



Week 9: ROS2

ROS2 Introduction Hybrid 4 + Lab4

- Hybrid ROS2 introduction: ROS2 introductory tutorials.
- Lab – 4 ROS2 installation
- <https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools.html>
 - Nodes
 - Topics
 - Services
 - Actions

The big Picture

- Create Python packages **aisd_vision** and **aisd_motion**, use **MediaPipe Hands + OpenCV** to detect hand positions and command motion.
- Create **aisd_hearing** packages using **Whisper (STT)** and **gTTS (TTS)**; record audio, transcribe speech.
- Deploy these packages in real application (Create3 robot)



The big Picture

OpenCV (Open-Source Computer Vision Library)

A powerful Python/C++ library for working with images and videos.

Handles low-level tasks like capturing frames from a webcam, resizing, color conversion, drawing, etc.

OpenCV reads the camera stream and provides each frame to the next stage (MediaPipe).

MediaPipe Hands

A **pre-trained hand-tracking model** developed by Google.

Detects **21 key landmarks** (fingertips, joints, wrist) for each hand in real time.

Returns normalized (x, y, z) positions of each point in the image.

The big Picture

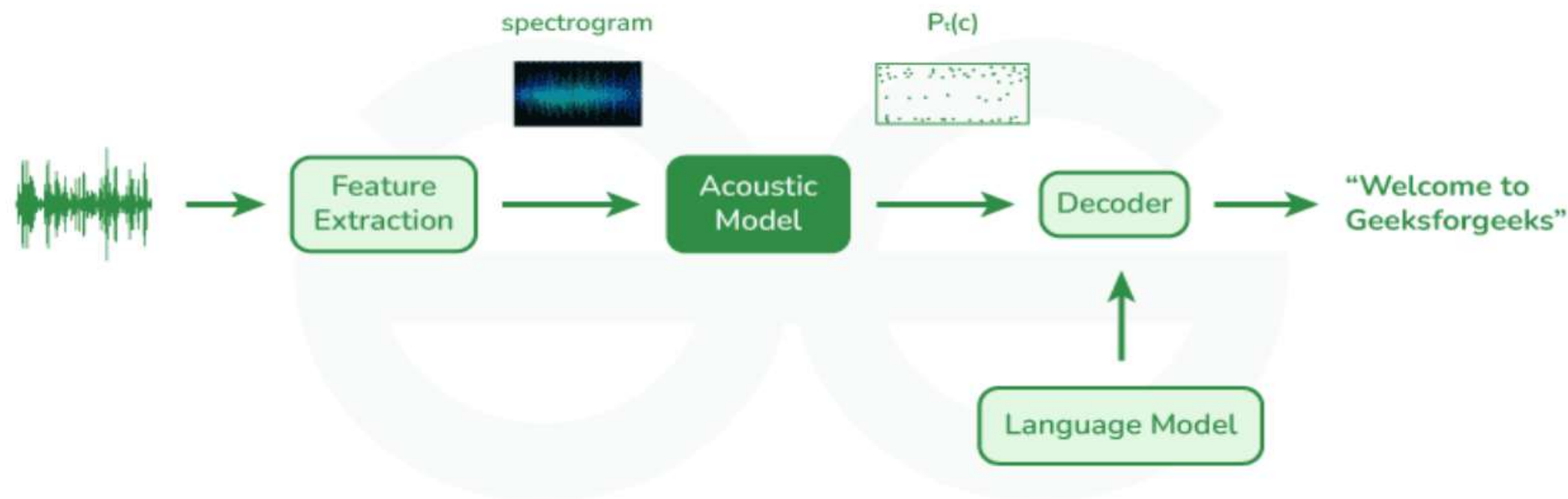
Whisper (Speech-to-Text, STT)

An **AI speech-recognition model** by **OpenAI**.

Converts spoken audio (from a microphone or WAV file) into written text.

Handles many languages and accents, robust to noise.

In your `aisd_hearing` package, it turns your voice into text strings.



