



THE HONG KONG  
POLYTECHNIC UNIVERSITY  
香港理工大學



Department of  
Aeronautical and Aviation Engineering  
航空及民航工程學系

Slide 1 of 29

# **Electronics & Information Technologies for Unmanned Aerial Systems**

## **Week 3: Project Session 1**

**Dr. Hailong Huang**

Assistant Professor

Department of Aeronautical and Aviation Engineering

The Hong Kong Polytechnic University

1. ROS2 Project Creation
2. An Example of using Topics Communication
3. Project Assignment 1

- **Connect your computer with the monitor, keyboard, and mouse**
- **Turn on your computer**  
**password: 1234**
- **Download project resources**

```
# download resources from Github  
git clone https://github.com/AIMSPolyU/AAE4202_project.git
```

1. ROS2 Project Creation
2. An Example of using Topics Communication
3. Project Assignment 1

# 1. ROS2 Project Creation

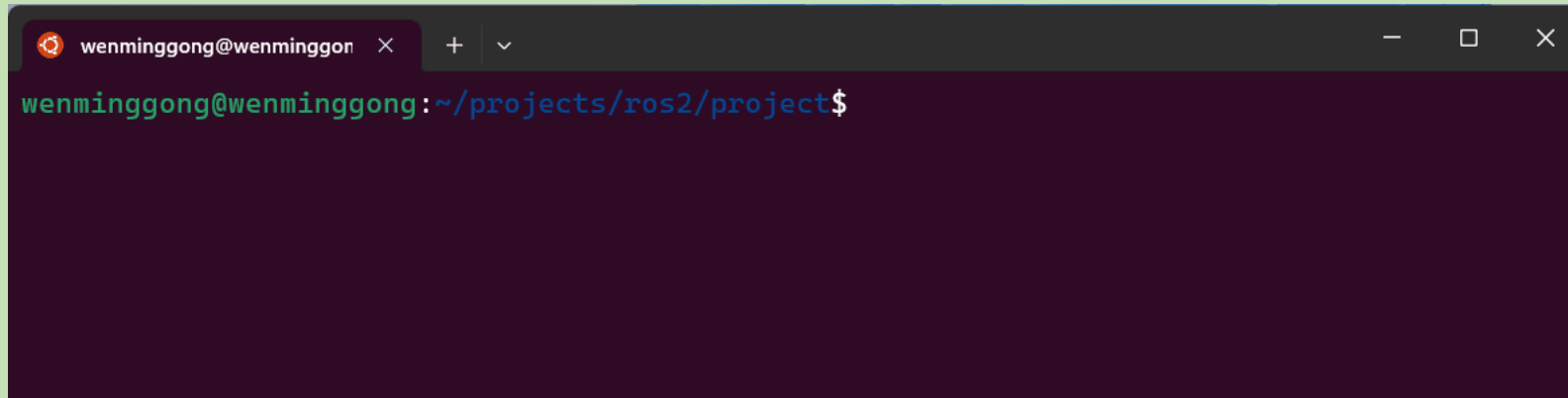
## Create a new ROS2 project from scratch

