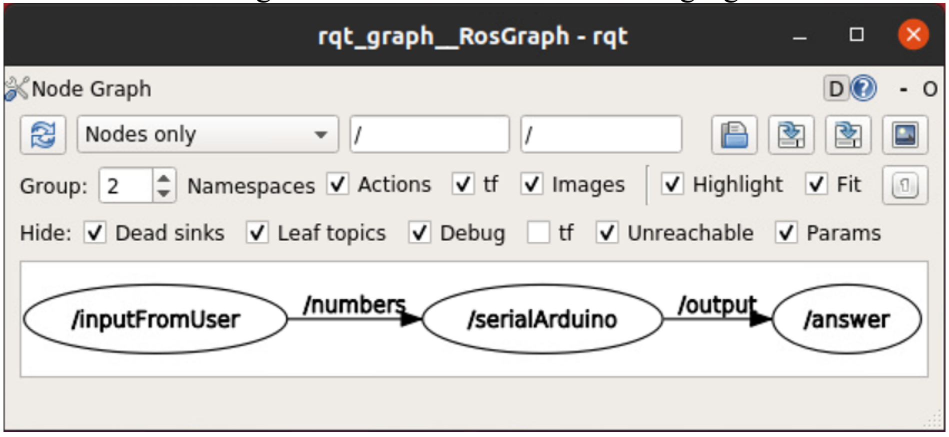
5. Lastly, a subscriber in raspberry pi will be initialized. Then subscribe the topic ‘/output’, which is published by Arduino from previous step.

```
def listener():
    rospy.init_node("listener")
    rospy.Subscriber("output", Int32, callback)
    rospy.spin()
```

6. The subscriber will subscribe the answer from ‘/output’ topic and print the answer in terminal.

```
def callback(data):
    #rospy.loginfo("")
    print("message from Arduino is "+str(data.data))
```

7. The overall configuration is shown in the following figure:


rqt_graph__RosGraph - rqt

Node Graph

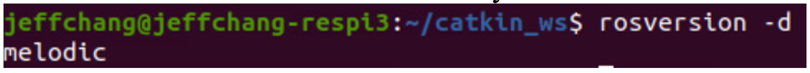
Nodes only | / | /

Group: 2 [✓ Namespaces] [✓ Actions] [✓ tf] [✓ Images] [✓ Highlight] [✓ Fit]

Hide: [✓ Dead sinks] [✓ Leaf topics] [✓ Debug] tf [✓ Unreachable] [✓ Params]

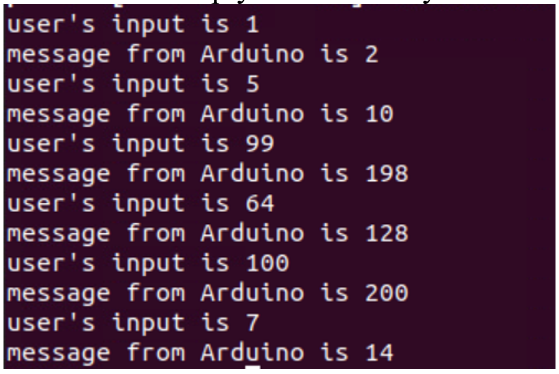
3. Result

A. Task 1: show the version of ROS by ‘$rosversion -d’ command:



```
jeffchang@jeffchang-respi3:~/catkin_ws$ rosversion -d
melodic
```

B. Task 2: multiply the answer by 2

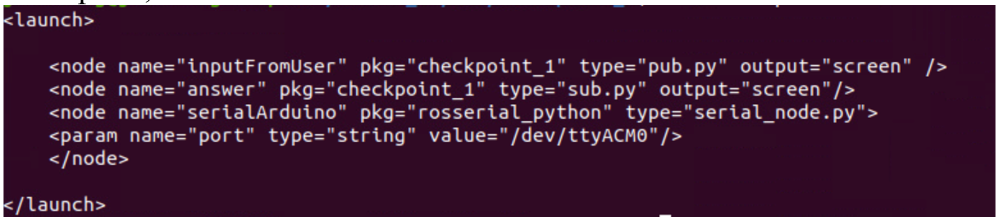


```
user's input is 1
message from Arduino is 2
user's input is 5
message from Arduino is 10
user's input is 99
message from Arduino is 198
user's input is 64
message from Arduino is 128
user's input is 100
message from Arduino is 200
user's input is 7
message from Arduino is 14
```

4. Discussion

1. We have tried to write publisher and subscriber in one node on raspberry pi, just like what has been done in Arduino node. But somehow it just can’t work perfectly. Therefore, the subscriber and publisher were written in different script.