The big Picture

- Train Physical Reinforcement learning agents (Next semester)



ROS2 Ecosystem

- ROS2 distributions (version):
<https://docs.ros.org/en/rolling/Releases.html>
- We will use Humble

Humble Hawksbill	May 23, 2022		May 2027	Audrow Nash
------------------	--------------	--	----------	-------------

ROS2 Ecosystem

Installations

- <https://docs.ros.org/en/humble/Installation/Ubuntu-Install-Debs.html>
- If you never worked in Ubuntu, take a look at the Linux revision posted in Brightspace. You need to know basic commands like these

```
ali@Ali:~$ pwd
/home/ali
ali@Ali:~$ ls
'2024-11-21 18-38-29.mkv'  ali22  Desktop  install  Pictures  ros2_iron  Templates
'2024-11-21 18-40-33.mkv'  aliros  Documents  log      Public    ros2_ws   Videos
AAAA                     build  Downloads  Music    Ros2_course  snap
ali@Ali:~$ ls build
COLCON_IGNORE  my_py_pkg
ali@Ali:~$ cd home
bash: cd: home: No such file or directory
ali@Ali:~$ cd ..
ali@Ali:/home$ pwd
/home
ali@Ali:/home$ cd ali
ali@Ali:~$ pwd
/home/ali
ali@Ali:~$ mkdir ROS2INTALL
ali@Ali:~$ ls
'2024-11-21 18-38-29.mkv'  ali22  Desktop  install  Pictures  ROS2INTALL  snap
'2024-11-21 18-40-33.mkv'  aliros  Documents  log      Public    ros2_iron  Templates
AAAA                     build  Downloads  Music    Ros2_course  ros2_ws   Videos
```


ROS2 Ecosystem

- After you install ROS2 humble distribution, check if it working or not. If you type ROS2 and then double tab you will see a list commands can be used.
- Try `ros2 run demo_nodes_cpp talker` ,,, This will run a C++ node called Talker inside a C++ Package called `demo_nodes_cpp`. The node will publish Hello World every second.

```
ali@Ali:~$ ros2
action          interface      run
bag             launch        security
component       lifecycle     service
daemon          multicast    topic
doctor          node         --use-python-default
lt-buffering
extension_points param          wtf
extensions      pkg

ali@Ali:~$ ros2 run demo_nodes_cpp talker
[INFO] [1732233149.314858111] [talker]: Publishing: 'Hello World: 1'
[INFO] [1732233150.314823861] [talker]: Publishing: 'Hello World: 2'
[INFO] [1732233151.314818338] [talker]: Publishing: 'Hello World: 3'
```

ROS2 Ecosystem

- Now in another terminal you can type `ros2 run demo_nodes_cpp listener`. This will run a C++ node that will listen to the published message and Type "I heard:...."

```
INFO] [1732233179.314687799] [talker]: Publishing: 'Hello World: 31'
INFO] [1732233180.314716117] [talker]: Publishing: 'Hello World: 32'
INFO] [1732233181.314675469] [talker]: Publishing: 'Hello World: 33'
INFO] [1732233182.314661930] [talker]: Publishing: 'Hello World: 34'
INFO] [1732233183.314679035] [talker]: Publishing: 'Hello World: 35'
INFO] [1732233184.314674357] [talker]: Publishing: 'Hello World: 36'
INFO] [1732233185.314672505] [talker]: Publishing: 'Hello World: 37'
INFO] [1732233186.314677329] [talker]: Publishing: 'Hello World: 38'
INFO] [1732233187.314630603] [talker]: Publishing: 'Hello World: 39'
INFO] [1732233188.314655009] [talker]: Publishing: 'Hello World: 40'
INFO] [1732233189.314638565] [talker]: Publishing: 'Hello World: 41'
INFO] [1732233190.314652161] [talker]: Publishing: 'Hello World: 42'
INFO] [1732233191.314635418] [talker]: Publishing: 'Hello World: 43'

ali@Ali: ~ 75x19
ali@Ali:~$ ros2 run demo_nodes_cpp listener
INFO] [1732233185.315018424] [listener]: I heard: [Hello World: 37]
INFO] [1732233186.315029885] [listener]: I heard: [Hello World: 38]
INFO] [1732233187.314964133] [listener]: I heard: [Hello World: 39]
INFO] [1732233188.314946060] [listener]: I heard: [Hello World: 40]
INFO] [1732233189.314926355] [listener]: I heard: [Hello World: 41]
INFO] [1732233190.315009620] [listener]: I heard: [Hello World: 42]
INFO] [1732233191.315160879] [listener]: I heard: [Hello World: 43]
```