*# 1. create a workspace*

*mkdir -p workspace\_name/src*

*# 2. enter workspace\_name/src*

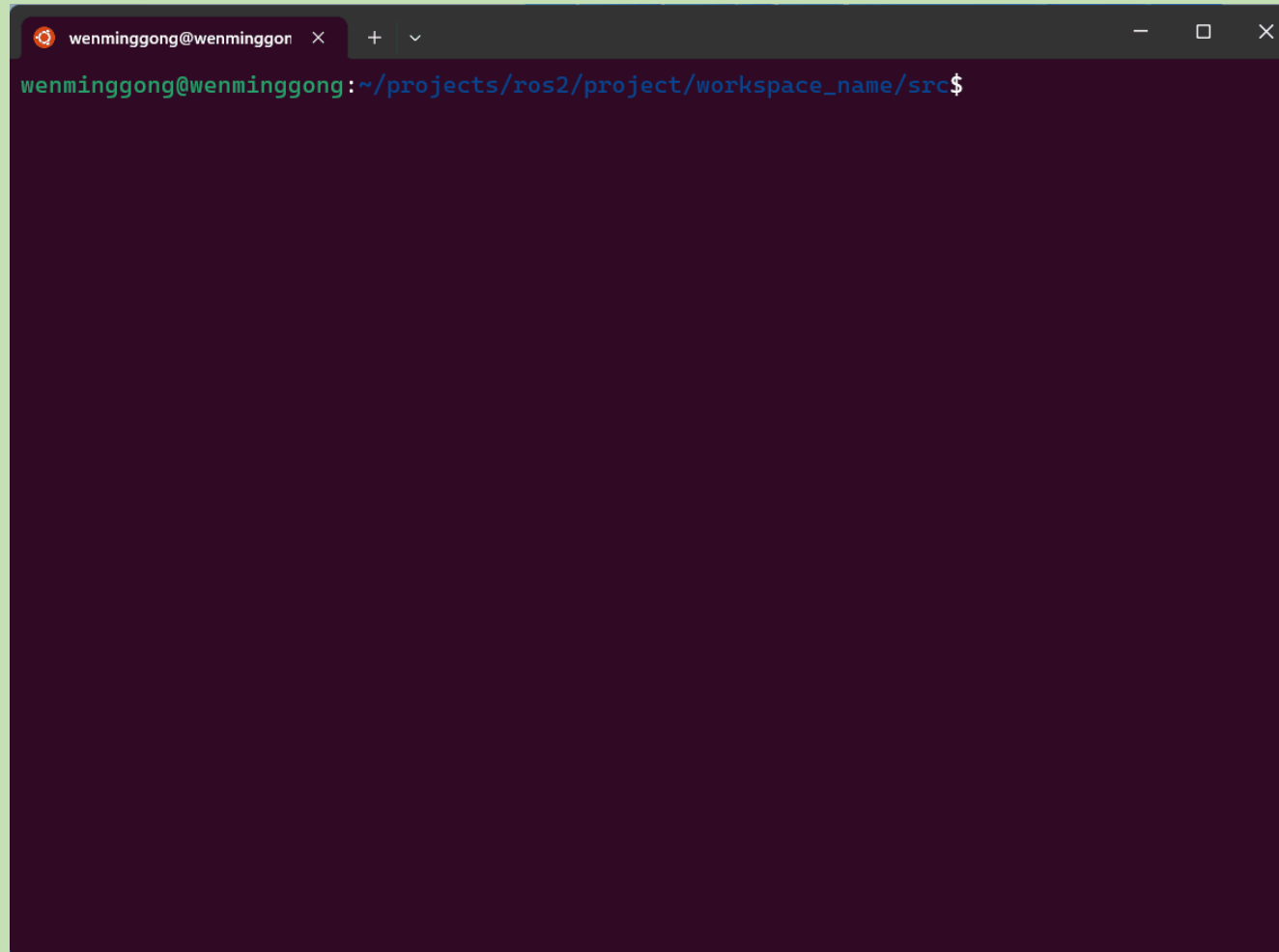
*cd workspace\_name/src*

A terminal window with a dark background and light text. The window title bar shows 'wenminggong@wenminggon' and standard window controls. The terminal text shows the user 'wenminggong' at the prompt 'wenminggong@wenminggong' with the current directory path '~ /projects/ros2/project\$'.

```
wenminggong@wenminggong:~/projects/ros2/project$
```

# 1. ROS2 Project Creation

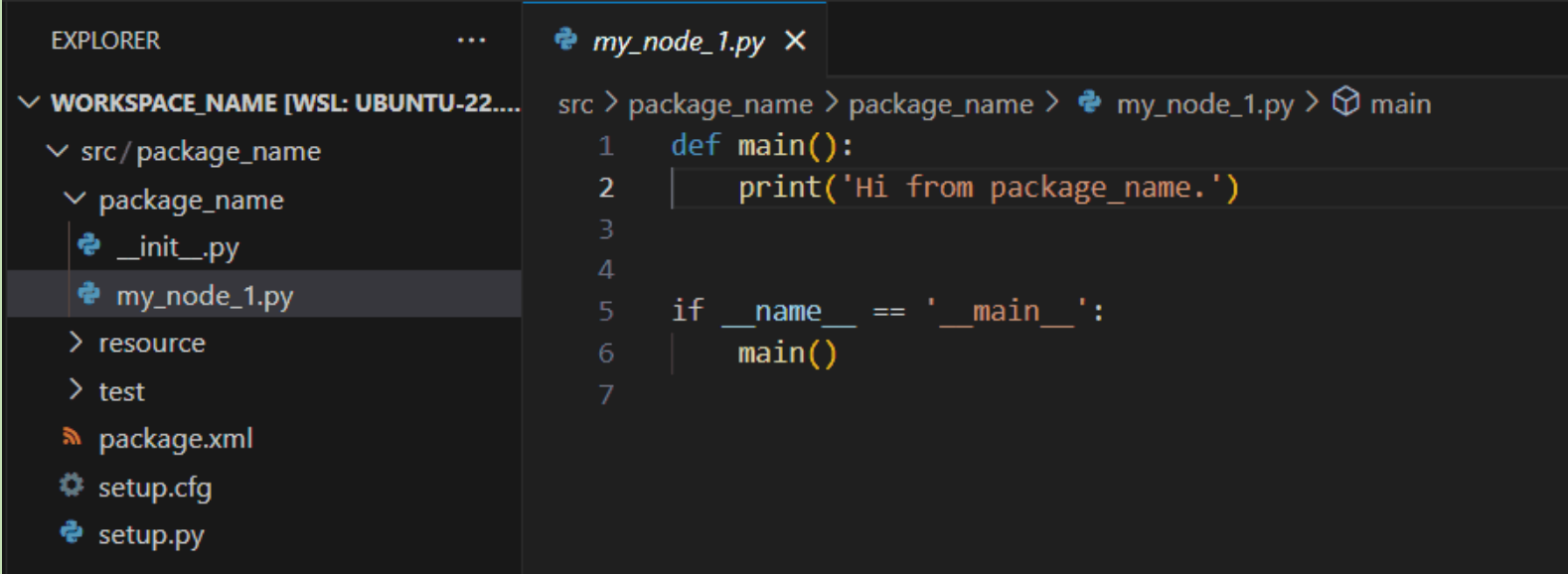
*# 3. create a new python package with a node*  
*ros2 pkg create package\_name --build-type ament\_python --dependencies rclpy*  
*--node-name my\_node\_1*

A terminal window with a dark background and light text. The window title bar shows 'wenminggong@wenminggon' and standard window controls. The prompt is 'wenminggong@wenminggong: ~/projects/ros2/project/workspace\_name/src\$'.

```
wenminggong@wenminggong: ~/projects/ros2/project/workspace_name/src$
```

