Midterm\_AIRobot\_sensor1

Li Kailang FDQAOR

---------------------------------------------------------------------------

**This is what I do in the exam:**

• Install the sensors package [unzip midterm\_sensor\_1 to your work space and install it]

Move it one by one

• Investigate in which topic it sends the messages, and what are the types used (1 pt)

chmod +x ~/ros2\_ws/src/midterm\_sensor\_1/midterm\_sensor\_1/sensor.py

cd ~/ros2\_ws

colcon build

source install/setup.bash

ros2 run midterm\_sensor\_1 sensor\_node

•Make a package for the solution with Your neptune code as package name (1 pt)

cd ~/ros2\_ws/src

ros2 pkg create --build-type ament\_python FDQAOR

• Make a node that can subscribe the the sensors data (3 pt)

Create sub\_sensor.py and copy code from sensor, Then change the code as a listener.

Move the sensor.py to FDQAOR file and change the xml and setup.py.

Then there are many bugs and I can’t solve it by myself.

---------------------------------------------------------------------------

**Here are some feedback after exams: (with chatgpt)**

1. The file name can’t be captials.

So I move the subscriber to sensor file and rewrite everything.

1. Every time I open a new window, must run those code:

colcon build --packages-select midterm\_sensor\_1 --clean

colcon build

source install/setup.bash

1. After that run the sensor, listener and TurtleSim, it can run.

---------------------------------------------------------------------------

**Below are details for this project:**

1. Unzip and install the sensor package

The problem asks us to unpack the sensor package (let's say it's called midterm\_sensor\_1). You'll need to unzip it into your workspace and install it.

bash

Copy code

# Unzip the sensor package to the workspace

unzip midterm\_sensor\_1.zip -d ~/ros2\_ws/src/

Make sure that the sensor.py file has execution permissions:

bash

Copy code

chmod +x ~/ros2\_ws/src/midterm\_sensor\_1/midterm\_sensor\_1/sensor.py

# Go to the workspace and build

cd ~/ros2\_ws

colcon build

source install/setup.bash

At this point, the sensor package has been built and installed, and you can move on to the next task.

2. Check the sensor message

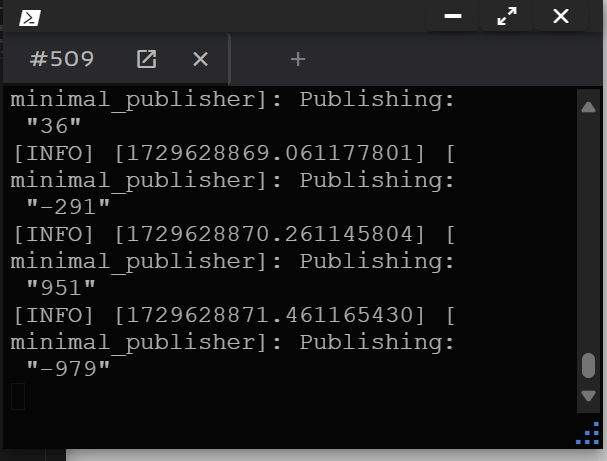
You need to confirm the type of message and topic posted by the sensor node. It is assumed that the sensor package has been started correctly.

Start the sensor node:

bash

Copy code

ros2 run midterm\_sensor\_1 sensor\_node



Use the ros2 topic command to view the threads posted by the sensor:

bash

Copy code

ros2 topic list

You should be able to see a thread name like /sensor1\_signal. Confirm that this topic is the publication of sensor data.

graphical user interface

Medium confidence descriptions are automatically generated

使用 ros2 topic echo 查看消息内容和类型：

bash

Copy code

ros2 topic echo /sensor1\_signal

This will show the format of the data published by the sensor, assuming the data is of type std\_msgs/msg/int32.

text

The description is automatically generated

3. Create a ROS 2 package

Depending on the question, you'll need to create a new package that will process the sensor's data and control the robot. Name the package according to your Neptune code, and the package name is FDQAOR.

bash

Copy code

c

4. Modify the setup.py

Go to the new package directory and modify the setup.py so that it can install and run your node.