Building Packages in ROS2

- We will use a Library called Colcon.
 - <https://docs.ros.org/en/humble/Tutorials/Beginner-Client-Libraries/Colcon-Tutorial.html>
- Install Colcon as follows

```
ali@Ali:~$ sudo apt install python3-colcon-common-extensions
[sudo] password for ali:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
python3-colcon-common-extensions is already the newest version (0.3.0-1).
The following packages were automatically installed and are no longer required:
  libwpe-1.0-1 libwpebackend-fdo-1.0-1
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 364 not upgraded.
```


Building Workspace in ROS2

- Build now you first Workspace: using colcon build

```
ali@Ali:~$ pwd
/home/ali
ali@Ali:~$ ls
'2024-11-23 19-30-20.mkv'  install  ros2_iron
build                     log      snap
Desktop                  Music    Templates
Documents                Pictures Videos
Downloads                Public
ali@Ali:~$ mkdir ros2_ws
ali@Ali:~$ ls
'2024-11-23 19-30-20.mkv'  install  ros2_iron
build                     log      ros2_ws
Desktop                  Music    snap
Documents                Pictures Templates
Downloads                Public   Videos
ali@Ali:~$ cd ros2_ws
ali@Ali:~/ros2_ws$ colcon build

Summary: 0 packages finished [0.17s]
ali@Ali:~/ros2_ws$ ls
build  install  log
ali@Ali:~/ros2_ws$ ls install
COLCON_IGNORE  _local_setup_util_ps1.py  setup.ps1
local_setup.bash  _local_setup_util_sh.py  setup.sh
local_setup.ps1  local_setup.zsh          setup.zsh
local_setup.sh   setup.bash
```