# 1. ROS2 Project Creation

*# 4. open workspace\_name/src/package\_name/package\_name/my\_node\_1.py and write your code*



The screenshot shows the Visual Studio Code interface. On the left, the Explorer panel displays the project structure: `WORKSPACE_NAME [WSL: UBUNTU-22...]` is expanded, showing `src/package_name`, which is further expanded to show `package_name`. Inside `package_name`, the files `__init__.py` and `my_node_1.py` are listed, with `my_node_1.py` selected. Below this, other files like `resource`, `test`, `package.xml`, `setup.cfg`, and `setup.py` are visible. On the right, the editor shows the content of `my_node_1.py`. The file path at the top is `src > package_name > package_name > my_node_1.py > main`. The code in the editor is as follows:

```
1  def main():
2      print('Hi from package_name.')
3
4
5  if __name__ == '__main__':
6      main()
7
```

# 1. ROS2 Project Creation

*# 5. modify configuration file: add node projection in workspace\_name/src/package\_name/setup.py*

```
1  from setuptools import find_packages, setup
2
3  package_name = 'package_name'
4
5  setup(
6      name=package_name,
7      version='0.0.0',
8      packages=find_packages(exclude=['test']),
9      data_files=[
10         ('share/ament_index/resource_index/packages',
11          ['resource/' + package_name]),
12         ('share/' + package_name, ['package.xml']),
13     ],
14     install_requires=['setuptools'],
15     zip_safe=True,
16     maintainer='wenminggong',
17     maintainer_email='nandalmj@163.com',
18     description='TODO: Package description',
19     license='TODO: License declaration',
20     tests_require=['pytest'],
21     entry_points={
22         'console_scripts': [
23             'my_node_1 = package_name.my_node_1:main'
24         ],
25     },
26 )
```

# add for every node, if a node is created using the command “*ros2 pkg create ...*”, it will be added by default



# 1. ROS2 Project Creation

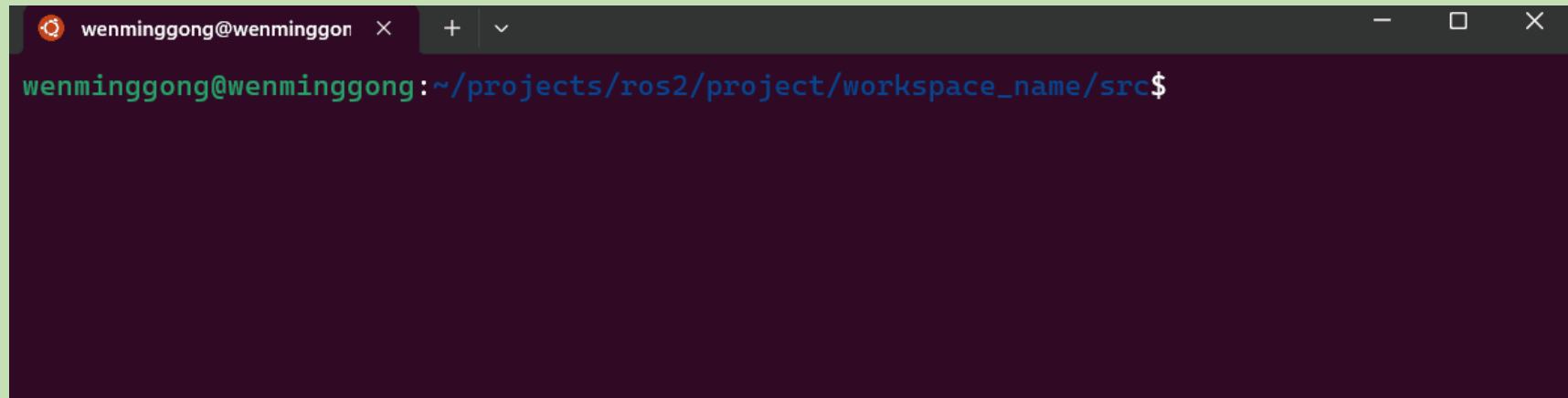
*# 6. build and compile project*

*# 6.1 go back to workspace\_name directory*

*cd ..*

*# 6.2 use colcon tool to build and compile project*

*colcon build --packages-select package\_name*

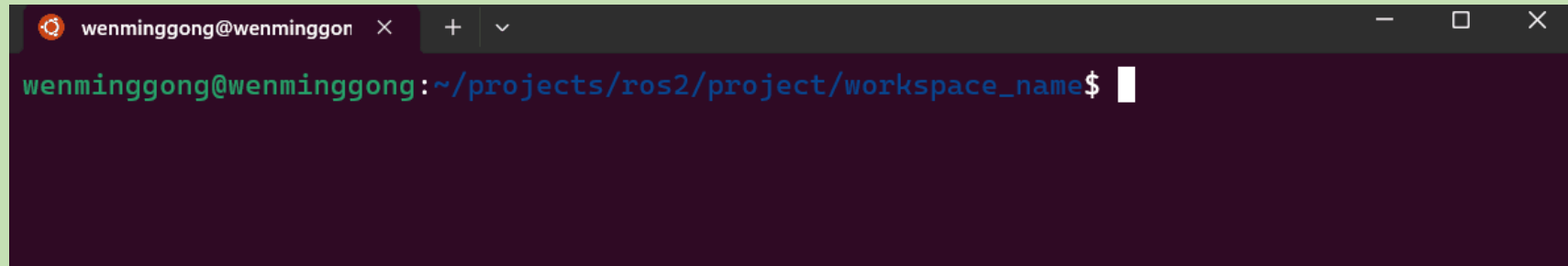
A terminal window with a dark purple background and a light green title bar. The title bar contains the text 'wenminggong@wenminggon' followed by a close button 'X', a plus sign '+', and a dropdown arrow 'v'. The terminal shows the prompt 'wenminggong@wenminggong:' followed by the directory path '~/projects/ros2/project/workspace\_name/src\$' in a light blue font. The rest of the terminal area is empty.

```
wenminggong@wenminggong:~/projects/ros2/project/workspace_name/src$
```

