Getting Started With **ROS2** and **TurtleBot4**

Presenter: Adnan Abdullah
PhD Student
Dept. Of ECE, UF

RoboPI: Robot Perception
& Intellgence Laboratory

ECE | Florida
Electrical & Computer Engineering

UF | UNIVERSITY of FLORIDA

Outline

- ROS1 vs ROS2
  - Package
  - Workspace
  - Build
  - Node
  - Launch file
  - ROS master
  - API

- ROS2 with TurtleBot
  - Prepare your pc
  - Configure Raspberry Pi
  - Configure Create 3
  - How to connect
  - Example
    - Keyboard control (TeleOp)
    - Navigation and SLAM

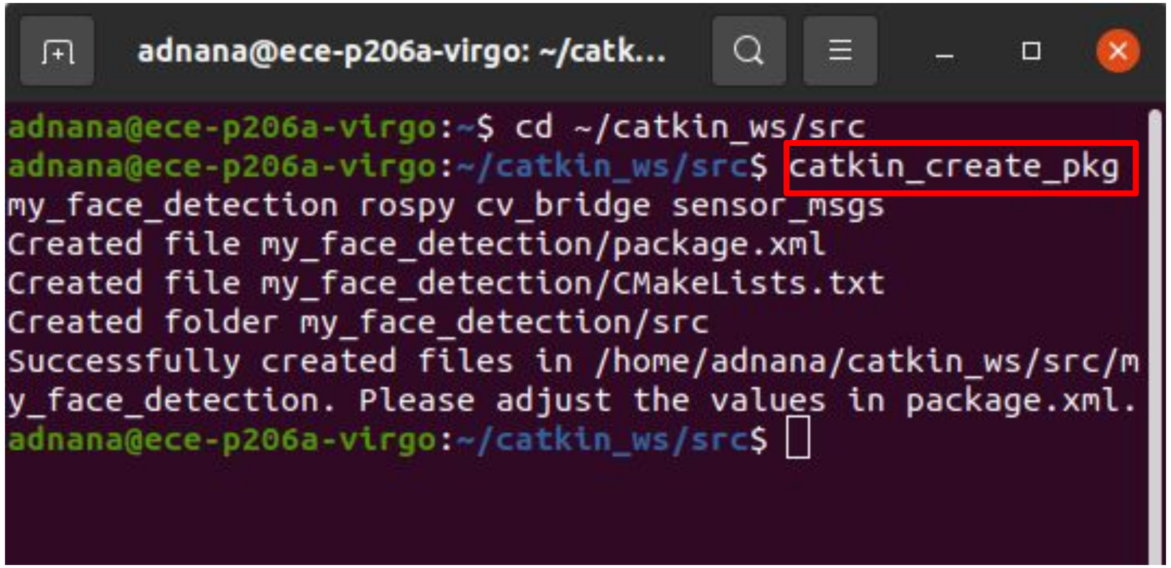
Creating a Package

ROS 1

- First, create a package with-

  `catkin_create_pkg`

- Then add any Cpp/Python file.

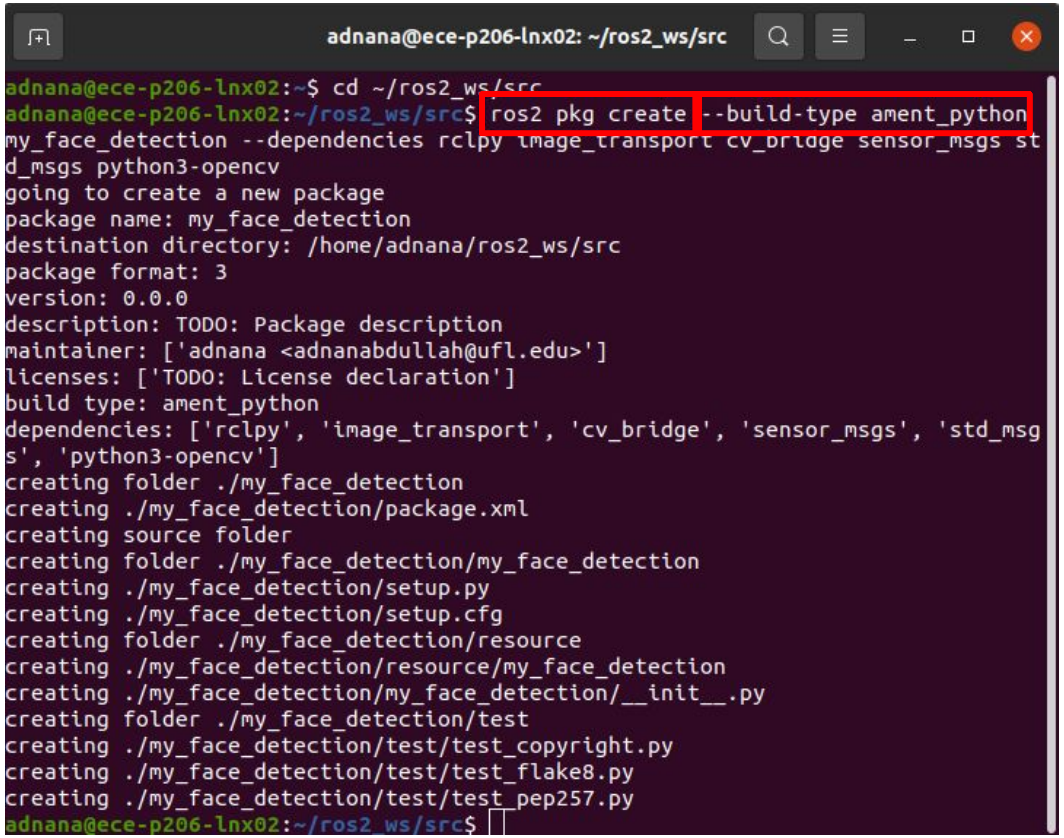


adnana@ece-p206a-virgo: ~/catk...

```
adnana@ece-p206a-virgo:~$ cd ~/catkin_ws/src
adnana@ece-p206a-virgo:~/catkin_ws/src$ catkin_create_pkg
my_face_detection rospy cv_bridge sensor_msgs
Created file my_face_detection/package.xml
Created file my_face_detection/CMakeLists.txt
Created folder my_face_detection/src
Successfully created files in /home/adnana/catkin_ws/src/m
y_face_detection. Please adjust the values in package.xml.
adnana@ece-p206a-virgo:~/catkin_ws/src$ 
```

ROS 2

- When creating the package, specify one build type: `ament_cmake` or `ament_python`.



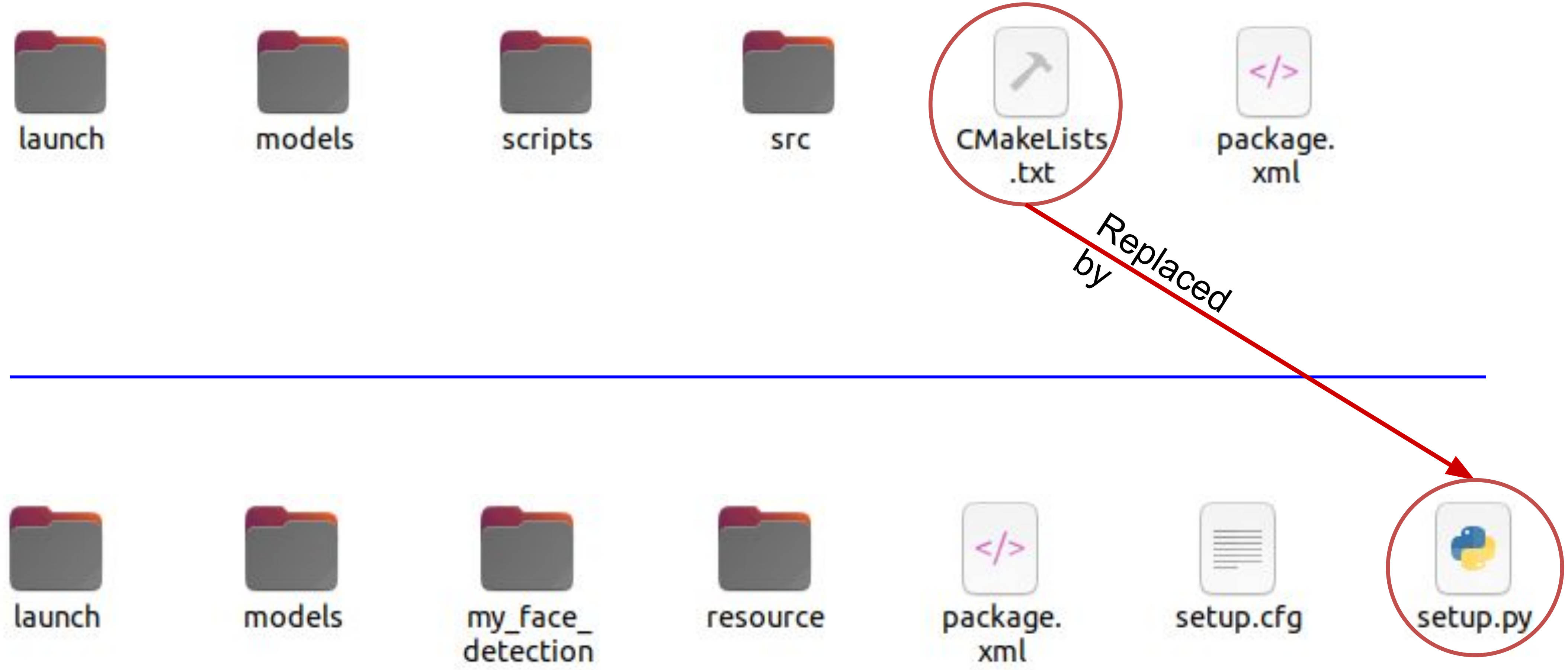
adnana@ece-p206-lnx02: ~/ros2_ws/src

```
adnana@ece-p206-lnx02:~$ cd ~/ros2_ws/src
adnana@ece-p206-lnx02:~/ros2_ws/src$ ros2 pkg create --build-type ament_python
my_face_detection --dependencies rclpy image_transport cv_bridge sensor_msgs st
d_msgs python3-opencv
going to create a new package
package name: my_face_detection
destination directory: /home/adnana/ros2_ws/src
package format: 3
version: 0.0.0
description: TODO: Package description
maintainer: ['adnana <adnanabdullah@ufl.edu>']
licenses: ['TODO: License declaration']
build type: ament_python
dependencies: ['rclpy', 'image_transport', 'cv_bridge', 'sensor_msgs', 'std_msg
s', 'python3-opencv']
creating folder ./my_face_detection
creating ./my_face_detection/package.xml
creating source folder
creating folder ./my_face_detection/my_face_detection
creating ./my_face_detection/setup.py
creating ./my_face_detection/setup.cfg
creating folder ./my_face_detection/resource
creating ./my_face_detection/resource/my_face_detection
creating ./my_face_detection/my_face_detection/__init__.py
creating folder ./my_face_detection/test
creating ./my_face_detection/test/test_copyright.py
creating ./my_face_detection/test/test_flake8.py
creating ./my_face_detection/test/test_pep257.py
adnana@ece-p206-lnx02:~/ros2_ws/src$ 
```