bash

Copy code

cd ~/ros2\_ws/src/FDQAOR

Edit setup.py:

python

Copy code

from setuptools import setup

package\_name = 'FDQAOR'

setup(

name=package\_name,

version='0.0.0',

packages=[package\_name],

data\_files=[

('share/ament\_index/resource\_index/packages',

['resource/' + package\_name]),

('share/' + package\_name, ['package.xml']),

],

install\_requires=['setuptools'],

zip\_safe=True,

maintainer='Your Name',

maintainer\_email='your.email@domain.com',

description='Robot controller for TurtleSim based on sensor input',

license='MIT',

tests\_require=['pytest'],

entry\_points={

'console\_scripts': [

'robot\_controller = FDQAOR.robot\_controller:main',

],

},

)

5. Create a control node

To create a Python script file:

bash

Copy code

mkdir ~/ros2\_ws/src/FDQAOR/FDQAOR

touch ~/ros2\_ws/src/FDQAOR/FDQAOR/\_\_init\_\_.py

touch ~/ros2\_ws/src/FDQAOR/FDQAOR/robot\_controller.py

Edit robot\_controller.py files, subscribe to sensor topics and control TurtleSim's movement based on data.

python

Copy code

import rclpy

from rclpy.node import Node

from geometry\_msgs.msg import Twist

from std\_msgs.msg import Int32

class RobotController(Node):

def \_\_init\_\_(self):

super().\_\_init\_\_('robot\_controller')

self.subscription = self.create\_subscription(

Int32,

'/sensor1\_signal', # the topic name of the sensor

self.listener\_callback,

10)

self.publisher\_ = self.create\_publisher(Twist, '/turtle1/cmd\_vel', 10)

def listener\_callback(self, msg):

twist = Twist()

# Deal with negative numbers, turn right 120 degrees

if msg.data < 0:

self.get\_logger().info('Negative value received: Turning right 120 degrees')

twist.angular.z = -2.094 # 右转120度

# Deal with positive odd numbers, advance 2 units

elif msg.data > 0 and msg.data % 2 == 1:

self.get\_logger().info('Positive odd value received: Moving forward 2 units')

twist.linear.x = 4.0 # 前进2个单位

# Deal with positive and even numbers, advance 1 unit

elif msg.data > 0 and msg.data % 2 == 0:

self.get\_logger().info('Positive even value received: Moving forward 1 unit')

twist.linear.x = 2.0 # 前进1个单位

# Publish exercise commands

self.publisher\_.publish(twist)

def main(args=None):

rclpy.init(args=args)

robot\_controller = RobotController()

rclpy.spin(robot\_controller)

robot\_controller.destroy\_node()

rclpy.shutdown()

if \_\_name\_\_ == '\_\_main\_\_':

main()

6. 修改 package.xml

Make sure the package.xml file contains dependencies:

xml

Copy code

<?xml version="1.0"?>

<package format="2">

<name>fdkwware</name>

<version>0.0.0</version>

<description>Robot controller for TurtleSim based on sensor input</description>

<maintainer email="your.email@domain.com">Your Name</maintainer>

<license>MIT</license>

<buildtool\_depend>ament\_cmake</buildtool\_depend>

<exec\_depend>rclpy</exec\_depend>

<exec\_depend>std\_msgs</exec\_depend>

<exec\_depend>geometry\_msgs</exec\_depend>

<exec\_depend>turtlesim</exec\_depend>

</package>

7. Build the package

Once you've done all the code writing and configuration, go back to the workspace root and build the package:

bash

Copy code

cd ~/ros2\_ws

colcon build

source install/setup.bash

8. Start TurtleSim and your control node

Start the TurtleSim simulation:

bash

Copy code

ros2 run turtlesim turtlesim\_node

Start the sensor node (assuming the package name is midterm\_sensor\_1):

bash

Copy code

ros2 run midterm\_sensor\_1 sensor\_node

text

The description is automatically generated

Start your robot control node:

bash

Copy code

ros2 run FDQAOR robot\_controller

text

The description is automatically generated

9. Testing and debugging

Now, the baby turtle in TurtleSim should move accordingly based on the data released by the sensor. If the sensor data is negative, the turtle will turn right; If it is a positive odd number, it will move forward by 2 units; If it is a positive even number, it will move forward by 1 unit.

The image contains the graphical user interface

The description is automatically generated