4.Publisher

4.1 Publisher

Publishers, as the name suggests, serve to publish messages. The message here can be sensor information transmitted from the lower computer to the upper computer, which is then packaged and sent to subscribers who have subscribed to the topic through the upper computer; It can also be the data from the upper computer that has been processed, packaged, and sent to subscribers who subscribe to the topic.

4.2. Create a workspace and topic feature pack

4.2.1 Creating a workspace

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/src
catkin_init_workspace
```

4.2.2 Compilation workspace

```
cd ~/catkin_ws/
catkin_make
```

4.2.3. Updating Environmental Variables

```
source devel/setup.bash
```

4.2.4. Checking Environmental Variables

```
echo $ROS_PACKAGE_PATH
```

4.2.5. Create Function Package

```
cd ~/catkin_ws/src
catkin_create_pkg learning_topic std_msgs rospy roscpp geometry_msgs turtlesim
```

4.2.6 Compilation Function Package

```
cd ~/catkin_ws
catkin_make
source ~/catkin_ws/devel/setup.bash
```

4.3. Creating a Publisher

4.3.1 Creation steps

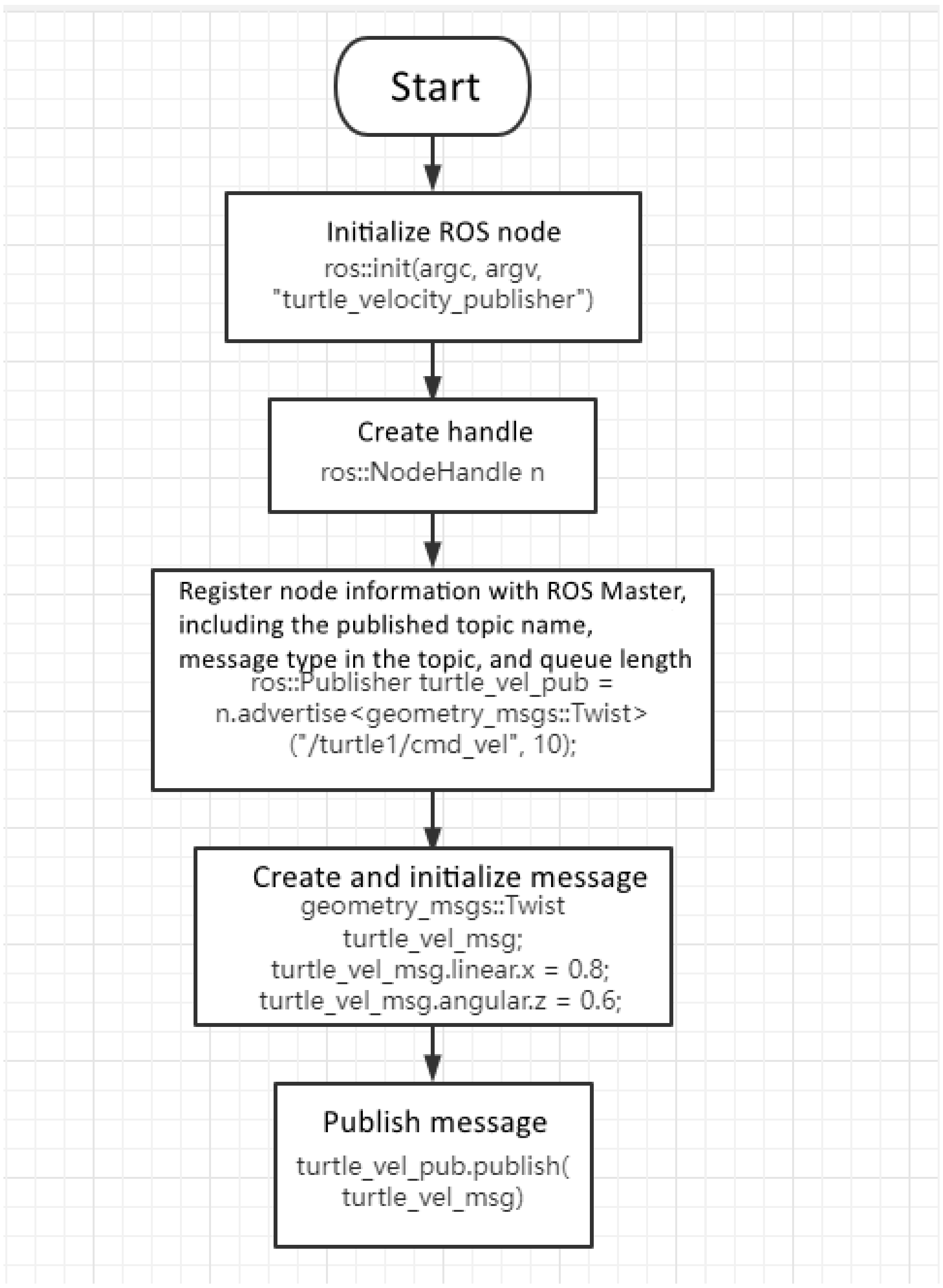
1) Initialize ROS node
2) Create handleRegister
3) node information with ROS Master, including the published topic name, message types in the topic, and queue length
4) Create and initialize message data
5) Cycle sending messages at a certain frequency

4.3.2. C++Language Implementation

1) In the src folder of the feature pack, create a c++file (with the suffix. cpp) and name it turtle_ velocity_ publisher.cpp
2) Copy and paste the program code below into the title_ velocity_ In the publisher.cpp file