Workspace

ROS 1

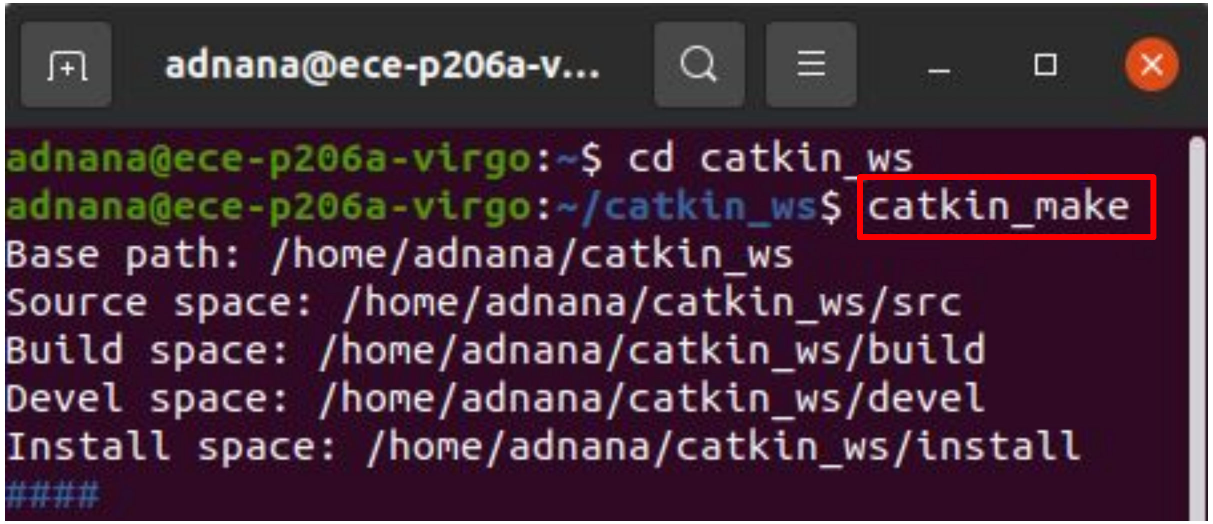
ROS 2



Building a package

ROS 1

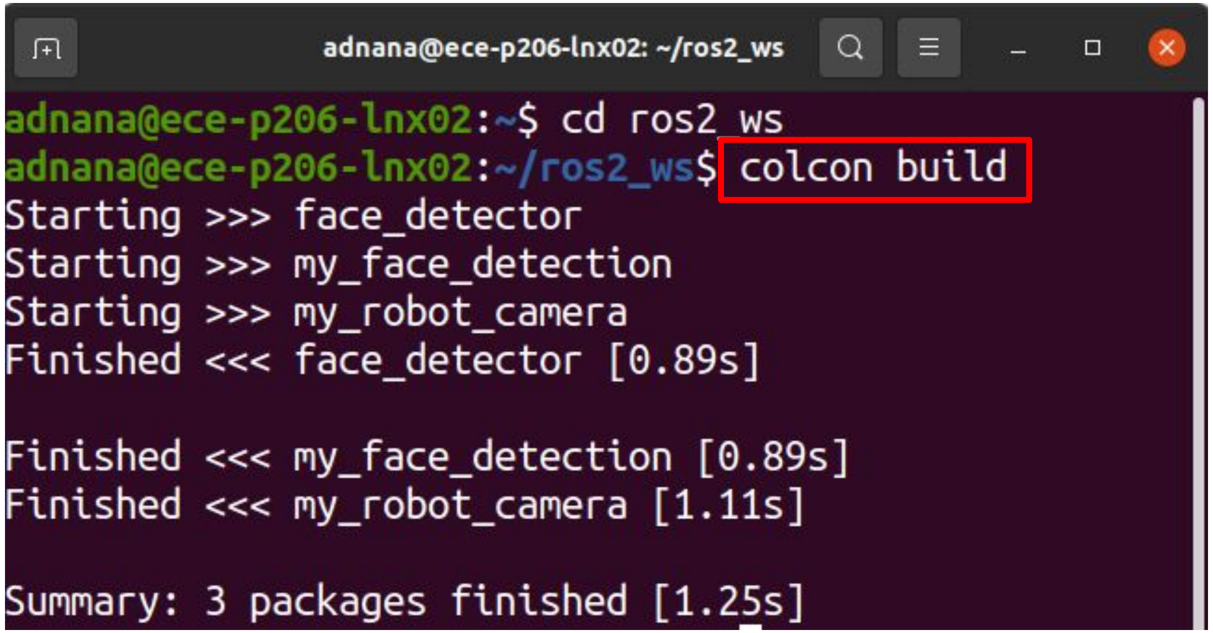
- `catkin_make` or `catkin_build` to build and install packages.



```
adnana@ece-p206a-virgo:~$ cd catkin_ws
adnana@ece-p206a-virgo:~/catkin_ws$ catkin_make
Base path: /home/adnana/catkin_ws
Source space: /home/adnana/catkin_ws/src
Build space: /home/adnana/catkin_ws/build
Devel space: /home/adnana/catkin_ws/devel
Install space: /home/adnana/catkin_ws/install
####
```

ROS 2

- Ament is the new building system.
- On top of that, there is colcon command line tool.
- To compile, use `colcon build`

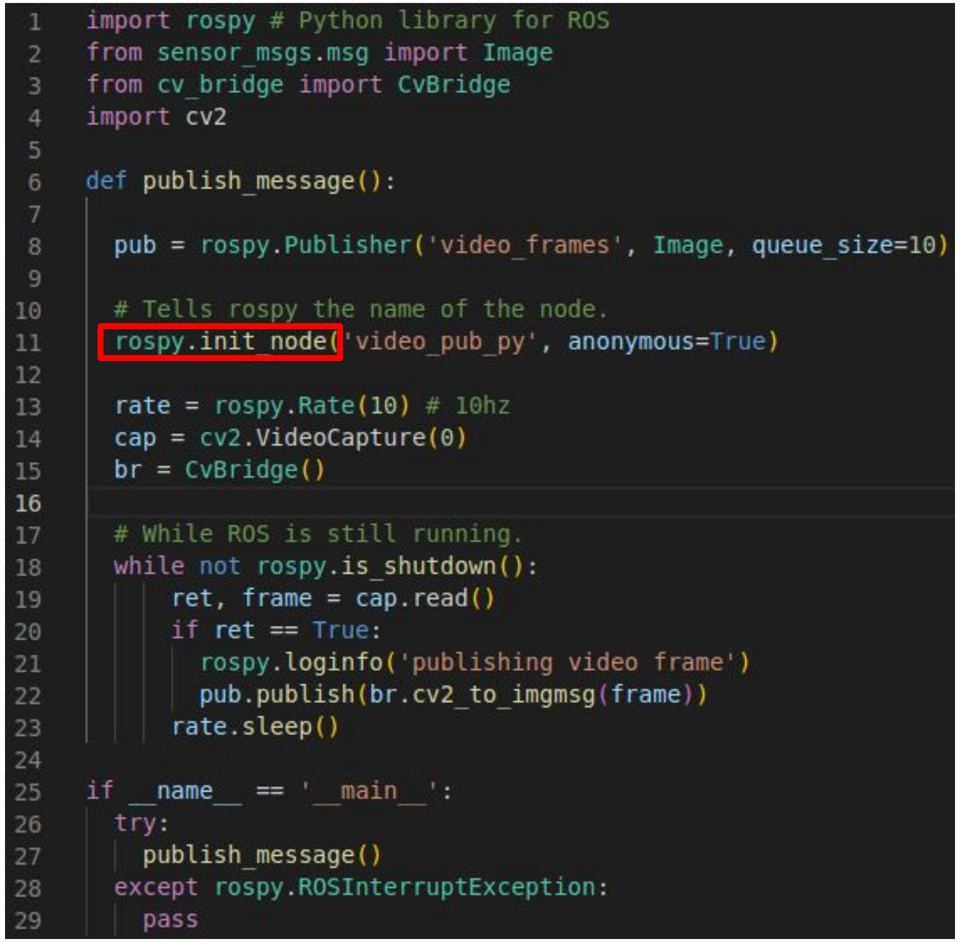


```
adnana@ece-p206-lnx02:~$ cd ros2_ws
adnana@ece-p206-lnx02:~/ros2_ws$ colcon build
Starting >>> face_detector
Starting >>> my_face_detection
Starting >>> my_robot_camera
Finished <<< face_detector [0.89s]

Finished <<< my_face_detection [0.89s]
Finished <<< my_robot_camera [1.11s]

Summary: 3 packages finished [1.25s]
```