Building Workspace in ROS2

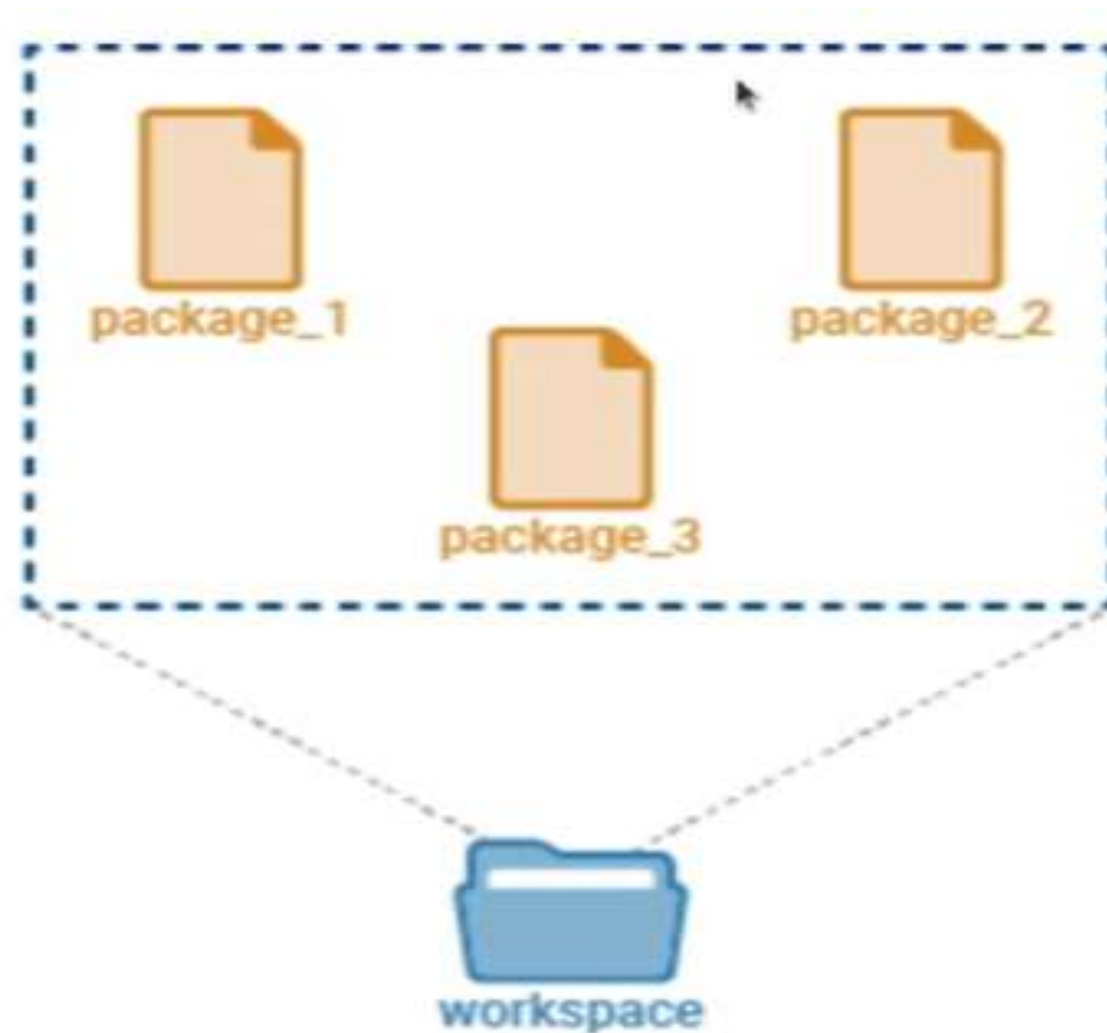
- Build now you first Packages: using colcon build

```
ali@Ali:~$ pwd
/home/ali
ali@Ali:~$ ls
'2024-11-23 19-30-20.mkv'  install  ros2_iron
build                     log      snap
Desktop                   Music    Templates
Documents                  Pictures Videos
Downloads                  Public
ali@Ali:~$ mkdir ros2_ws
ali@Ali:~$ ls
'2024-11-23 19-30-20.mkv'  install  ros2_iron
build                     log      ros2_ws
Desktop                   Music    snap
Documents                  Pictures Templates
Downloads                  Public   Videos
ali@Ali:~$ cd ros2_ws
ali@Ali:~/ros2_ws$ colcon build

Summary: 0 packages finished [0.17s]
ali@Ali:~/ros2_ws$ ls
build  install  log
ali@Ali:~/ros2_ws$ ls install
COLCON_IGNORE  _local_setup_util_ps1.py  setup.ps1
local_setup.bash  _local_setup_util_sh.py  setup.sh
local_setup.ps1  local_setup.zsh          setup.zsh
local_setup.sh   setup.bash
```

Building Packages in ROS2

- Every workspace can have several packages. Each package contains modules (codes that send and receives messages).



Building Packages in ROS2

- Every workspace can have several packages. Each package contains modules (codes that send and receive messages).
- You can build a package called `my_py_pkg` using `ament` as follows:
- Note that you need `rclpy` to be able to write python codes.

```
ali@Ali:~$ pwd
/home/ali
ali@Ali:~$ ls
'2024-11-24 19-05-27.mkv' Desktop Downloads log Pictures ros2_iron snap Videos
build Documents install Music Public ros2_ws Templates
ali@Ali:~$ ls ros2_ws
build install log
ali@Ali:~$ cd ros2_ws
ali@Ali:~/ros2_ws$ mkdir src
ali@Ali:~/ros2_ws$ ls
build install log src
ali@Ali:~/ros2_ws$ cd src
ali@Ali:~/ros2_ws/src$ ros2 pkg create my_py_pkg --build-type ament_python --dependencies rclpy
```