# 1. ROS2 Project Creation

*# 7. set environment variables for ROS2  
. install/setup.bash*

*# 8. run my\_node\_1  
ros2 run package\_name my\_node\_1*

A screenshot of a terminal window with a dark purple background. The window title bar shows 'wenminggong@wenminggon' and standard window controls. The terminal prompt is 'wenminggong@wenminggong:~/projects/ros2/project/workspace\_name\$' with a white cursor. The terminal is currently empty.

```
wenminggong@wenminggong:~/projects/ros2/project/workspace_name$
```

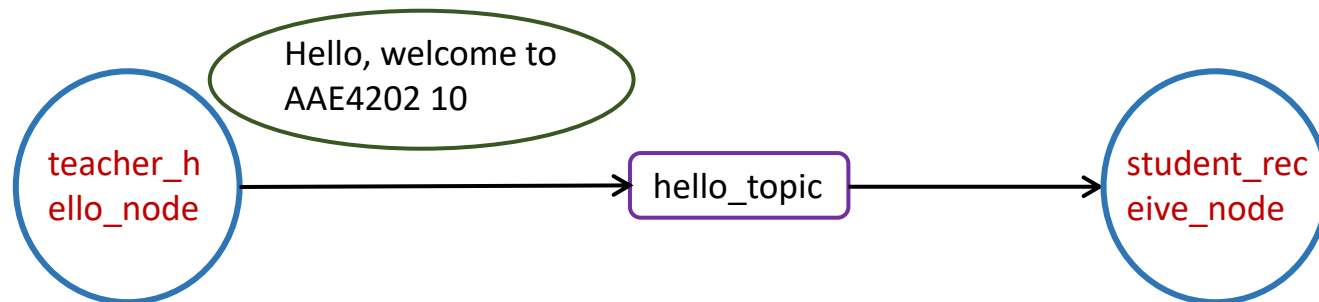
1. ROS2 Project Creation
2. An Example of using Topics Communication
3. Project Assignment 1

## 2. An Example of using Topics Communication

Slide 12 of 29

### Task Description

- Practice using **topics** communication as it is most frequently used
- A teacher node called *teacher\_hello\_node*, it is responsible for publishing a “hello topic” to the student node
  - Send the hello message at 1hz (once a second)
  - Hello message with increased number: “Hello, welcome to AAE4202 10”
- A student node called *student\_receive\_node*, it is required to receive hello message from the teacher node
  - Print the message in the screen



## 2. An Example of using Topics Communication

---

### **Procedure**

- Create new project (workspace, package, node)
- Write down codes for publisher node and subscriber node
- Modify configuration file
- Compile and run

## 2. An Example of using Topics Communication

Slide 14 of 29

### teacher\_hello\_node

```
1  # import ros2 python api
2  import rclpy
3  from rclpy.node import Node
4  # import standard message format
5  from std_msgs.msg import String
```

import necessary packages

```
6
7  # define node class
8  > class TeacherPublisher(Node): ...
```

teacher node implementation

```
41
42  def main():
43      # initialize ros2
44      rclpy.init()
45
46      # execute callbacks until shutdown
47      rclpy.spin(node=TeacherPublisher(node_name="teacher_hello_node"))
48
49      # shutdown a previously initialization
50      rclpy.shutdown()
51
```

main function implementation

```
52  if __name__ == '__main__':
53      main()
54
```

## 2. An Example of using Topics Communication

### teacher\_hello\_node

```
7 # define node class
8 class TeacherPublisher(Node):
9     def __init__(self, node_name: str) -> None:
10         super().__init__(node_name)
11         # print info
12         self.get_logger().info("teacher node created!")
13
14         # create a topic publisher
15         # parameters:
16         #   # msg_type: the type of message
17         #   # topic: the name of topic
18         #   # qos_profile: a QoSProfile for setting the communication policy or a history depth to store
19         self.hello_publisher = self.create_publisher(msg_type=String, topic="hello_topic", qos_profile=10)
20
21         # create a timer
22         # parameters:
23         #   # timer_period_sec: the period of the timer
24         #   # callback: a user-defined callback function that is called when the timer expires
25         self.timer = self.create_timer(timer_period_sec=1.0, callback=self._publish)
26
27         # create a counter
28         self.counter = 1
29
30     def _publish(self):
31         # create hello message
32         hello_message = String()
33         hello_message.data = "Hello, welcome to AAE4202 {}".format(self.counter)
34
35         # publish hello message
36         self.hello_publisher.publish(msg=hello_message)
37         # print info
38         self.get_logger().info("publish: {}".format(hello_message.data))
39
40         self.counter += 1
```