```
#include <ros/ros.h>
#include <geometry_msgs/Twist.h>
int main(int argc, char **argv){

    ros::init(argc, argv, "turtle_velocity_publisher");

    ros::NodeHandle n;


    ros::Publisher turtle_vel_pub = n.advertise<geometry_msgs::Twist>
("/turtle1/cmd_vel", 10);

    ros::Rate loop_rate(10);

    while (ros::ok()){

        geometry_msgs::Twist turtle_vel_msg;
        turtle_vel_msg.linear.x = 0.8;
        turtle_vel_msg.angular.z = 0.6;

        turtle_vel_pub.publish(turtle_vel_msg);/


        ROS_INFO("Publsh turtle velocity command[%0.2f m/s, %0.2f rad/s]",
turtle_vel_msg.linear.x, turtle_vel_msg.angular.z);

        loop_rate.sleep();/
    }
    return 0;
}
```

3. Program flowchart, corresponding to 1.3.1 content for viewing


Start
Initialize ROS node
ros::init(argc, argv, "turtle_velocity_publisher")
Create handle
ros::NodeHandle n
Register node information with ROS Master, including the published topic name, message type in the topic, and queue length
ros::Publisher turtle_vel_pub = n.advertise<geometry_msgs::Twist>("/turtle1/cmd_vel", 10);
Create and initialize message
geometry_msgs::Twist turtle_vel_msg;
turtle_vel_msg.linear.x = 0.8;
turtle_vel_msg.angular.z = 0.6;
Publish message
turtle_vel_pub.publish(turtle_vel_msg)

4. Configure in CMakelist.txt, under the build area, add the following content

```
add_executable(turtle_velocity_publisher src/turtle_velocity_publisher.cpp)
target_link_libraries(turtle_velocity_publisher ${catkin_LIBRARIES})
```

5. Compiling code under workspace directory

```
cd ~/catkin_ws
catkin_make
source devel/setup.bash
```

6. run a program

- Run score

```
roscore
```

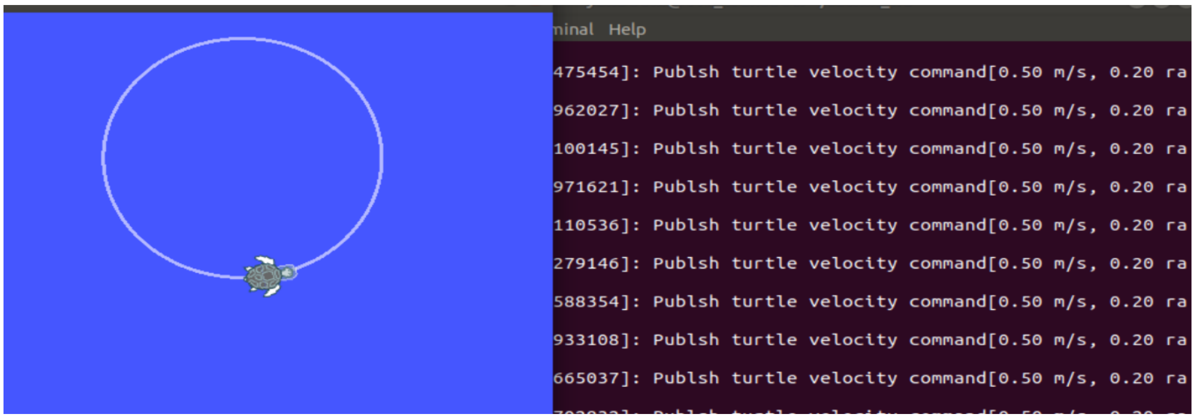
- Running the Little Turtle Node