Writing a Node

ROS 1

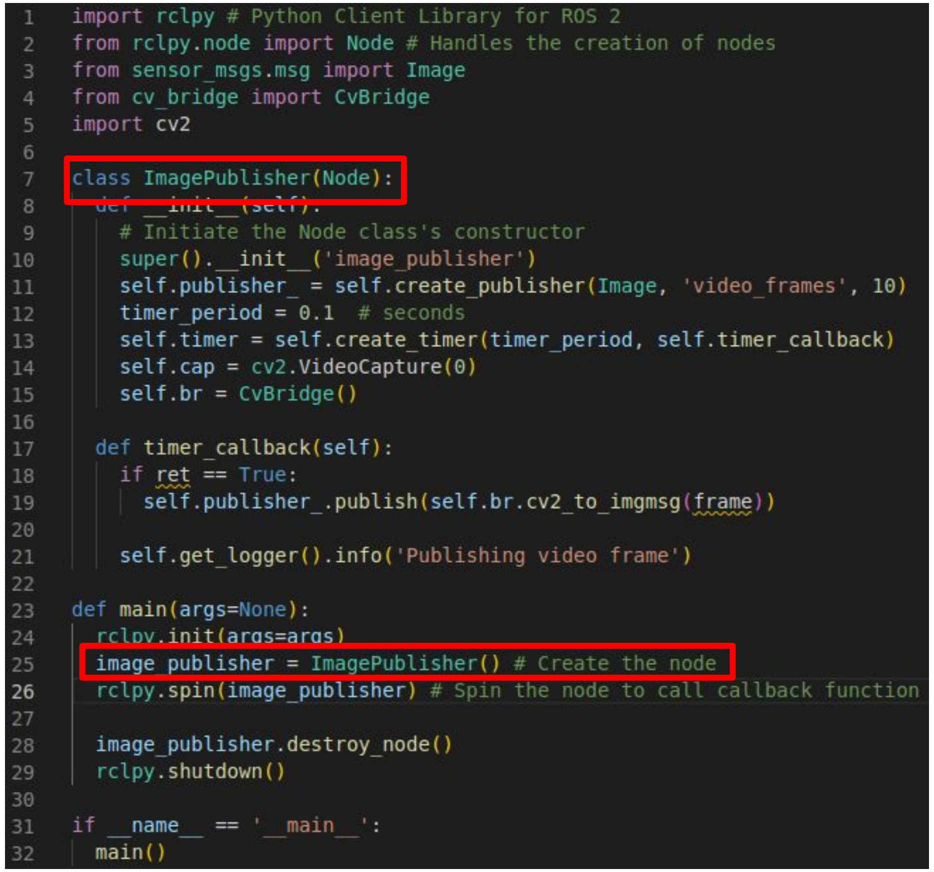
- Class is not necessary.



```
import rospy # Python library for ROS
from sensor_msgs.msg import Image
from cv_bridge import CvBridge
import cv2

def publish_message():

  pub = rospy.Publisher('video_frames', Image, queue_size=10)

  # Tells rospy the name of the node.
  rospy.init_node('video_pub_py', anonymous=True)

  rate = rospy.Rate(10) # 10hz
  cap = cv2.VideoCapture(0)
  br = CvBridge()

  # While ROS is still running.
  while not rospy.is_shutdown():
      ret, frame = cap.read()
      if ret == True:
        rospy.loginfo('publishing video frame')
        pub.publish(br.cv2_to_imgmsg(frame))
      rate.sleep()

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

ROS 2

- Create a class, all ROS2 functionalities will be in this class.

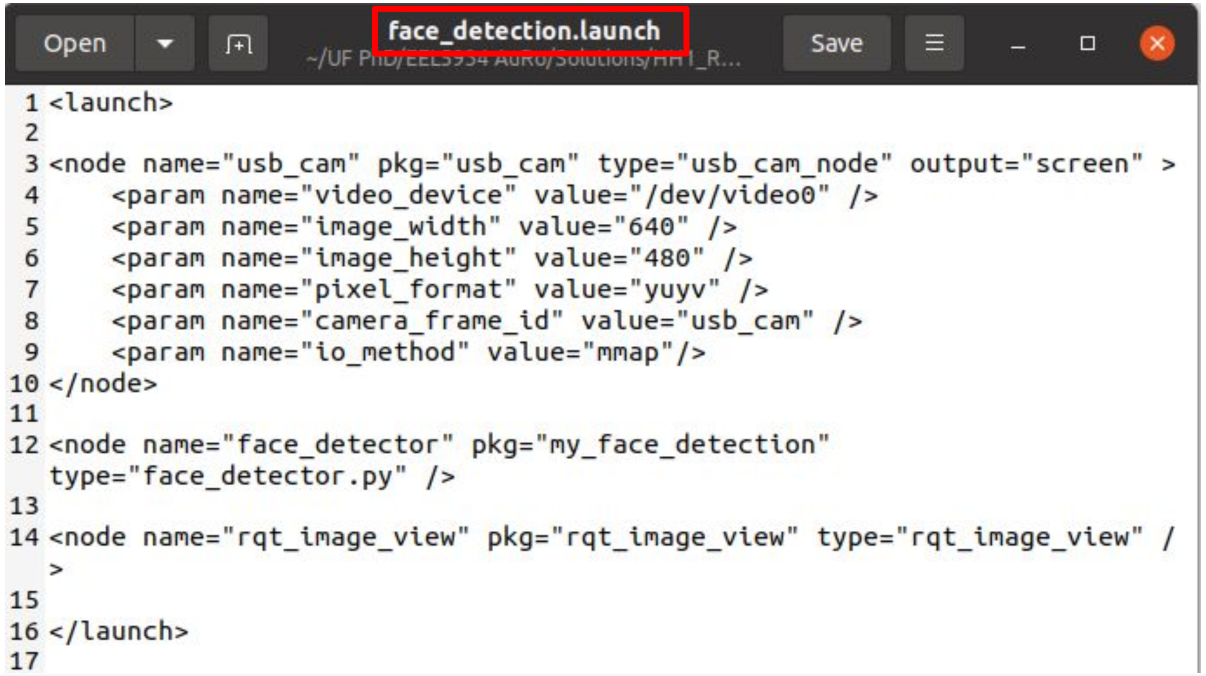


```
import rclpy # Python Client Library for ROS 2
from rclpy.node import Node # Handles the creation of nodes
from sensor_msgs.msg import Image
from cv_bridge import CvBridge
import cv2

class ImagePublisher(Node):
  def __init__(self):
    # Initiate the Node class's constructor
    super().__init__('image_publisher')
    self.publisher_ = self.create_publisher(Image, 'video_frames', 10)
    timer_period = 0.1  # seconds
    self.timer = self.create_timer(timer_period, self.timer_callback)
    self.cap = cv2.VideoCapture(0)
    self.br = CvBridge()

  def timer_callback(self):
    if ret == True:
      self.publisher_.publish(self.br.cv2_to_imgmsg(frame))

    self.get_logger().info('Publishing video frame')

def main(args=None):
  rclpy.init(args=args)
  image_publisher = ImagePublisher() # Create the node
  rclpy.spin(image_publisher) # Spin the node to call callback function

  image_publisher.destroy_node()
  rclpy.shutdown()

if __name__ == '__main__':
  main()
```

Launch File

ROS 1

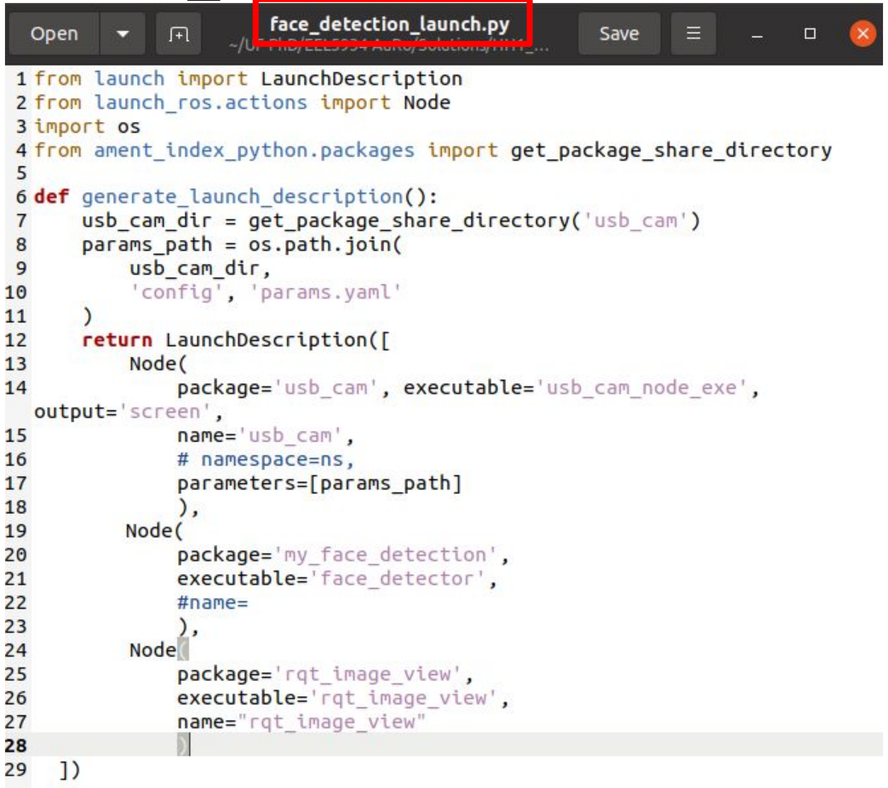
- .launch XML file
- `roslaunch my_face_detection face_detection.launch`



