EXPERIMENT 10
Basics of ROS

Aim:

To install the ROS Noetic version and learn to create a workspace in ROS and to run simple program.

Software/ Package Used:

- Ubuntu 18.04
- ROS - Noetic

Programs:

1. Write a ROS program to configure a node and send a message and configure two different users to receive the same message.

PUBLISHER CODE:

```
#!/usr/bin/env python3
import rospy
from std_msgs.msg import String
def talker():
        pub = rospy.Publisher('chatter', String, queue_size=10)
        rospy.init_node('talker', anonymous=True)
        rate = rospy.Rate(10) # 10hz
        while not rospy.is_shutdown():
        hello_str = "hello world %s" % rospy.get_time()
        rospy.loginfo(hello_str)
        pub.publish(hello_str)
        rate.sleep()
if __name__ == '__main__':
        try:
        talker()
        except rospy.ROSInterruptException:
        pass
```

SUBSCRIBER 1 CODE:

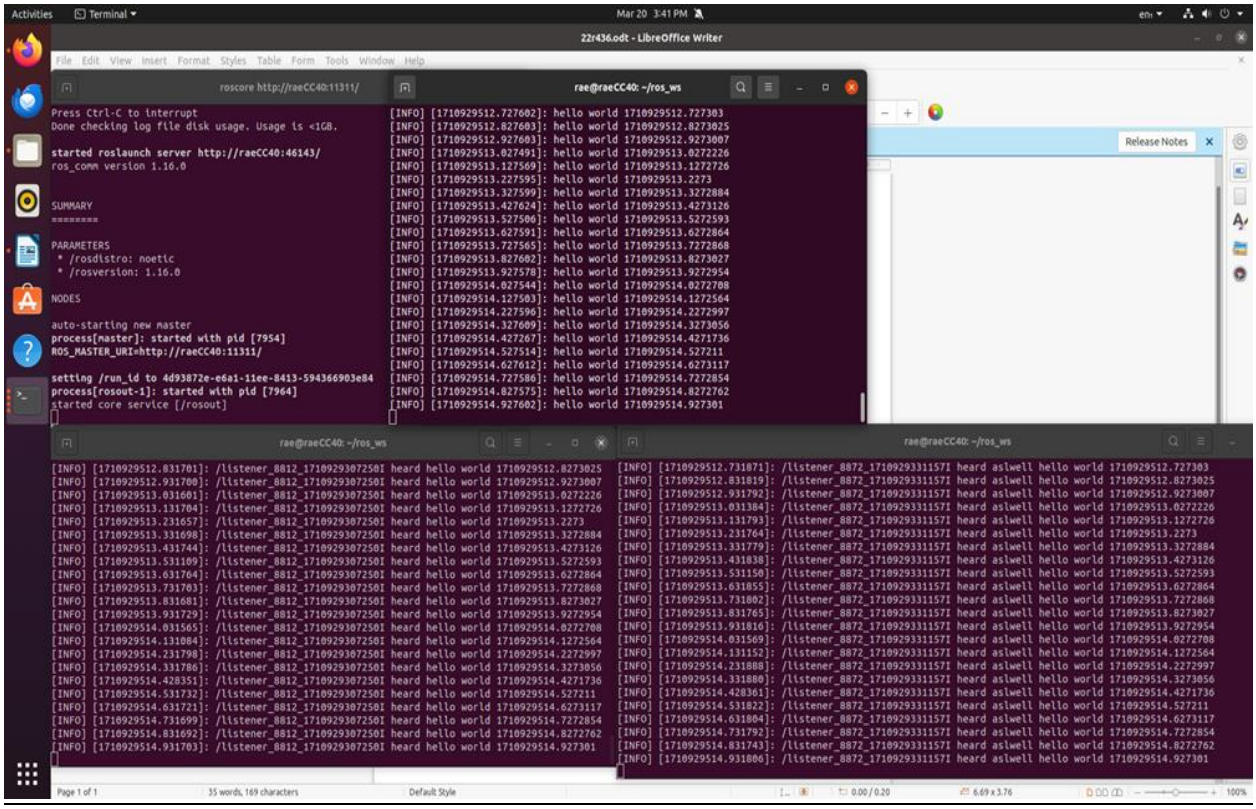
```
#!/usr/bin/env python3
import rospy
from std_msgs.msg import String
def callback(data):
        rospy.loginfo(rospy.get_caller_id() + 'I heard %s', data.data)
def listener():
```