2. Nevertheless, we have written launch file, which is not required in the checkpoint, to run all the node at the same time.

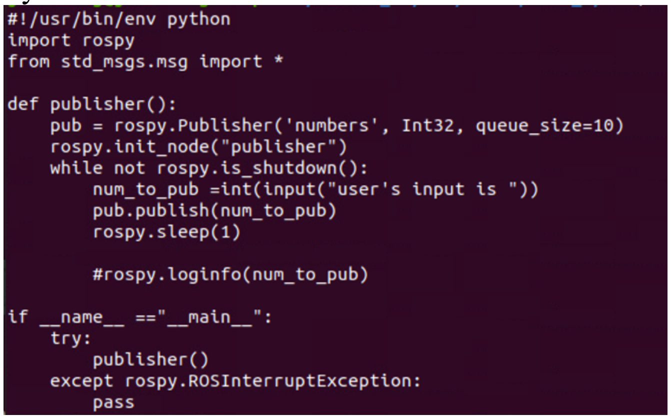


```
<launch>

    <node name="inputFromUser" pkg="checkpoint_1" type="pub.py" output="screen" />
    <node name="answer" pkg="checkpoint_1" type="sub.py" output="screen"/>
    <node name="serialArduino" pkg="rosserial_python" type="serial_node.py">
    <param name="port" type="string" value="/dev/ttyACM0"/>
    </node>

</launch>
```

3. It is well known that the ROS platform is to communicate different device or programming language. We have finished this project in both C++ and python and they all work perfectly. The only difference is that how to make the scripts executable. C++ file where compile by editing the Cmake file while python file using the $chmod +x command.

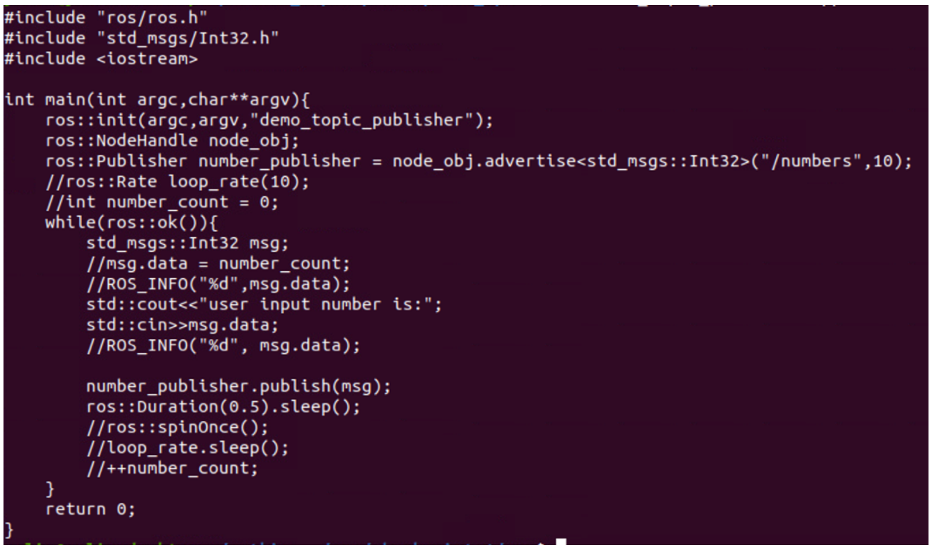
A. Publisher:

Python:



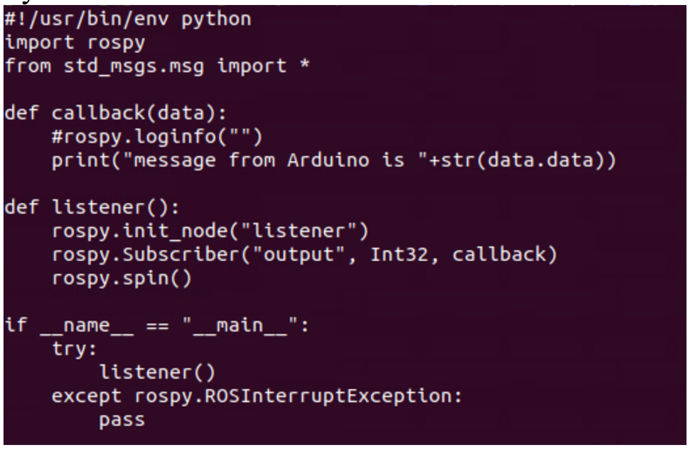
```
#!/usr/bin/env python
import rospy
from std_msgs.msg import *

def publisher():
    pub = rospy.Publisher('numbers', Int32, queue_size=10)
    rospy.init_node("publisher")
    while not rospy.is_shutdown():
        num_to_pub =int(input("user's input is "))
        pub.publish(num_to_pub)
        rospy.sleep(1)

        #rospy.loginfo(num_to_pub)

if __name__ =="__main__":
    try:
        publisher()
    except rospy.ROSInterruptException:
        pass
```

C++:



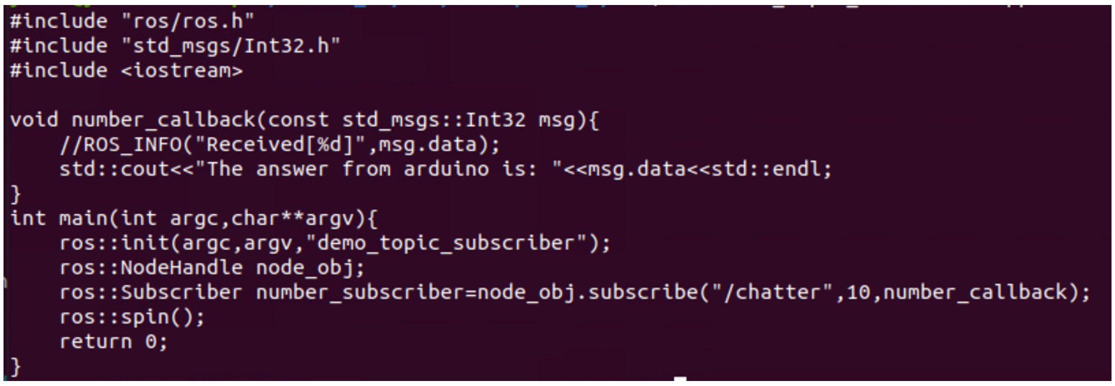
```
#include "ros/ros.h"
#include "std_msgs/Int32.h"
#include <iostream>

int main(int argc,char**argv){
    ros::init(argc,argv,"demo_topic_publisher");
    ros::NodeHandle node_obj;
    ros::Publisher number_publisher = node_obj.advertise<std_msgs::Int32>("/numbers",10);
    //ros::Rate loop_rate(10);
    //int number_count = 0;
    while(ros::ok()){
        std_msgs::Int32 msg;
        //msg.data = number_count;
        //ROS_INFO("%d",msg.data);
        std::cout<<"user input number is:";
        std::cin>>msg.data;
        //ROS_INFO("%d", msg.data);

        number_publisher.publish(msg);
        ros::Duration(0.5).sleep();
        //ros::spinOnce();
        //loop_rate.sleep();
        //++number_count;
    }
    return 0;
}
```

B. Subscriber:
Python:



```
#!/usr/bin/env python
import rospy
from std_msgs.msg import *

def callback(data):
    #rospy.loginfo("")
    print("message from Arduino is "+str(data.data))

def listener():
    rospy.init_node("listener")
    rospy.Subscriber("output", Int32, callback)
    rospy.spin()

if __name__ == "__main__":
    try:
        listener()
    except rospy.ROSInterruptException:
        pass
```

C++:



```
#include "ros/ros.h"
#include "std_msgs/Int32.h"
#include <iostream>

void number_callback(const std_msgs::Int32 msg){
    //ROS_INFO("Received[%d]",msg.data);
    std::cout<<"The answer from arduino is: "<<msg.data<<std::endl;
}
int main(int argc,char**argv){
    ros::init(argc,argv,"demo_topic_subscriber");
    ros::NodeHandle node_obj;
    ros::Subscriber number_subscriber=node_obj.subscribe("/chatter",10,number_callback);
    ros::spin();
    return 0;
}
```