Building Packages in ROS2

- Every workspace can have several packages. Each package contains modules (codes that send and receive messages).
- You can build a package called `my_py_pkg` using `ament` as follows:
- Note that you need `rclpy` to be able to write python codes.

```
ali@Ali:~$ pwd
/home/ali
ali@Ali:~$ ls
'2024-11-24 19-05-27.mkv' Desktop Downloads log Pictures ros2_iron snap Videos
build Documents install Music Public ros2_ws Templates
ali@Ali:~$ ls ros2_ws
build install log
ali@Ali:~$ cd ros2_ws
ali@Ali:~/ros2_ws$ mkdir src
ali@Ali:~/ros2_ws$ ls
build install log src
ali@Ali:~/ros2_ws$ cd src
ali@Ali:~/ros2_ws/src$ ros2 pkg create my_py_pkg --build-type ament_python --dependencies rclpy
```

Building Packages in ROS2

- Now check that your Python Package was build.
- You need to redo “colcon build” every time you install or update any of the packages inside the workspace.

```
ali@Ali:~$ cd ros2_ws
ali@Ali:~/ros2_ws$ colcon build
Starting >>> my_py_pkg
Finished <<< my_py_pkg [0.55s]

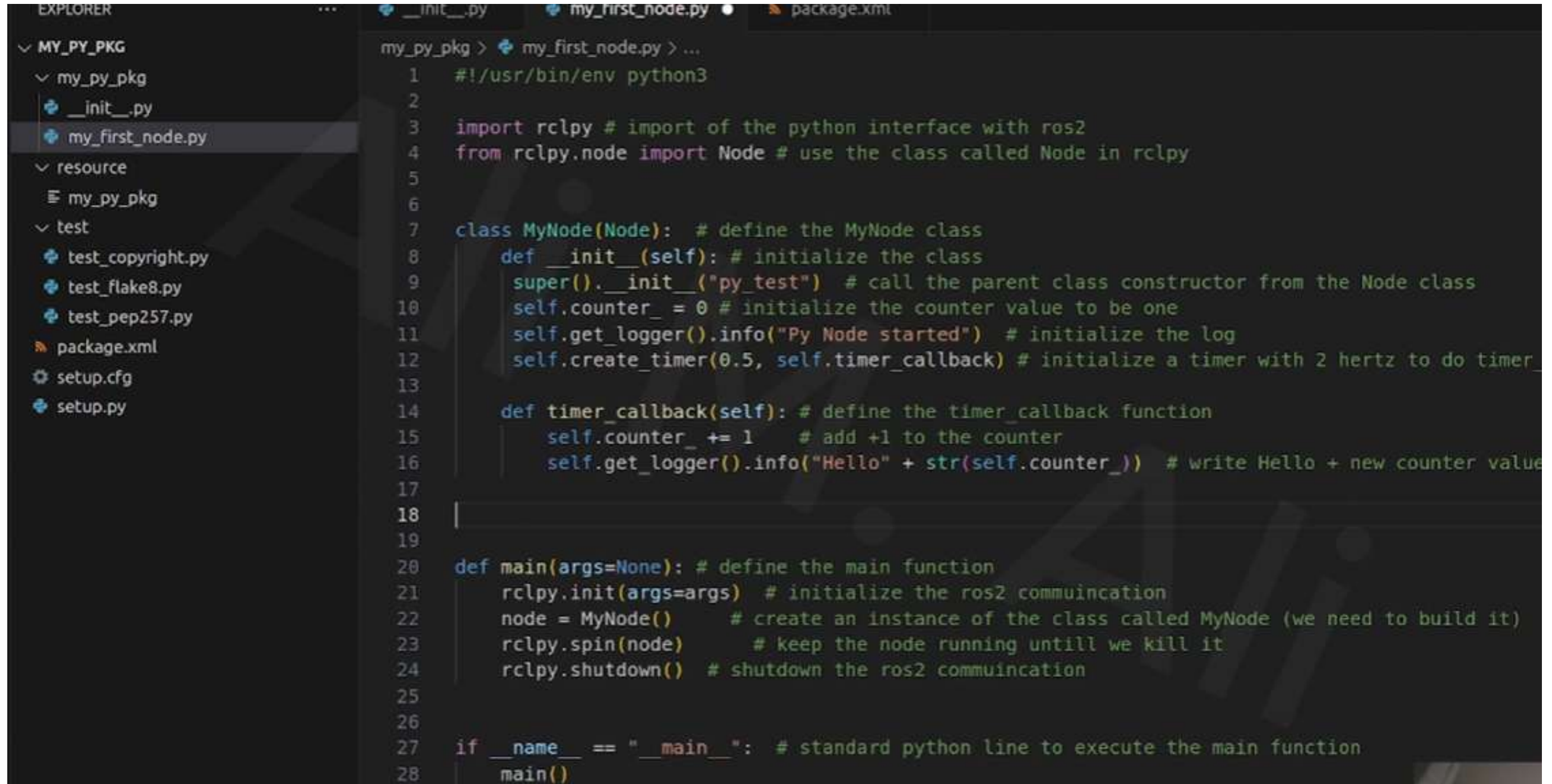
Summary: 1 package finished [0.72s]
ali@Ali:~/ros2_ws$ colcon build --packages-select my_py_pkg
Starting >>> my_py_pkg
Finished <<< my_py_pkg [0.54s]

Summary: 1 package finished [0.71s]
ali@Ali:~/ros2_ws$
```