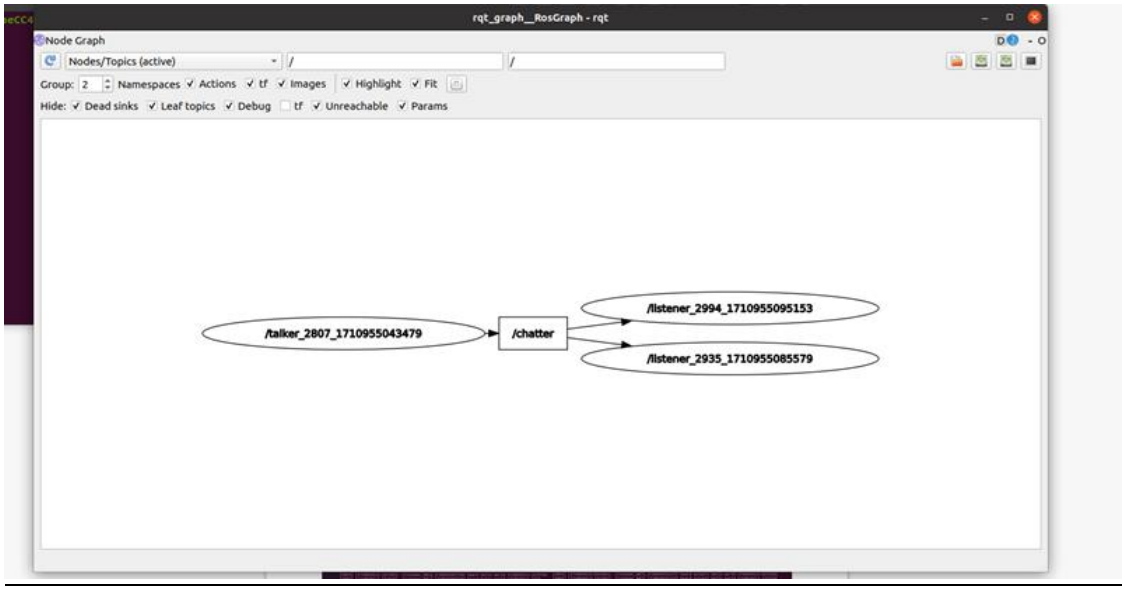
```
        rospy.init_node('listener', anonymous=True)
        rospy.Subscriber('chatter', String, callback)
        # spin() simply keeps python from exiting until this node is stopped
        rospy.spin()
if __name__ == '__main__':
        listener()
```

SUBSCRIBER 2 CODE:

```
#!/usr/bin/env python3
import rospy
from std_msgs.msg import String
def callback(data):
        rospy.loginfo(rospy.get_caller_id() + 'I heard as well %s', data.data)
def listener():
        rospy.init_node('listener', anonymous=True)
        rospy.Subscriber('chatter', String, callback)
        rospy.spin()
if __name__ == '__main__':
        listener()
```

OUTPUT:



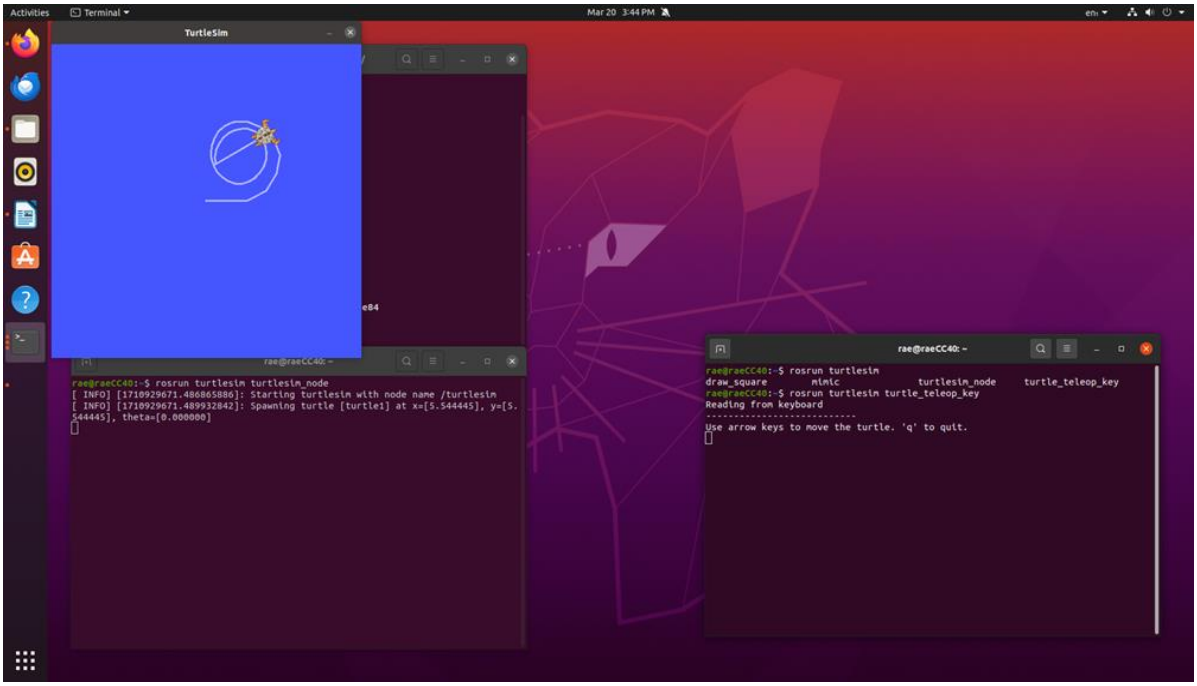


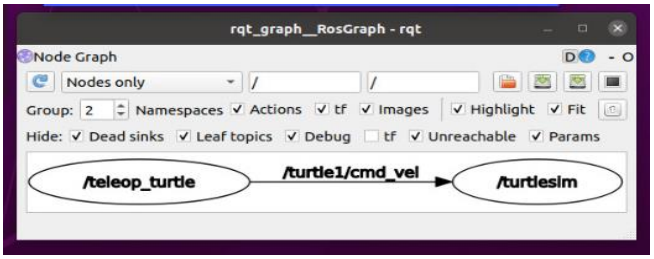
2. Run turtlesim

COMMANDS:

- rosrun turtlesim turtlesim_node
- rosrun turtlesim turtle_teleop_key
- rqt_graph

OUTPUT:



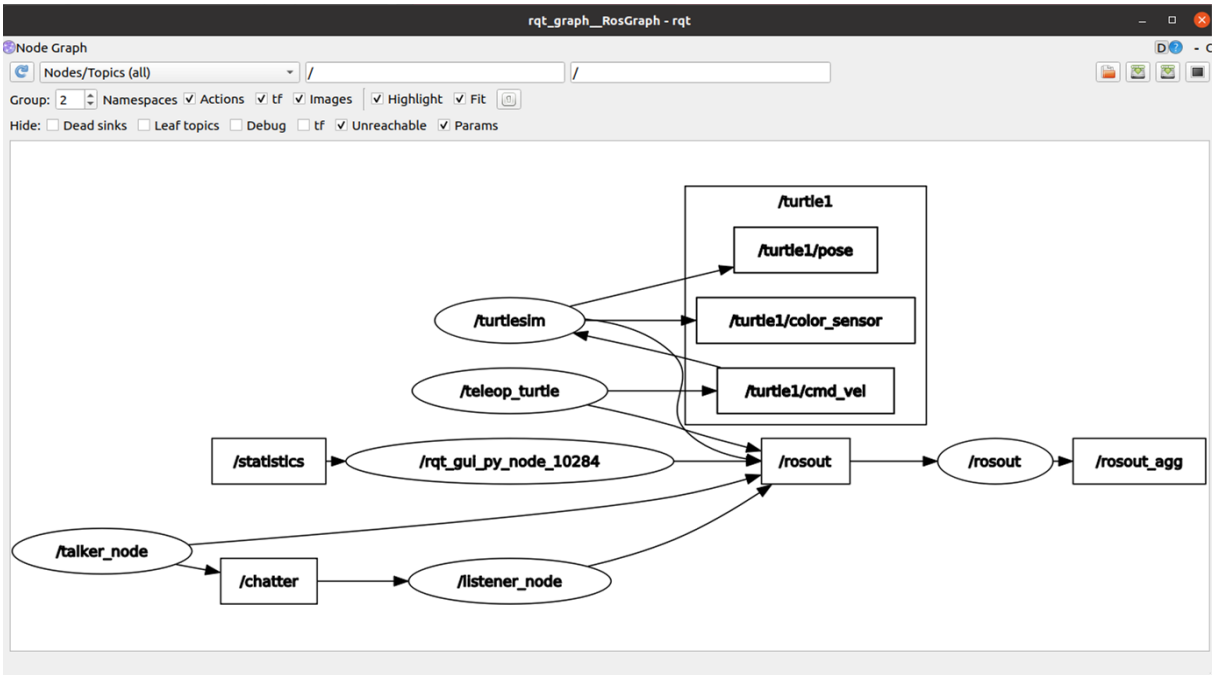


3. RQT Graph.

COMMANDS:

- rqt_graph

OUTPUT:



4. Bridge ROS with openCV. Read an image into ROS and rotate the image.

CODE:

```
#!/usr/bin/env python3
import rospy # Python library for ROS
from sensor_msgs.msg import Image # Image is the message type
from cv_bridge import CvBridge # Package to convert between ROS and OpenCV
Images
import cv2 # OpenCV library
def callback(data):
  br = CvBridge()
    rospy.loginfo("receiving video frame")
    current_frame = br.imgmsg_to_cv2(data)
  current_frame=cv2.circle(current_frame,(60,60),10,(0,255,255),-1)
  current_frame=cv2.rotate(current_frame,cv2.ROTATE_90_CLOCKWISE)
  cv2.imshow("camera", current_frame)
  cv2.waitKey(0)
def receive_message():
  rospy.init_node('video_sub_py', anonymous=True)
```

```
 rospy.Subscriber('video_frames', Image, callback)
 rospy.spin()
 cv2.destroyAllWindows()
if _name_ == '_main_':
 receive_message()
```

OUTPUT:

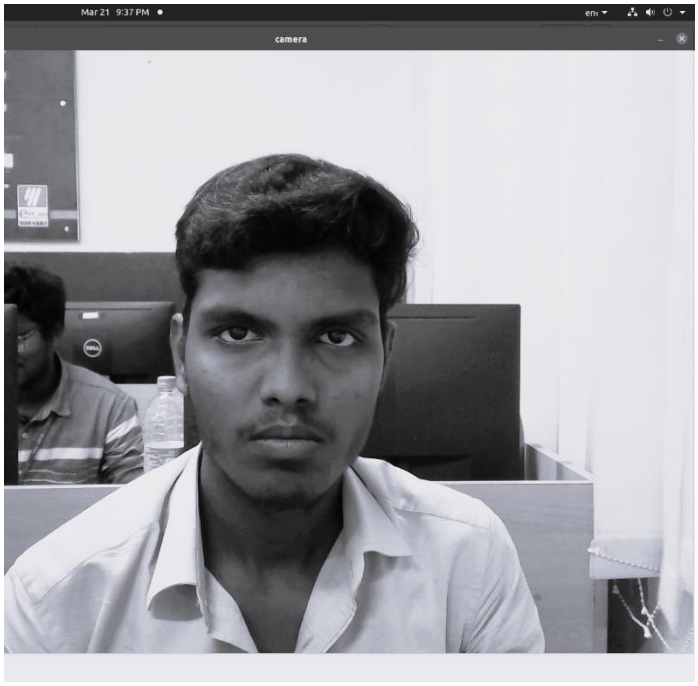
5. Read an image into ROS and perform color conversions on an image.