```
rosrun turtlesim turtlesim_node
```

- Run the publisher and continuously send speed to Little Turtle

```
rosrun learning_topic turtle_velocity_publisher
```

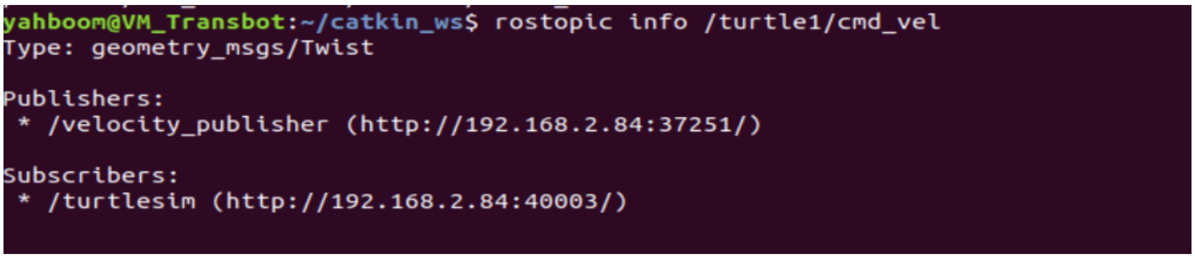
7. Screenshot of operation effect



```
475454]: Publsh turtle velocity command[0.50 m/s, 0.20 ra
962027]: Publsh turtle velocity command[0.50 m/s, 0.20 ra
100145]: Publsh turtle velocity command[0.50 m/s, 0.20 ra
971621]: Publsh turtle velocity command[0.50 m/s, 0.20 ra
110536]: Publsh turtle velocity command[0.50 m/s, 0.20 ra
279146]: Publsh turtle velocity command[0.50 m/s, 0.20 ra
588354]: Publsh turtle velocity command[0.50 m/s, 0.20 ra
933108]: Publsh turtle velocity command[0.50 m/s, 0.20 ra
665037]: Publsh turtle velocity command[0.50 m/s, 0.20 ra
```

8. Program operation instructions

- Entering the rostopic list on the terminal to view the topic list will reveal/tourle1/cmd_ The topic of vel
- We use rostopic info/tourle1/cmd_ Looking at vel, you will find that



```
yahboom@VM_Transbot:~/catkin_ws$ rostopic info /turtle1/cmd_vel
Type: geometry_msgs/Twist

Publishers:
 * /velocity_publisher (http://192.168.2.84:37251/)