Building Nodes in ROS2

```
ali@Ali:~$ cd ros2_ws
ali@Ali:~/ros2_ws$ ls
build  install  log  src
ali@Ali:~/ros2_ws$ cd src
ali@Ali:~/ros2_ws/src$ ls
my_cpp_pkg  my_py_pkg
ali@Ali:~/ros2_ws/src$ cd my_py_pkg
ali@Ali:~/ros2_ws/src/my_py_pkg$ ls
my_py_pkg  package.xml  resource  setup.cfg  setup.py  test
ali@Ali:~/ros2_ws/src/my_py_pkg$ cd my_py_pkg
ali@Ali:~/ros2_ws/src/my_py_pkg/my_py_pkg$ touch my_first_node.py
```


Building Nodes in ROS2



The image shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a package named 'MY_PY_PKG' with a subdirectory 'my_py_pkg' containing files like '__init__.py', 'my_first_node.py', 'test_copyright.py', 'test_flake8.py', 'test_pep257.py', 'package.xml', 'setup.cfg', and 'setup.py'. The code editor shows the implementation of 'my_first_node.py'.

```
1  #!/usr/bin/env python3
2
3  import rclpy # import of the python interface with ros2
4  from rclpy.node import Node # use the class called Node in rclpy
5
6
7  class MyNode(Node): # define the MyNode class
8      def __init__(self): # initialize the class
9          super().__init__("py_test") # call the parent class constructor from the Node class
10         self.counter_ = 0 # initialize the counter value to be one
11         self.get_logger().info("Py Node started") # initialize the log
12         self.create_timer(0.5, self.timer_callback) # initialize a timer with 2 hertz to do timer
13
14         def timer_callback(self): # define the timer_callback function
15             self.counter_ += 1 # add +1 to the counter
16             self.get_logger().info("Hello" + str(self.counter_)) # write Hello + new counter value
17
18
19
20 def main(args=None): # define the main function
21     rclpy.init(args=args) # initialize the ros2 commuincation
22     node = MyNode() # create an instance of the class called MyNode (we need to build it)
23     rclpy.spin(node) # keep the node running untill we kill it
24     rclpy.shutdown() # shutdown the ros2 commuincation
25
26
27 if __name__ == "__main__": # standard python line to execute the main function
28     main()
```