CODE:

```
#!/usr/bin/env python3
import rospy # Python library for ROS
from sensor_msgs.msg import Image # Image is the message type
from cv_bridge import CvBridge # Package to convert between ROS and OpenCV
Images
import cv2 # OpenCV library
 def publish_message():
  pub = rospy.Publisher('video_frames', Image, queue_size=10)
  rospy.init_node('video_pub_py', anonymous=True)
  rate = rospy.Rate(10) # 10hz
  cap = cv2.imread('/home/rae/Downloads/test.png',0)
  br = CvBridge()
   while not rospy.is_shutdown():
```

```
            if True:
            rospy.loginfo('publishing video frame')
            pub.publish(br.cv2_to_imgmsg(cap))
            rate.sleep()
if _name_ == '_main_':
  try:
            publish_message()
  except rospy.ROSInterruptException:
            pass
```

OUTPUT:



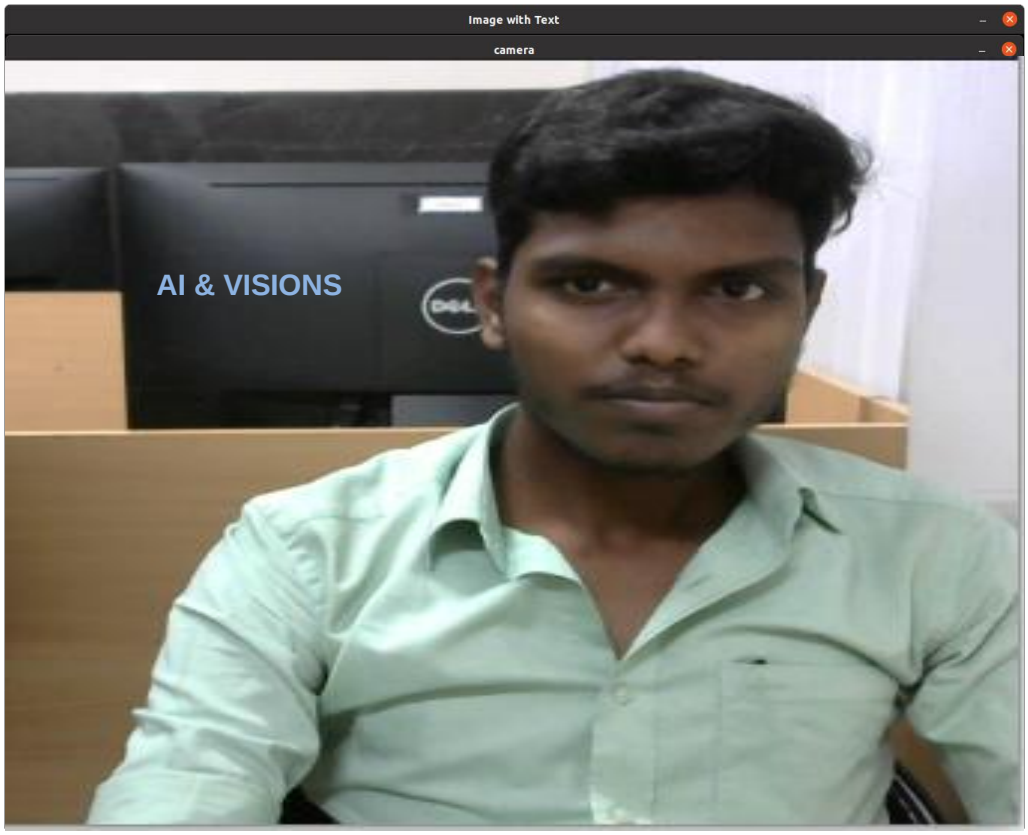
6. Write AI and vison on an image.

CODE:

```
#!/usr/bin/env python3
import rospy # Python library for ROS
from sensor_msgs.msg import Image # Image is the message type
from cv_bridge import CvBridge # Package to convert between ROS and OpenCV
Images
import cv2 # OpenCV library
 def publish_message():
  pub = rospy.Publisher('video_frames', Image, queue_size=10)
  rospy.init_node('video_pub_py', anonymous=True)
  rate = rospy.Rate(10) # 10hz
  cap = cv2.imread('/home/rae/Downloads/black screen.png')
current_frame=cv2.putText(current_frame,"AI &
VISION",(50,50),cv2.FONT_HERSHEY_SIMPLEX,1,(255,0,255),2,cv2.LINE_AA)
```

```
 br = CvBridge()
  while not rospy.is_shutdown():
        if True:
        rospy.loginfo('publishing video frame')
        pub.publish(br.cv2_to_imgmsg(cap))
        rate.sleep()
if _name_ == '_main_':
  try:
        publish_message()
  except rospy.ROSInterruptException:
        pass
```