```
<launch>

<node name="usb_cam" pkg="usb_cam" type="usb_cam_node" output="screen" >
    <param name="video_device" value="/dev/video0" />
    <param name="image_width" value="640" />
    <param name="image_height" value="480" />
    <param name="pixel_format" value="yuyv" />
    <param name="camera_frame_id" value="usb_cam" />
    <param name="io_method" value="mmap"/>
</node>

<node name="face_detector" pkg="my_face_detection"
type="face_detector.py" />

<node name="rqt_image_view" pkg="rqt_image_view" type="rqt_image_view" /
>

</launch>

```

ROS 2

- .py Python script
- `ros2 launch my_face_detection face_detection_launch.py`

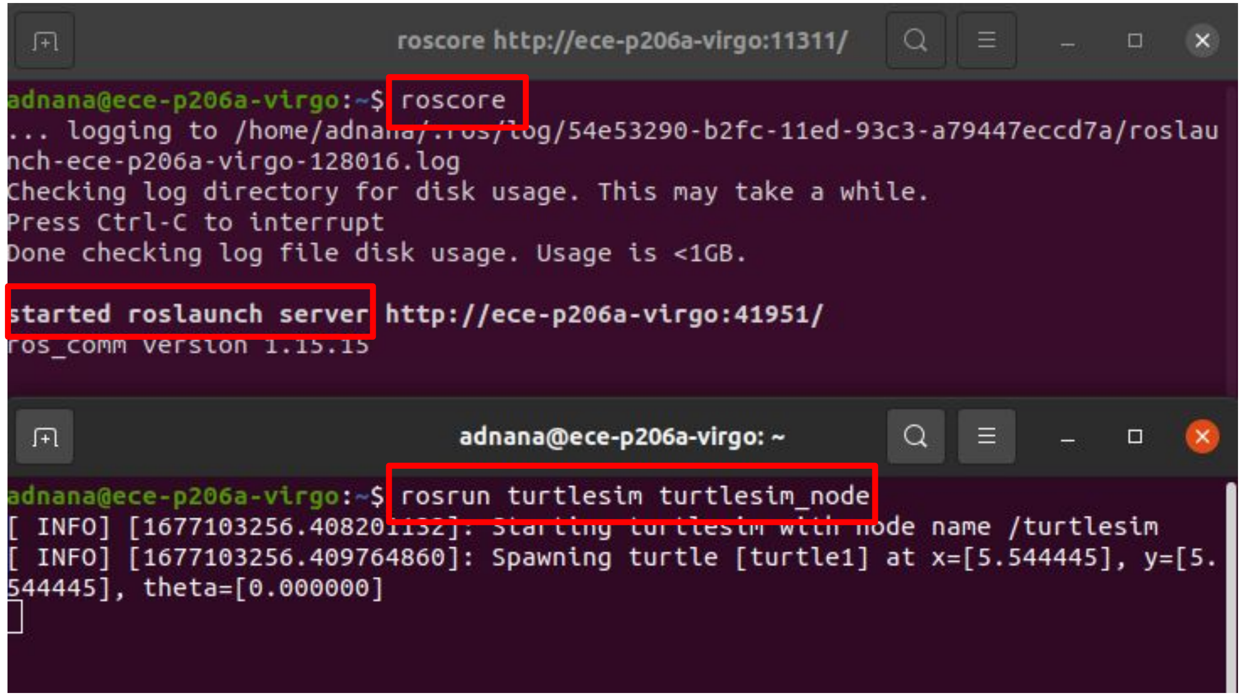


```
from launch import LaunchDescription
from launch_ros.actions import Node
import os
from ament_index_python.packages import get_package_share_directory

def generate_launch_description():
    usb_cam_dir = get_package_share_directory('usb_cam')
    params_path = os.path.join(
        usb_cam_dir,
        'config', 'params.yaml'
    )
    return LaunchDescription([
        Node(
            package='usb_cam', executable='usb_cam_node_exe',
output='screen',
            name='usb_cam',
            # namespace=ns,
            parameters=[params_path]
            ),
        Node(
            package='my_face_detection',
            executable='face_detector',
            #name=
            ),
        Node(
            package='rqt_image_view',
            executable='rqt_image_view',
            name="rqt_image_view"
            )
  ])
```

ROS Master

ROS 1

- Start a ROS master before running a node.
- `roscore` then `rosrun`

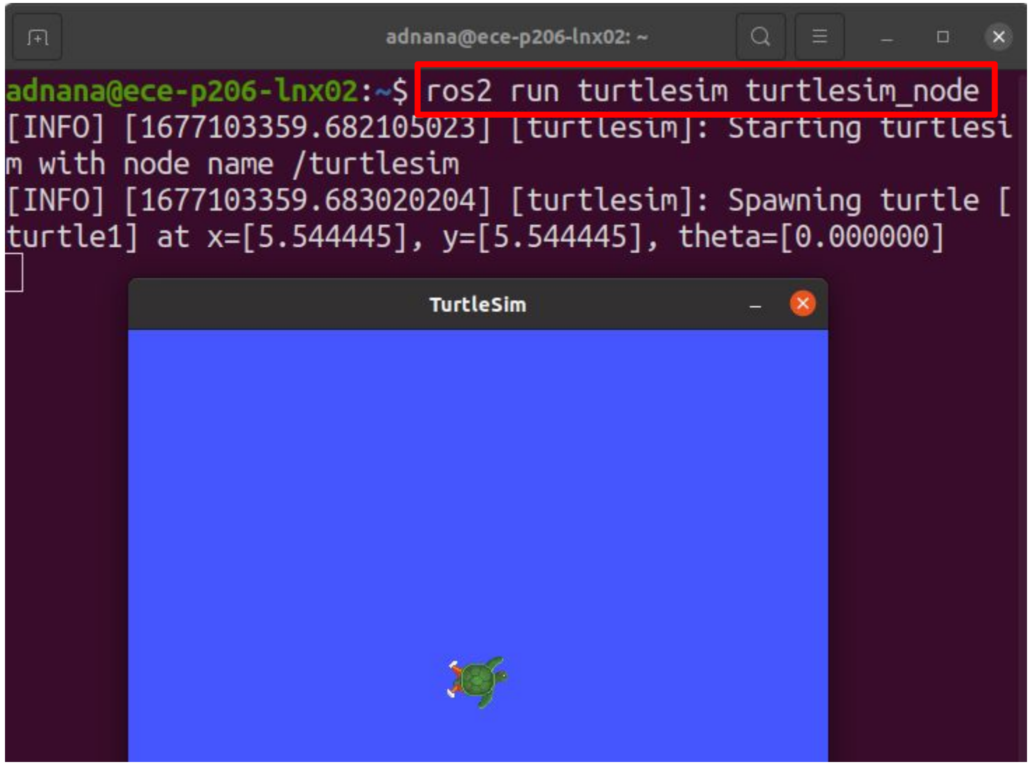

roscore http://ece-p206a-virgo:11311/
adnana@ece-p206a-virgo: ~

```
adnana@ece-p206a-virgo:~$ roscore
... logging to /home/adnana/.ros/log/54e53290-b2fc-11ed-93c3-a79447eccd7a/roslau
nch-ece-p206a-virgo-128016.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://ece-p206a-virgo:41951/
ros_comm version 1.15.15
```