#### initialization function

- print log info
- create a topic publisher
- create a timer

#### publishing function

- create hello message
- publish hello message
- print log info

## 2. An Example of using Topics Communication

Slide 16 of 29

### student\_receive\_node

```
1  # import ros2 python api
2  ✓ import rclpy
3  from rclpy.node import Node
4
5  # import standard message format
6  from std_msgs.msg import String
7
```

import necessary packages

```
8
9  # define node class
10 > class StudentSubscriber(Node): ...
26
```

teacher node implementation

```
27
28  ✓ def main():
29      # initialize ros2
30      rclpy.init()
31
32      # execute callbacks until shutdown
33      rclpy.spin(node=StudentSubscriber(node_name="student_receive_node"))
34
35      # shutdown a previously initialization
36      rclpy.shutdown()
37
```

main function implementation

```
38
39  ✓ if __name__ == '__main__':
40      main()
41
```



## 2. An Example of using Topics Communication

Slide 17 of 29

### student\_receive\_node

```
9  # define node class
10 class StudentSubscriber(Node):
11     def __init__(self, node_name: str) -> None:
12         super().__init__(node_name)
13         # print info
14         self.get_logger().info("student node created!")
15
16         # create a topic subscriber
17         # parameters:
18         #   # msg_type: the type of message
19         #   # topic: the name of topic
20         #   # qos_profile: a QoSProfile for setting the communication policy or a history depth to store
21         self.hello_subscriber = self.create_subscription(msg_type=String, topic="hello_topic", callback=self._subscribe, qos_profile=1)
22
23     def _subscribe(self, hello_message):
24         # print info
25         self.get_logger().info("received message: {}".format(hello_message.data))
26
```

initialization function

- print log info
- create a topic subscriber

subscribing function

- print received message

## 2. An Example of using Topics Communication

Slide 18 of 29

modify configuration file: setup.py

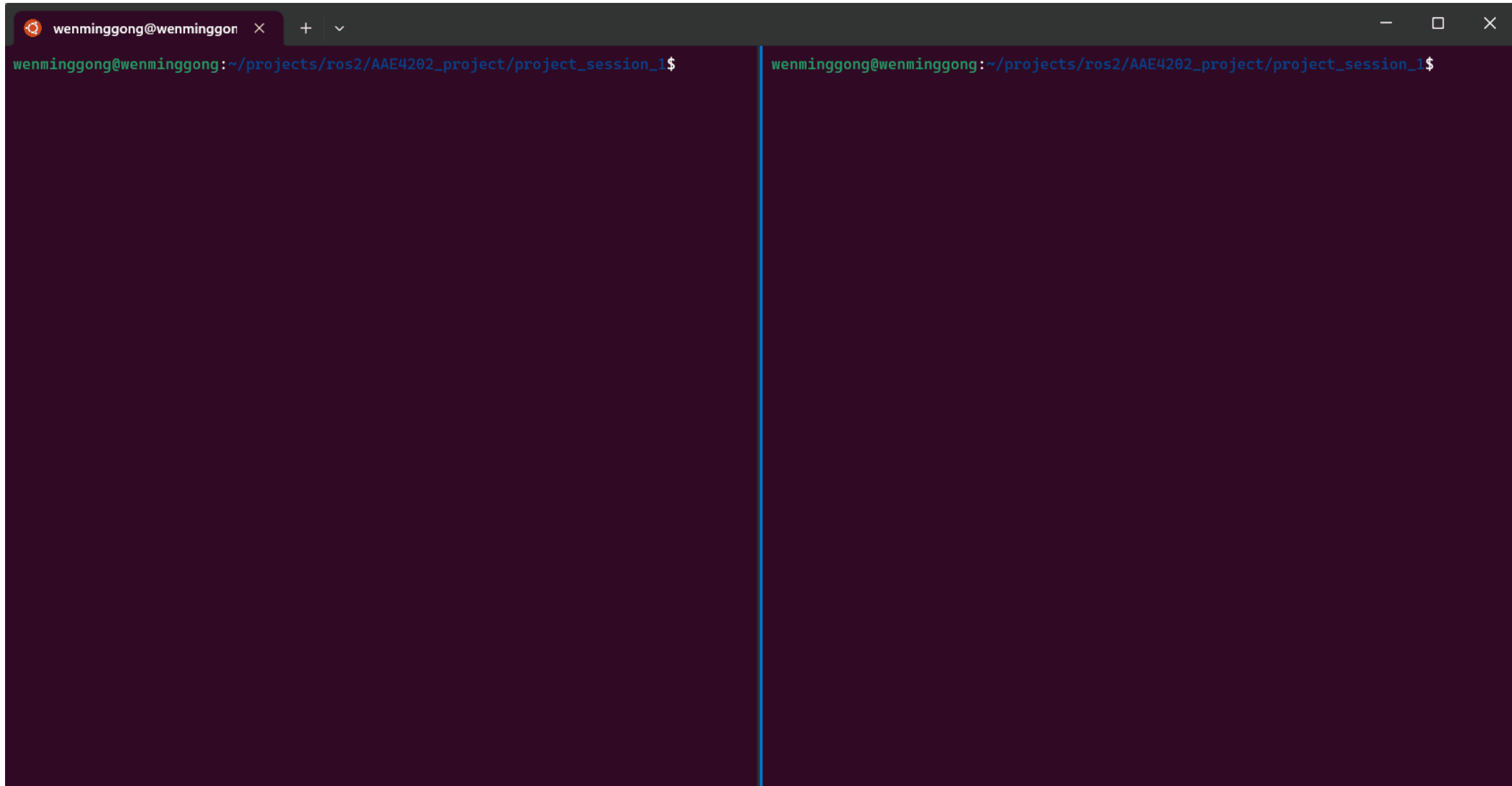
```
setup.py X
src > teacher_student_pkg > setup.py > ...
1  from setuptools import find_packages, setup
2
3  package_name = 'teacher_student_pkg'
4
5  setup(
6      name=package_name,
7      version='0.0.0',
8      packages=find_packages(exclude=['test']),
9      data_files=[
10         ('share/ament_index/resource_index/packages',
11          ['resource/' + package_name]),
12         ('share/' + package_name, ['package.xml']),
13     ],
14     install_requires=['setuptools'],
15     zip_safe=True,
16     maintainer='wenminggong',
17     maintainer_email='nandalmj@163.com',
18     description='TODO: Package description',
19     license='TODO: License declaration',
20     tests_require=['pytest'],
21     entry_points={
22         'console_scripts': [
23             'teacher_hello_node = teacher_student_pkg.teacher_hello_node:main',
24             'student_receive_node = teacher_student_pkg.student_receive_node:main'
25         ],
26     },
27 )
```