Subscribers:
 * /turtlesim (http://192.168.2.84:40003/)
```

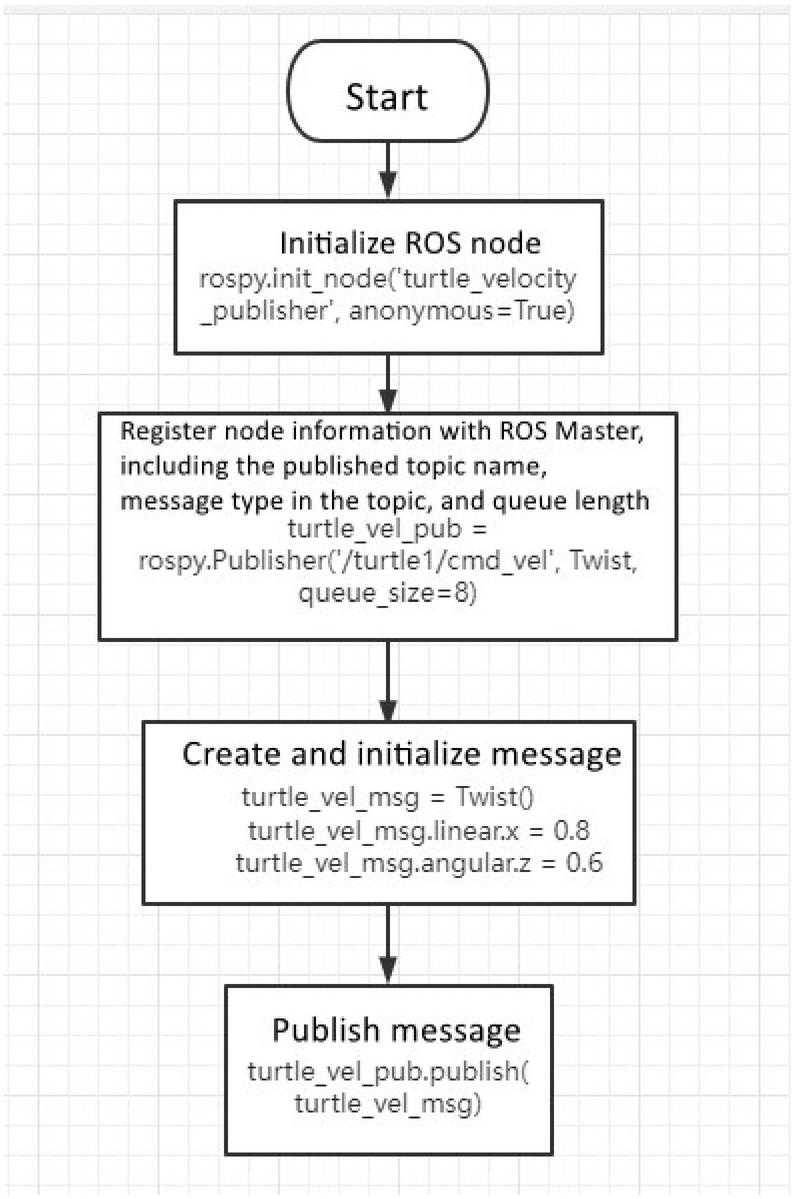
This indicates that Little Turtle is a subscription/tourle1/cmd_ The speed topic is vel, so the publisher keeps sending speed data. After receiving it, the little turtle starts to move according to the speed

4.3.3. Python Language Implementation

1. In the feature pack directory, create a new folder called scripts, and then create a new Python file (with the suffix. py) under the scripts folder, named turtle_ velocity_ publisher.py
2. Copy and paste the program code below into the title_ velocity_ In the publisher.py file

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import rospy
from geometry_msgs.msg import Twist

def turtle_velocity_publisher():

    rospy.init_node('turtle_velocity_publisher', anonymous=True) # ROS node
initialization


    turtle_vel_pub = rospy.Publisher('/turtle1/cmd_vel', Twist, queue_size=8)
    rate = rospy.Rate(10) #Set the frequency of the loop

    while not rospy.is_shutdown():
        # Initializes a message of type geometry_msgs::Twist
        turtle_vel_msg = Twist()
        turtle_vel_msg.linear.x = 0.8
        turtle_vel_msg.angular.z = 0.6

        # Publish a message
        turtle_vel_pub.publish(turtle_vel_msg)
        rospy.loginfo("linear is :%0.2f m/s, angular is :%0.2f rad/s",
                turtle_vel_msg.linear.x, turtle_vel_msg.angular.z)


        rate.sleep()# Delay according to the cycle frequency

if __name__ == '__main__':
    try:
        turtle_velocity_publisher()
    except rospy.ROSInterruptException:
        pass
```

3. Process Flow Chart


Start
Initialize ROS node
rospy.init_node('turtle_velocity_publisher', anonymous=True)
Register node information with ROS Master, including the published topic name, message type in the topic, and queue length
turtle_vel_pub = rospy.Publisher('/turtle1/cmd_vel', Twist, queue_size=8)
Create and initialize message
turtle_vel_msg = Twist()
turtle_vel_msg.linear.x = 0.8
turtle_vel_msg.angular.z = 0.6
Publish message
turtle_vel_pub.publish(turtle_vel_msg)

4. run a program

- Run score

```
roscore
```

- Running the Little Turtle Node

```
rosrun turtlesim turtlesim_node
```

- Run the publisher and continuously send speed to Little Turtle

```
rosrun learning_topic turtle_velocity_publisher
```

Note: Before running, it is necessary to give the title_ velocity_ Publisher.py adds executable permissions in the title_ velocity_ Open the terminal in the publisher.py folder,

```
sudo chmod a+x turtle_velocity_publisher.py
```

All Python requires adding execute file permissions, otherwise an error will be reported!5) Refer to 1.3.2 for operational effects and program instructions.