```
adnana@ece-p206a-virgo:~$ rosrun turtlesim turtlesim_node
[ INFO] [1677103256.408201132]: Starting turtlesim with node name /turtlesim
[ INFO] [1677103256.409764860]: Spawning turtle [turtle1] at x=[5.544445], y=[5.
544445], theta=[0.000000]
```

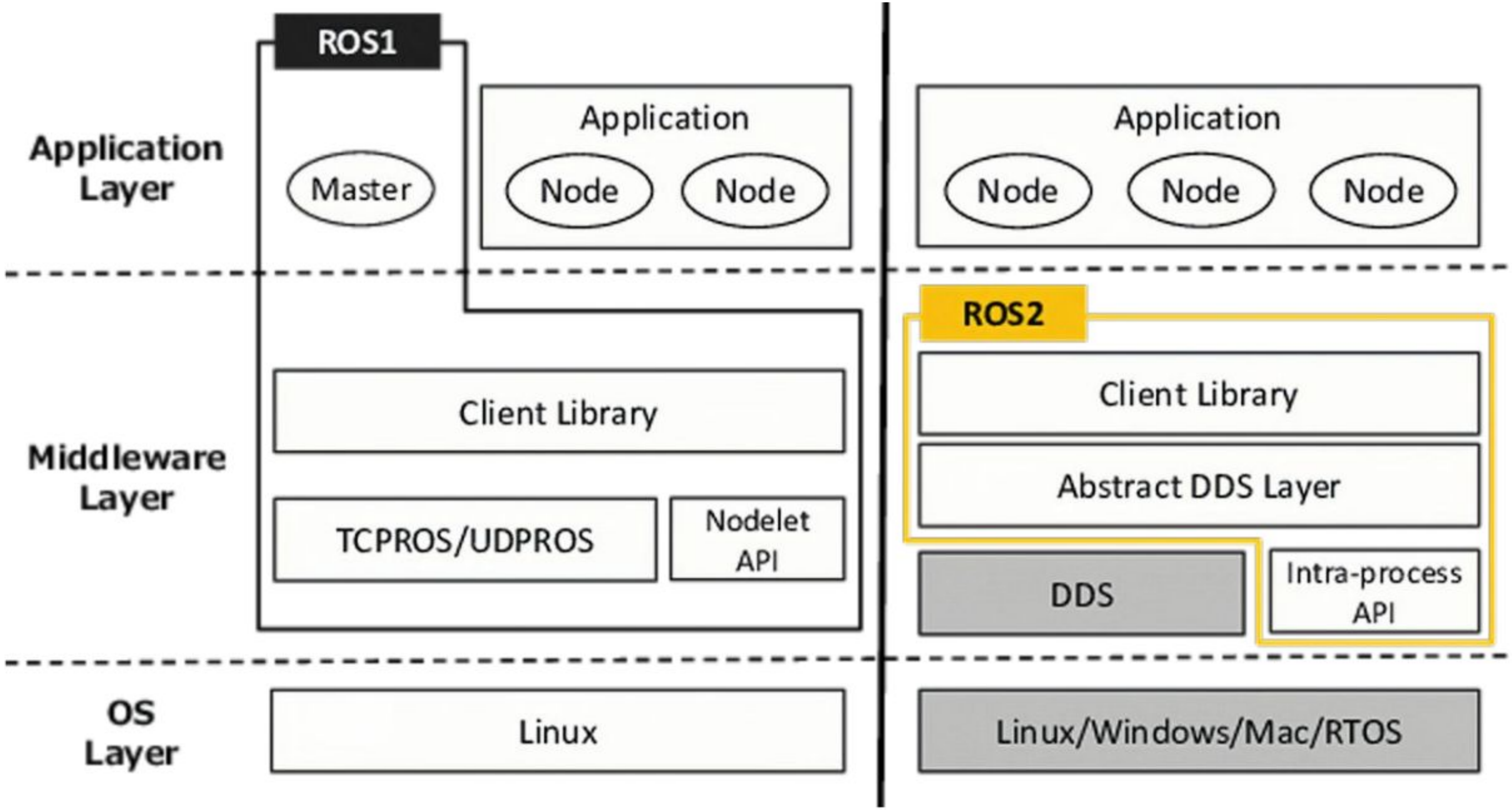
ROS 2

- No more ROS master! Each node has the capacity to discover other nodes.
- `ros2 run`



```
adnana@ece-p206-lnx02:~$ ros2 run turtlesim turtlesim_node
[INFO] [1677103359.682105023] [turtlesim]: Starting turtlesi
m with node name /turtlesim
[INFO] [1677103359.683020204] [turtlesim]: Spawning turtle [
turtle1] at x=[5.544445], y=[5.544445], theta=[0.000000]
```

ROS Master


Application Layer
Middleware Layer
OS Layer
ROS1
Master
Application
Node
Node
Client Library
TCPROS/UDPROS
Nodelet API
Linux
Application
Node
Node
Node
ROS2
Client Library
Abstract DDS Layer
DDS
Intra-process API
Linux/Windows/Mac/RTOS

Parameters

ROS 1

- Parameters are handled by the parameter server, which is itself handled by the ROS master.

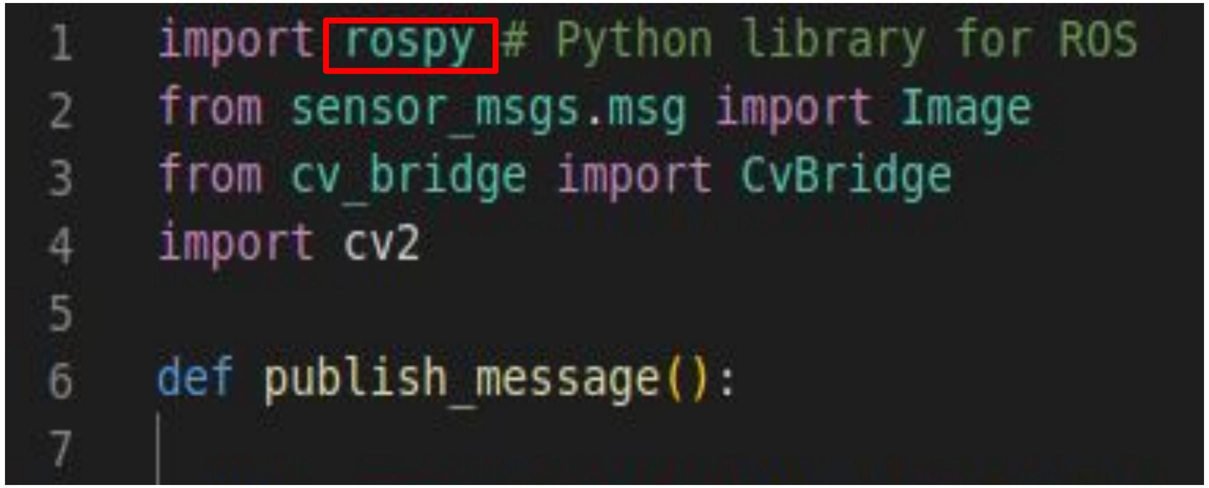
ROS 2

- No master, so no global parameter anymore.
- Each parameter is specific to a node.

API

ROS 1

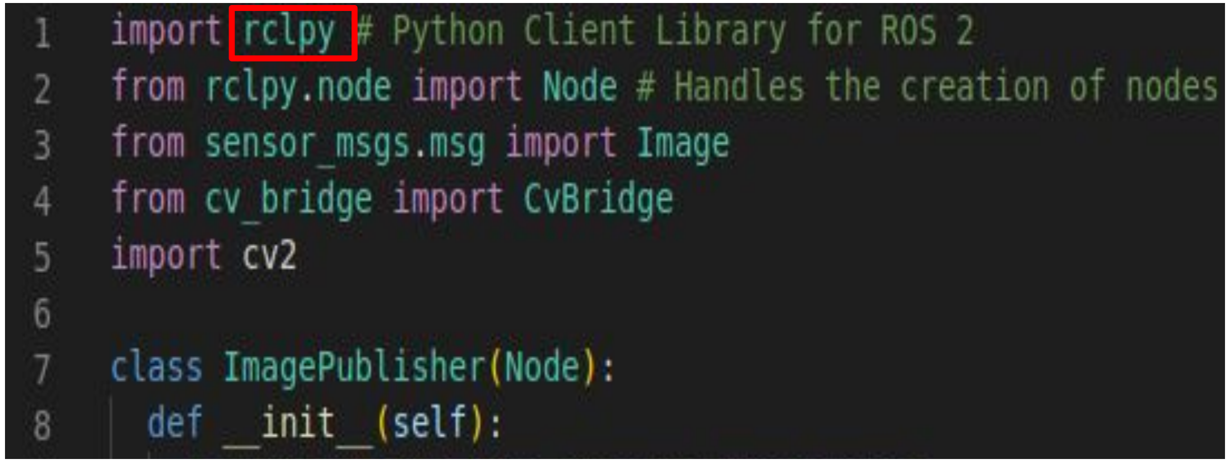
- roscpp and rospy
- Both libraries are completely independent and built from scratch.
- Some features are developed for one, and not the other.



```
import rospy # Python library for ROS
from sensor_msgs.msg import Image
from cv_bridge import CvBridge
import cv2