# add node projection

## 2. An Example of using Topics Communication

Slide 19 of 29

### Experimental Results



1. ROS2 Project Creation
2. An Example of using Topics Communication
3. Project Assignment 1

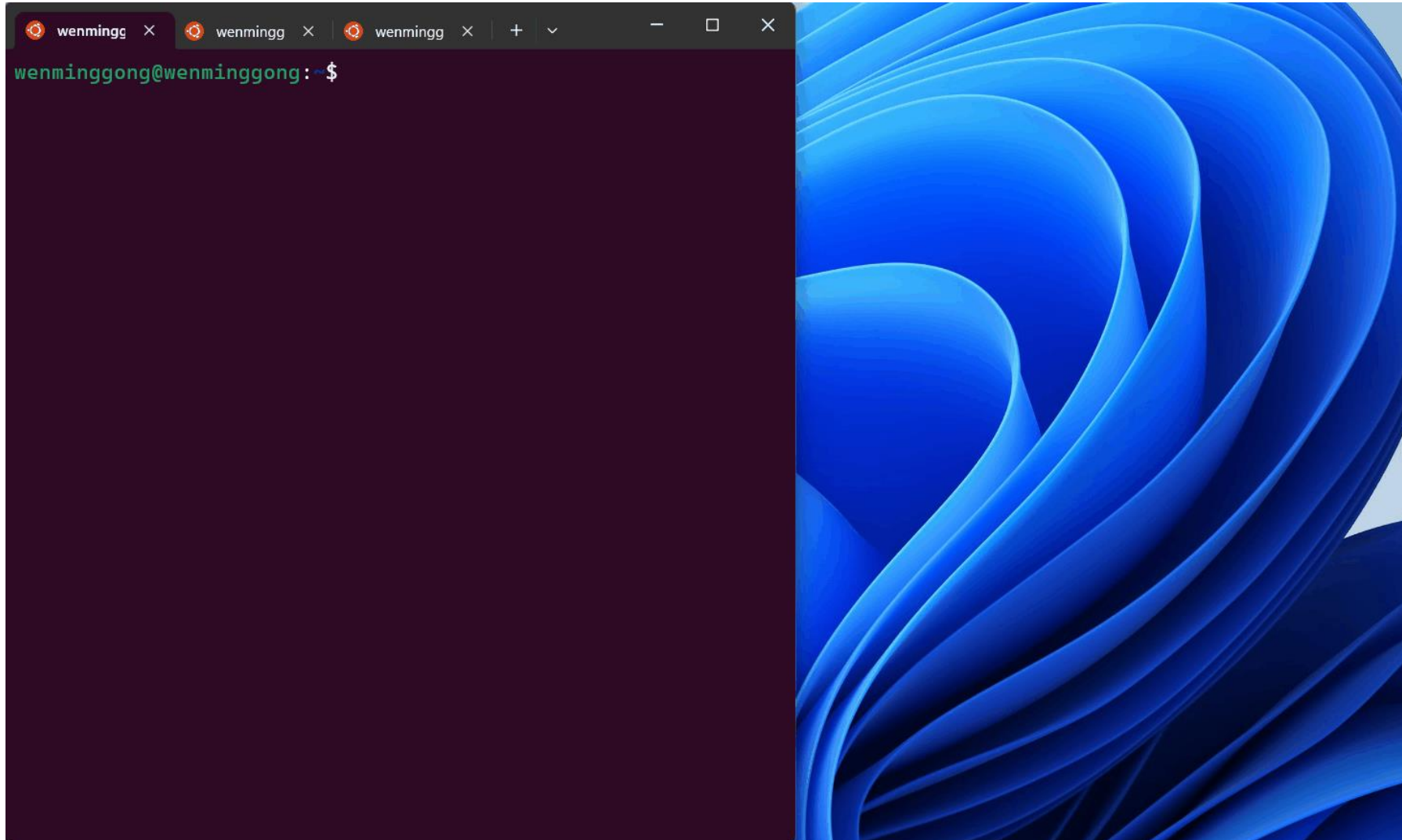
## Prerequisites

- Turtlesim: a lightweight simulator for learning ROS2, it illustrates what ROS2 does at the most basic level
- Two common nodes
  - turtlesim\_node: open the turtlesim simulator
  - turtlesim\_teleop\_key: control the turtle in the simulator by keyboard