def publish_message():

```

ROS 2

- One base library- rcl, implemented in C. Foundation for all ROS2 core features.
- We use another client library built on top of rcl- rclcpp or rclpy.

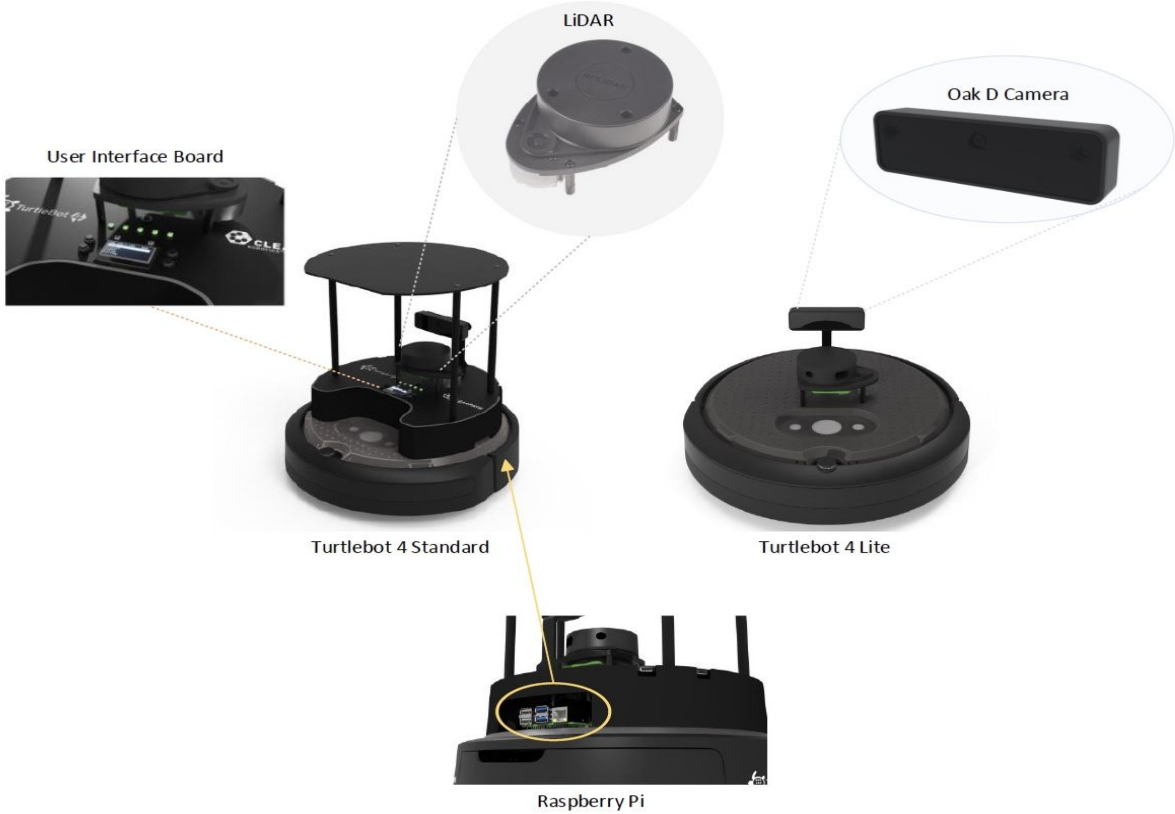


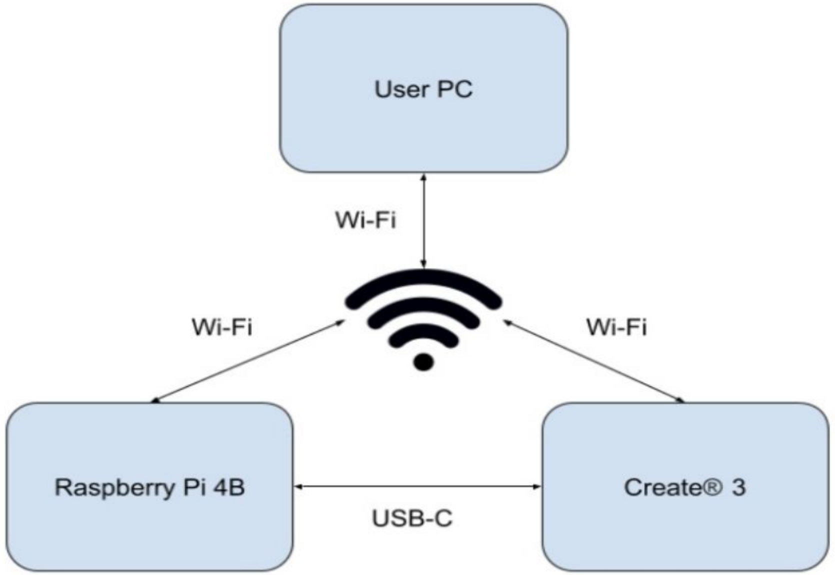
```
import rclpy # Python Client Library for ROS 2
from rclpy.node import Node # Handles the creation of nodes
from sensor_msgs.msg import Image
from cv_bridge import CvBridge
import cv2

class ImagePublisher(Node):
    def __init__(self):
```

Let's do some ROS2 with Turtlebot!

Getting Familiar...

- Two versions- Turtlebot4 **Standard** and Turtlebot4 **Lite**
- Two computers- **Raspberry Pi** and **Create 3**
- Two sensors- **LiDar** and the Oak D **camera** (Oak D pro / Oak-D-lite)





Prepare your PC to get connected (Just Once)

- Download an xml file using this command:

```
wget https://raw.githubusercontent.com/turtlebot/turtlebot4_setup/galactic/conf/cyclonedds_pc.xml
```

- Find your WiFi network interface name with this command:

`ip link` (The name would be something like- wlp0s20f3)

- Open the xml file with `gedit` and add the following line below `<DontRoute>true</DontRoute>` this line:

```
<NetworkInterfaceAddress>"your wifi interface name"</NetworkInterfaceAddress>
```

- Move and export it with the following commands:

```
sudo mv cyclonedds pc.xml /etc/
export CYCLONEDDS_URI=/etc/cyclonedds_pc.xml
source ~/.bashrc
```

- Install necessary packages on your pc:

```
sudo apt install ros-galactic-teleop-twist-keyboard
sudo apt-get install ros-galactic-turtlebot4-viz
sudo apt-get install ros-galactic-turtlebot4-navigation
```

Configure Raspberry Pi WiFi (Just Once)

- Place the bot on the charging doc. It will turn on.
- Connect your pc to the WiFi called `Turtlebot4`. Password is `Turtlebot4` by default.
- Once connected, ssh into Raspberry Pi from your pc by running this command:

```
ssh ubuntu@10.42.0.1
```

- In /usr/local/bin folder of Raspberry Pi there is a script called `wifi.sh`. Edit it to connect the Raspberry Pi to your home WiFi:

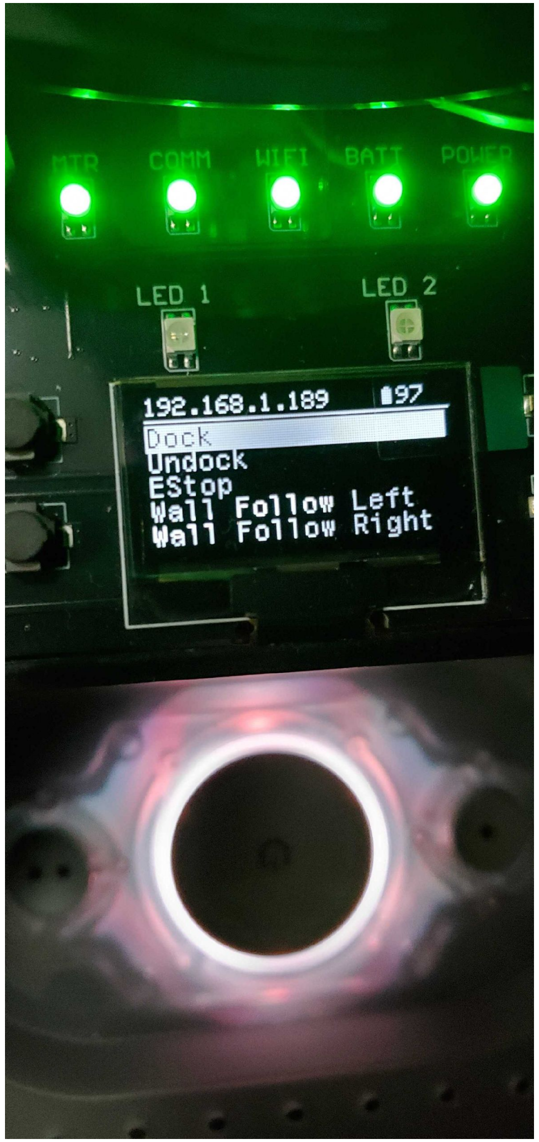
```
sudo wifi.sh -s '<your WIFI SSID>' -p '<your WIFI_PASSWORD>' -r <REGULATORY_DOMAIN> && sudo reboot
```

The Regulatory Domain is based on the country you live in (For USA: `US`).

- Once it is rebooted, run the following command on your pc to see Raspberry Pi's IP:

```
ros2 topic echo /ip
```

You should see the ip printed on a regular interval.

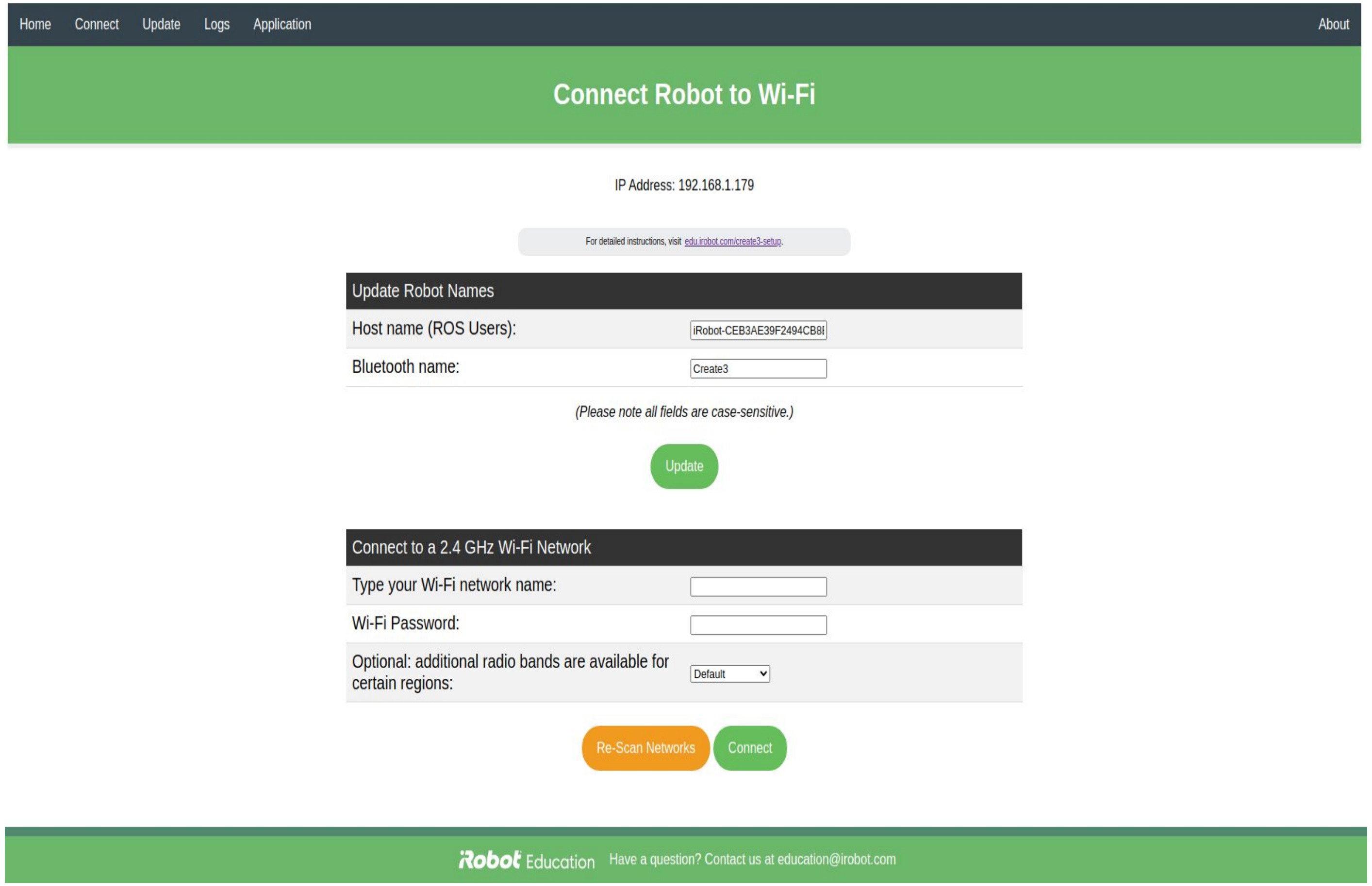


To connect to a new network: ssh into Raspberry Pi while connected on current network and use command:

```
sudo wifi.sh -s '<new SSID>' -p '<new PASSWORD>' -a && sudo reboot
```

Configure Connect3 WiFi (Just Once)

- Press the two buttons surrounding the home button at the same time, the light ring will flash, wait until it turns blue.
- Connect your pc to the WiFi network called `Create-XXXX`.
- In a browser go to `192.168.10.1`
- Go to the Connect tab, enter your WiFi SSID and password, and then click ‘Connect'. Once connected, the Turtlebot will play a chime.
- Check to make sure the create 3 is publishing topics by running `ros2 topic list` on your pc.



Connect Robot to Wi-Fi

IP Address: 192.168.1.179

For detailed instructions, visit edu.irobot.com/create3-setup.

Update Robot Names

Host name (ROS Users): iRobot-CEB3AE39F2494CB8

Bluetooth name: Create3

(Please note all fields are case-sensitive.)

Connect to a 2.4 GHz Wi-Fi Network

Type your Wi-Fi network name:

Wi-Fi Password:

Optional: additional radio bands are available for certain regions: Default

iRobot Education Have a question? Contact us at education@irobot.com

Follow this link for more info.

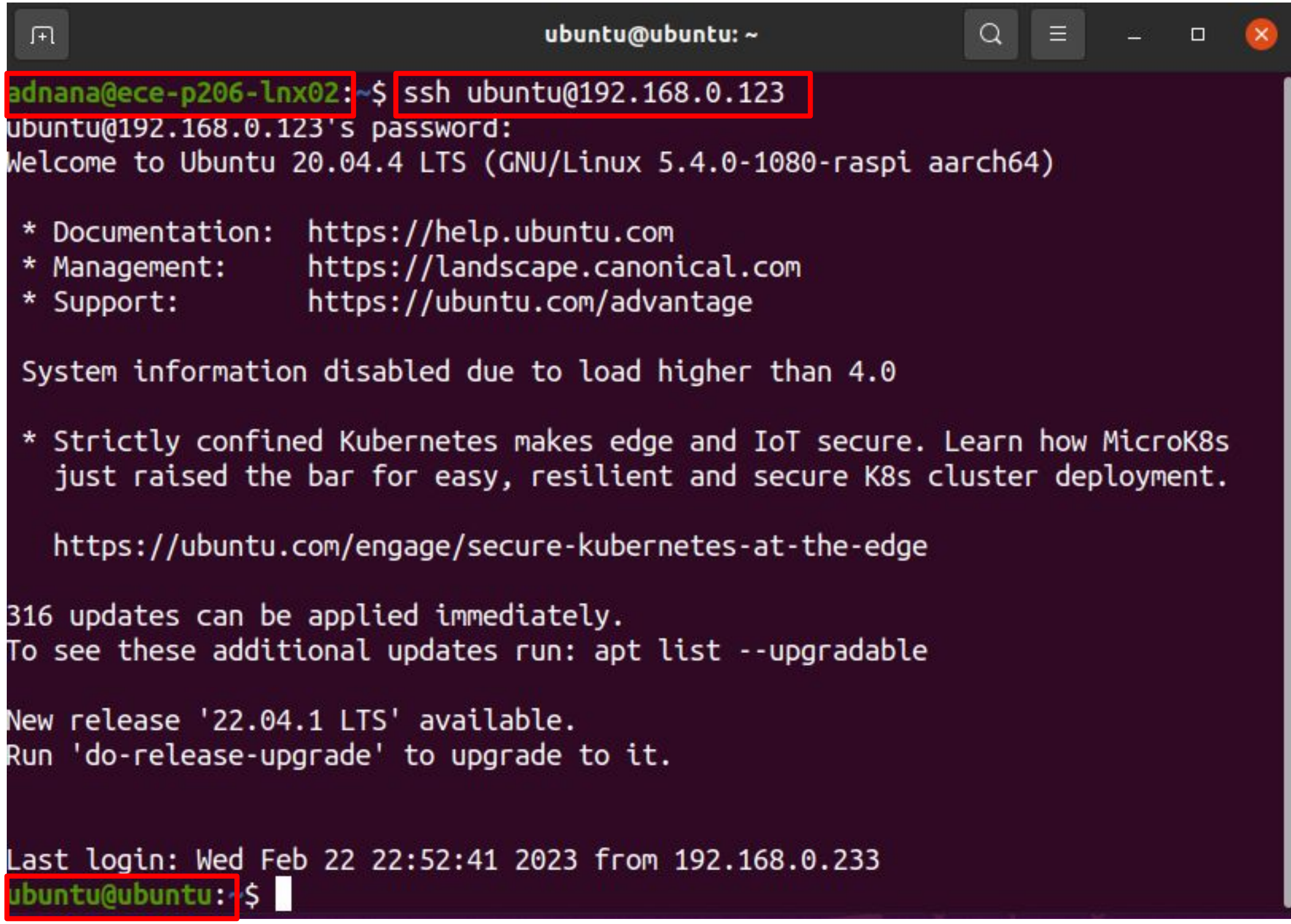
Let's Get Connected… (Everytime you start the bot)

- Place the bot on the charging doc. It will turn on.
- There will be two chimes-
  - The first indicates that Raspberry Pi is ready and
  - The second indicates that Create3 is ready.
- To find the ip address of Raspberry Pi, run on your pc:

```
ros2 topic echo /ip
```

- Connect to Raspberry Pi via ssh, run on your pc:

```
ssh ubuntu@192.168.x.xxx
```



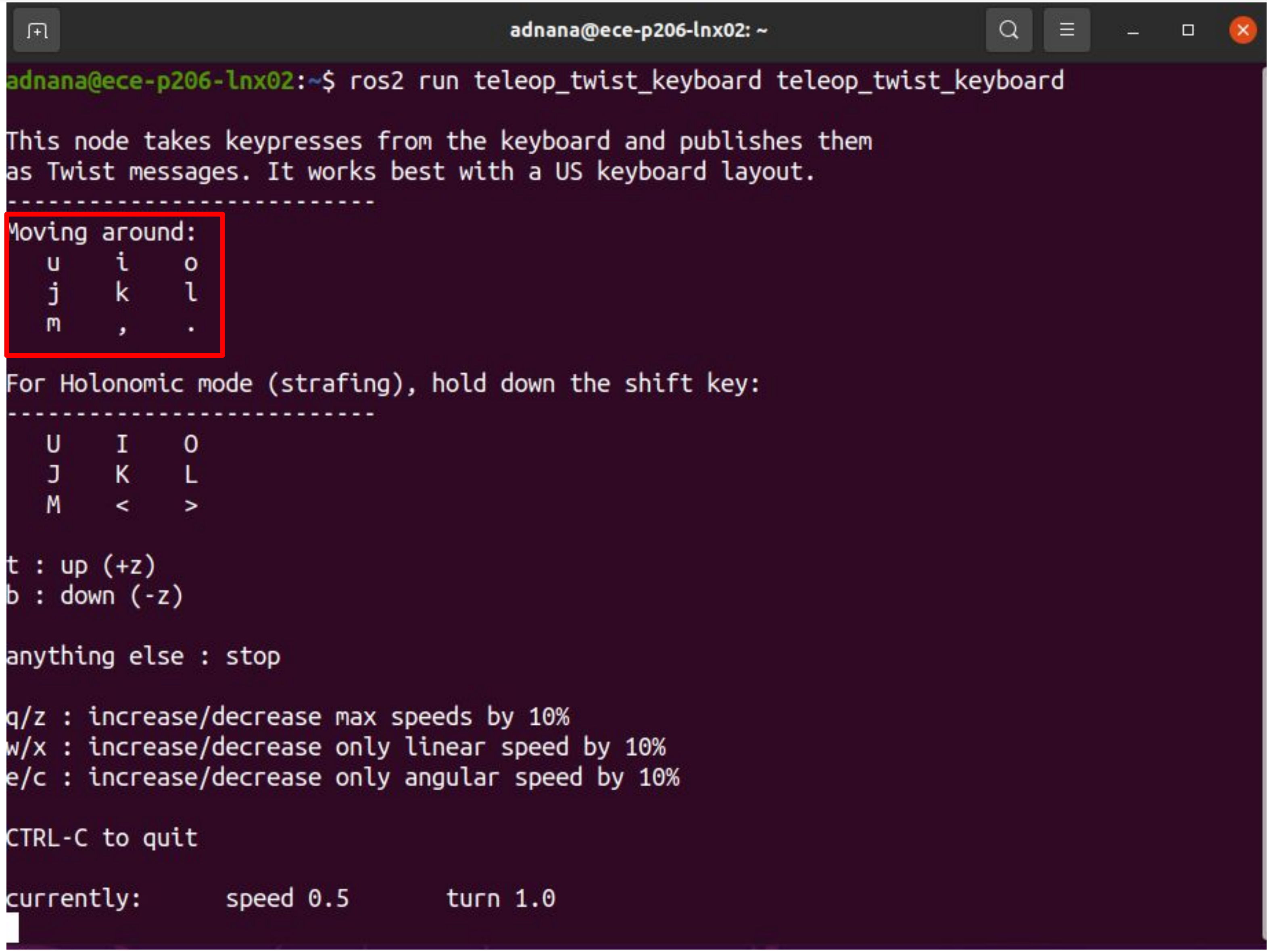
```
adnana@ece-p206-lnx02:~$ ssh ubuntu@192.168.0.123
ubuntu@192.168.0.123's password:
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.4.0-1080-raspi aarch64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 System information disabled due to load higher than 4.0

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

316 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

New release '22.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Wed Feb 22 22:52:41 2023 from 192.168.0.233
ubuntu@ubuntu:~$ 
```

Time to show some move!

- In a new terminal on your pc, run:

```
ros2 run
teleop_twist_keyboard
teleop_twist_keyboard
```

- You should be able to move your bot with keyboard.



```
adnana@ece-p206-lnx02:~$ ros2 run teleop_twist_keyboard teleop_twist_keyboard

This node takes keypresses from the keyboard and publishes them
as Twist messages. It works best with a US keyboard layout.
---------------------------
Moving around:
   u    i    o
   j    k    l
   m    ,    .

For Holonomic mode (strafing), hold down the shift key:
---------------------------
   U    I    O
   J    K    L
   M    <    >

t : up (+z)
b : down (-z)

anything else : stop

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%

CTRL-C to quit

currently:      speed 0.5       turn 1.0
```

Mapping with SLAM and RViz

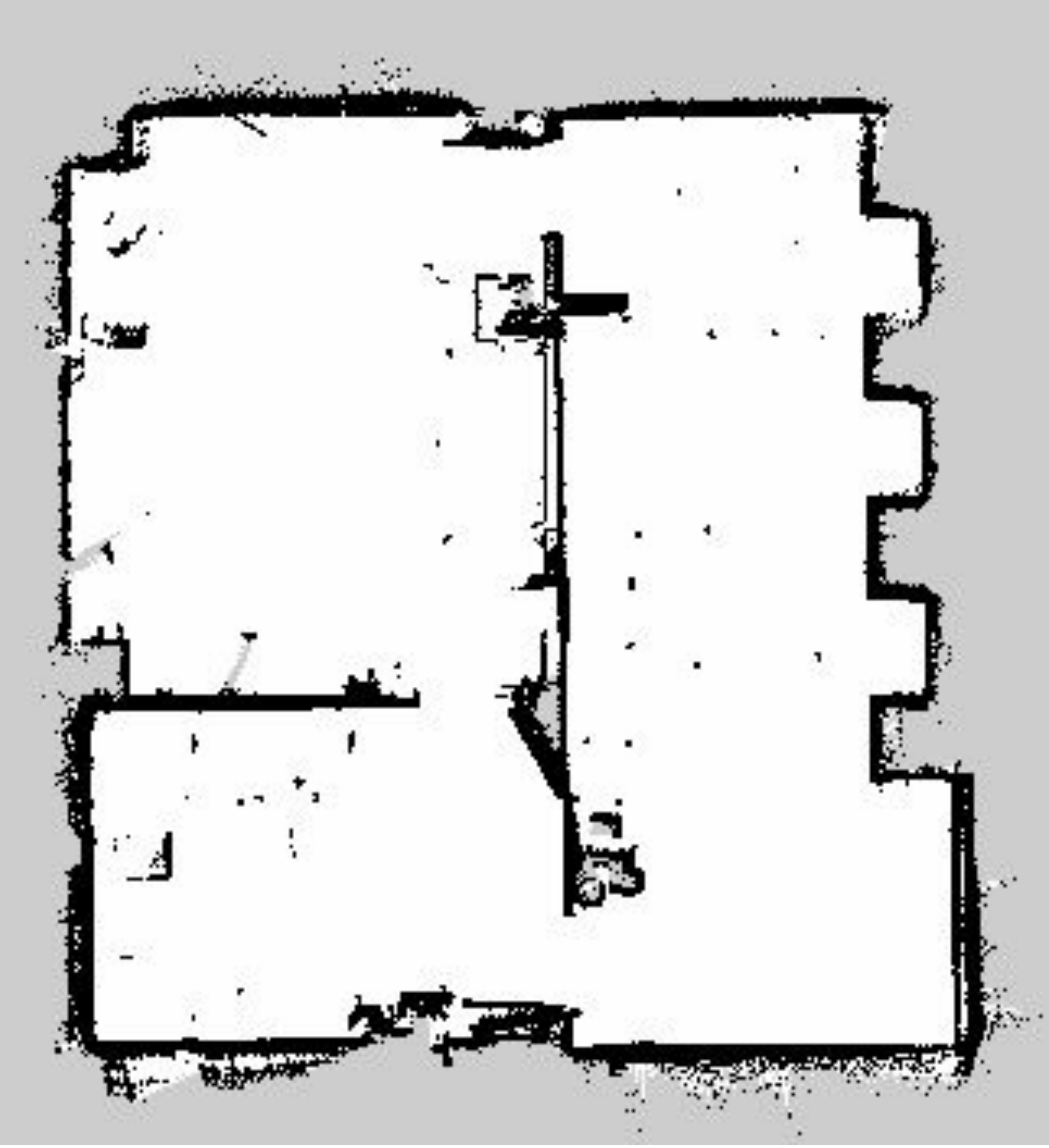
- In a new terminal on your pc run:

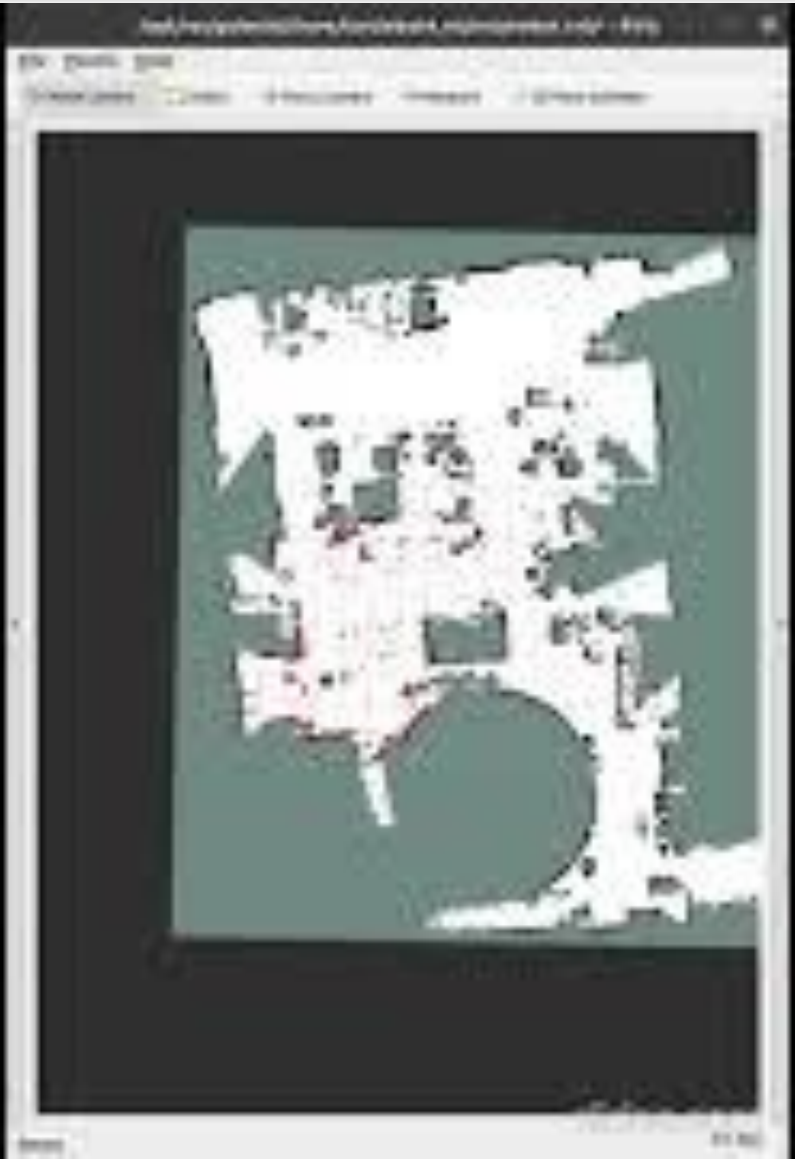
```
ros2 launch turtlebot4_navigation
slam_sync.launch.py
```

- In another terminal run:

```
ros2 launch turtlebot4_viz
view_robot.launch.py
```

- Move your bot and it will create a map of the surrounding.





Thank you!

Questions?
Comments?