# 3. Project Assignment 1

Slide 22 of 29

## Prerequisites

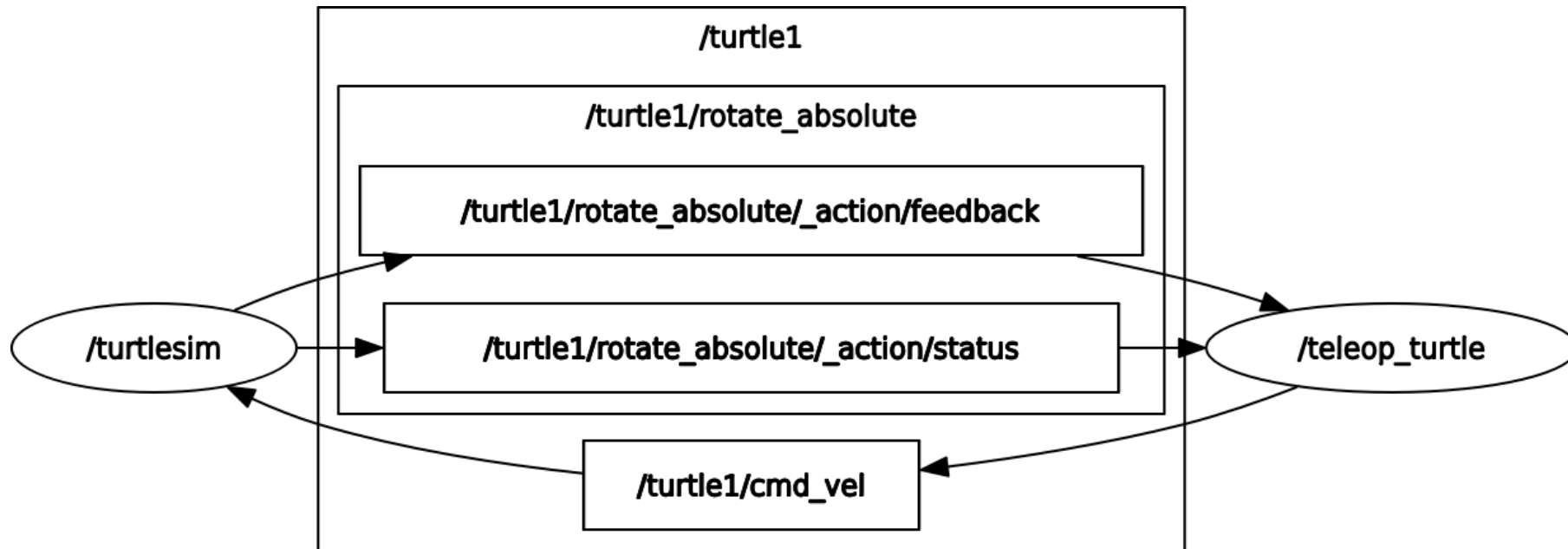


# 3. Project Assignment 1

## Prerequisites

- rqt: a graphical user interface (GUI) tool for ROS2
- Use rqt tool to show the ROS2 graph

*rqt\_graph*



## Task Description

- Implement a turtle control node
  - Control the turtle to move around a circle
  - Publish control message at a frequency of 2hz
  - The radius of the circle is 2
- Implement a turtle pose show node
  - Print the turtle's position in real time, including x, y, and theta
  - Print the radius of the circle which the turtle follows



## Task Description

- Assignment requirements
  - Record the process and the results of this experiment
  - Write the experiment report to describe the experiment **process and results**
  - Send the experiment report to **the TA (Liu) by email** before the next project session (**before Thursday week 5**)
  - Only one submission per group
  - Please submit the report in **pdf** format and sign the **names of all members**

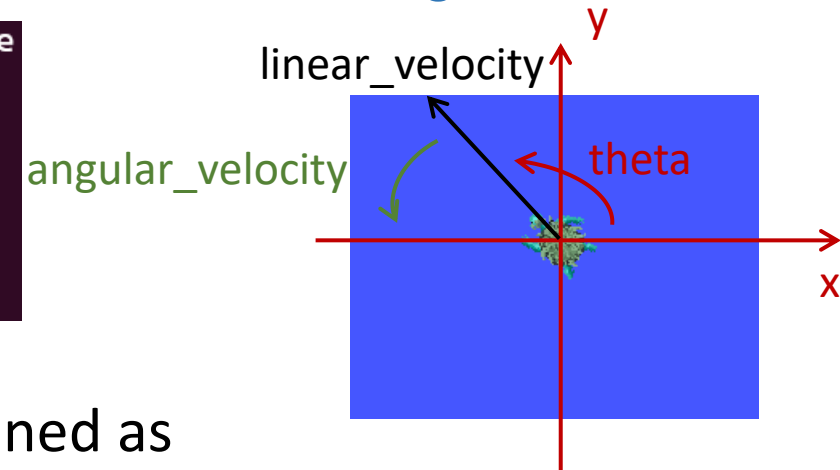
# 3. Project Assignment 1

## Task Description

- The turtlesim\_node publishes its pose through `/turtle1/pose` topic, and subscribes control command through `/turtle1/cmd_vel` topic
- The message format of `/turtle1/pose` topic is defined as `turtlesim/msg/Pose`

```
wenminggong@wenminggong:~$ ros2 interface show turtlesim/msg/Pose
float32 x
float32 y
float32 theta

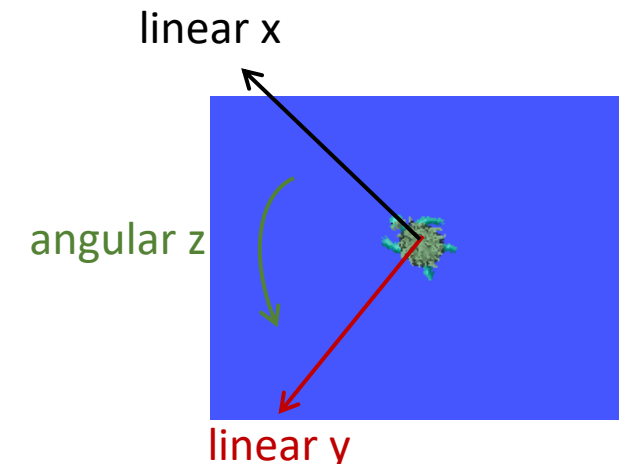
float32 linear_velocity
float32 angular_velocity
```



- The message format of `/turtle1/cmd_vel` topic is defined as `geometry_msgs/msg/Twist`

```
wenminggong@wenminggong:~$ ros2 interface show geometry_msgs/msg/Twist
# This expresses velocity in free space broken into its linear and angular parts.

Vector3  linear
  float64 x
  float64 y
  float64 z
Vector3  angular
  float64 x
  float64 y
  float64 z
```



## Task Description

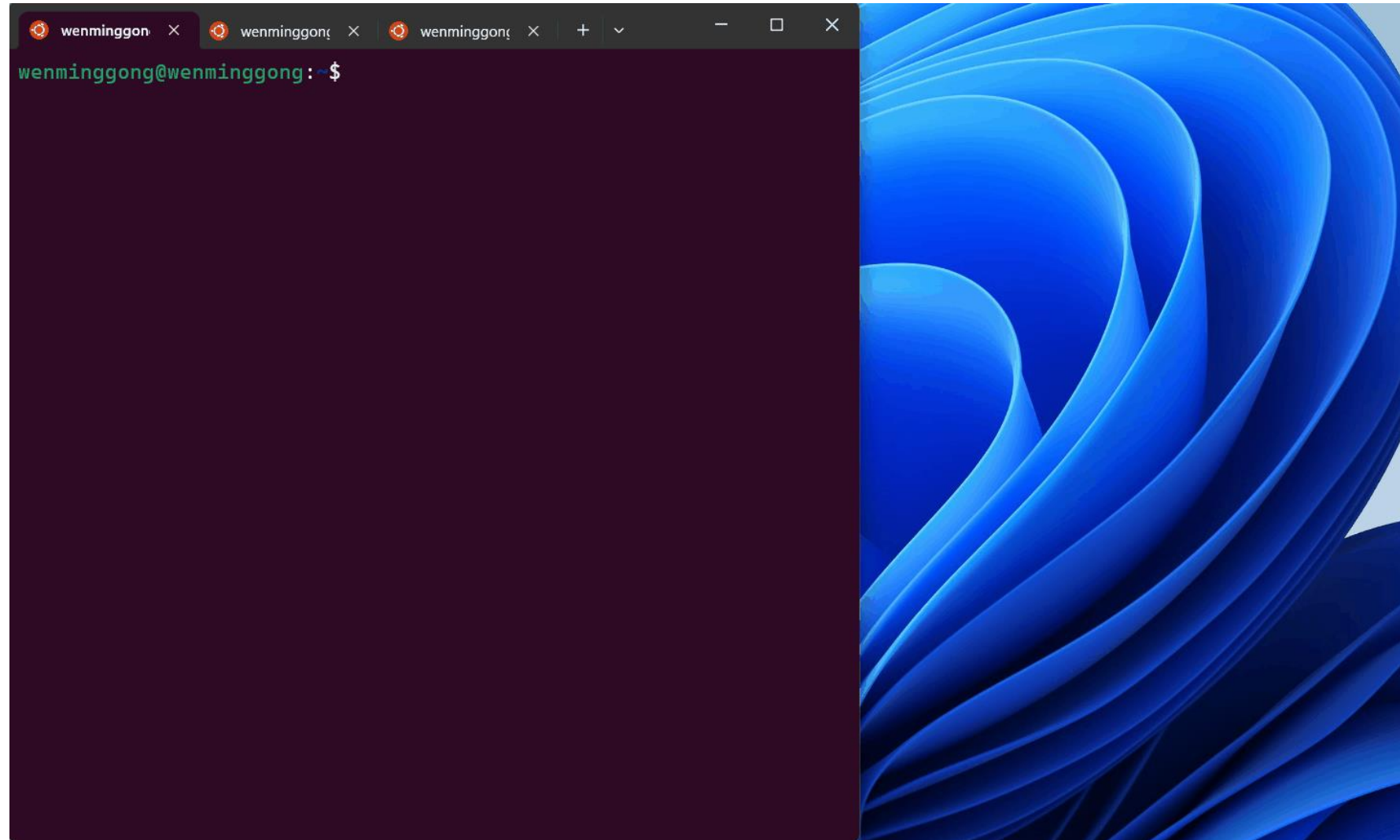
- Message dependencies need to be added when the package is created

```
# create a new python package with a node  
ros2 pkg create package_name --build-type ament_python --dependencies rclpy  
turtlesim geometry_msgs --node-name my_node_1
```

# 3. Project Assignment 1

Slide 28 of 29

## Demonstration



# Thank You!

**Dr. Hailong Huang**

Assistant Professor

Department of Aeronautical and Aviation Engineering

The Hong Kong Polytechnic University

<https://sites.google.com/view/hailong-huang/home>

Email: [hailong.huang@polyu.edu.hk](mailto:hailong.huang@polyu.edu.hk)