OUTPUT:



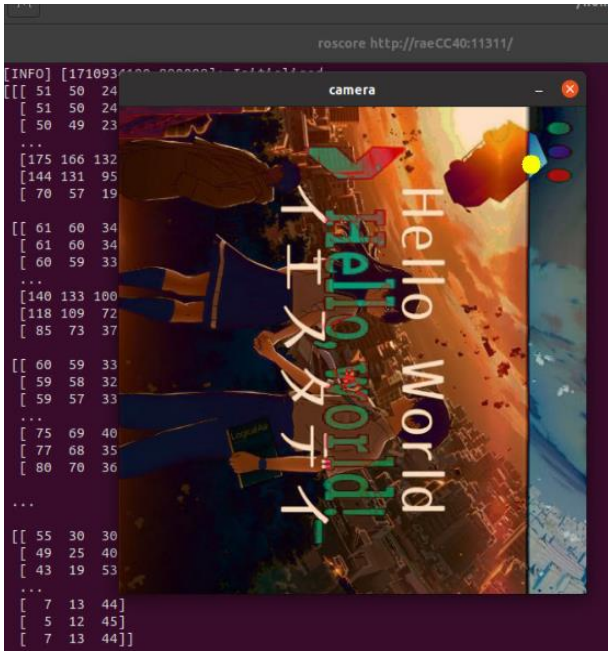
7. Find the difference between the two images.

CODE:

```
import rospy # Python library for ROS
from sensor_msgs.msg import Image # Image is the message type
from cv_bridge import CvBridge # Package to convert between ROS and OpenCV
Images
import cv2 # OpenCV library
def publish_message():
  pub = rospy.Publisher('video_frames', Image, queue_size=10)
```

```
    rospy.init_node('video_pub_py', anonymous=True)
    rate = rospy.Rate(10) # 10hz
    cap = cv2.imread('/home/rae/Downloads/test.png',0)
    br = CvBridge()
    while not rospy.is_shutdown():
            rospy.loginfo('publishing video frame')
            pub.publish(br.cv2_to_imgmsg(cap))
            rate.sleep()
if _name_ == '_main_':
    try:
            publish_message()
    except rospy.ROSInterruptException:
            pass
```

OUTPUT:



8. Write a python program in ROS to sort a given set of numbers.

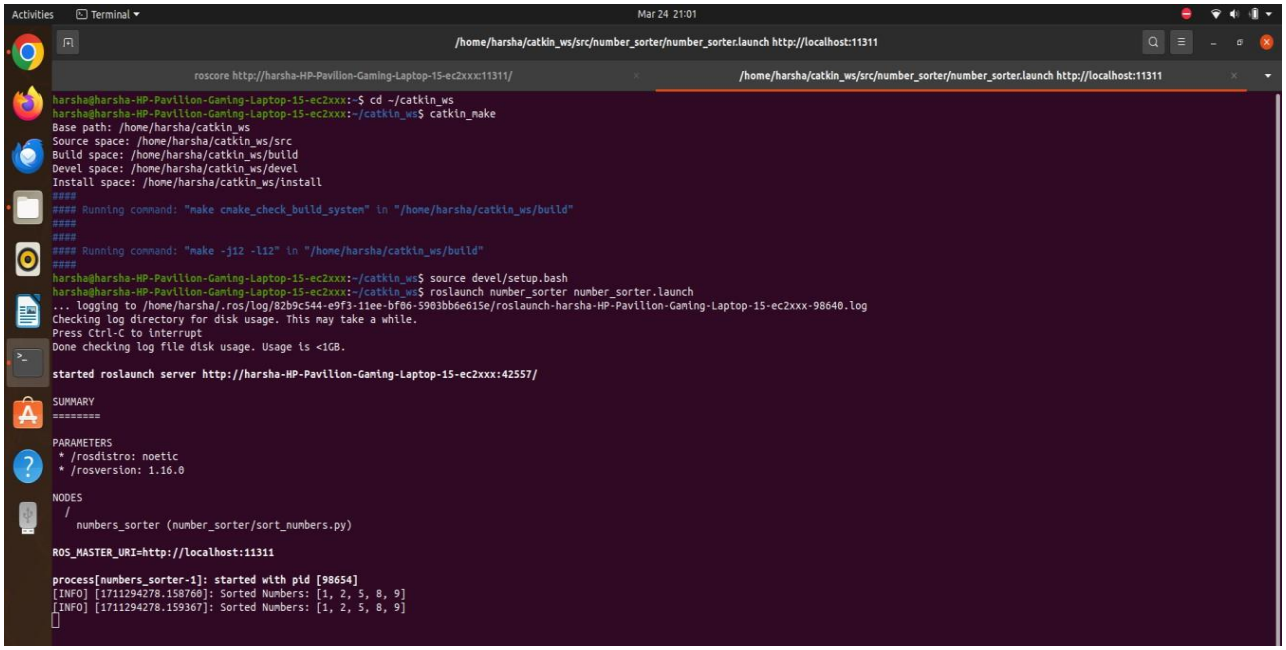
CODE:

```
import rospy
from std_msgs.msg import Int32MultiArray
def sort_numbers(numbers):
   sorted_numbers = sorted(numbers.data)
   rospy.loginfo("Sorted Numbers:
   {}".format(sorted_numbers)) return
   sorted_numbers
def numbers_callback(data):
```

```
    rospy.loginfo("Received Numbers:
    {}".format(data.data)) sorted_numbers =
    sort_numbers(data)
    rospy.signal_shutdown("Numbers sorted successfully. Shutting down...") #
    Shutdown after sorting
  def numbers_sorter():
    rospy.init_node('numbers_sorter',
    anonymous=True)
    numbers = Int32MultiArray(data=[5, 2, 8, 1, 9])  # Predefined numbers to
    sort sorted_numbers = sort_numbers(numbers)
    rospy.loginfo("Sorted Numbers:
    {}".format(sorted_numbers)) rospy.spin()

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

OUTPUT:



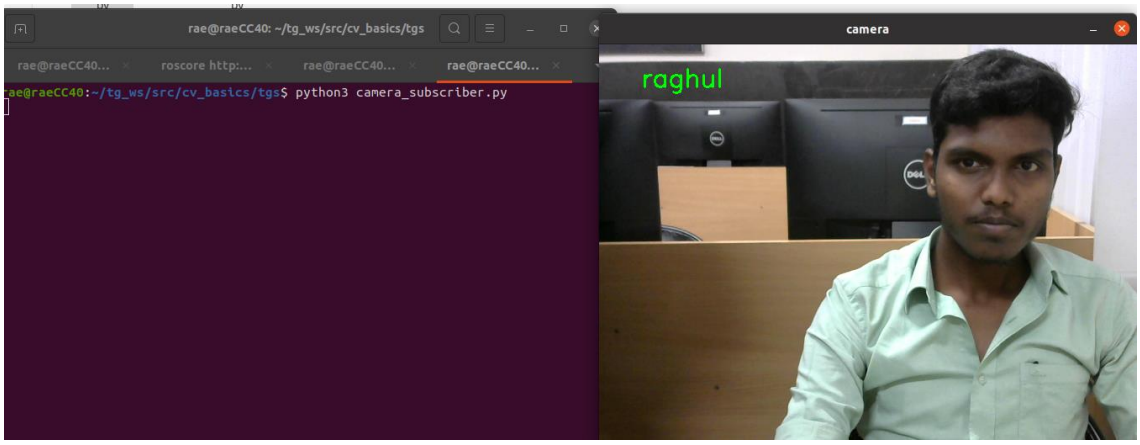
9. Stream the video from USB camera in RoS and write your name on the Stream.

CODE:

```
#!/usr/bin/env python3
import rospy # Python library for ROS
from sensor_msgs.msg import Image # Image is the message type
from cv_bridge import CvBridge # Package to convert between ROS and OpenCV
Images
import cv2 # OpenCV library
```

```
def callback(data):
  br = CvBridge()
  rospy.loginfo("receiving video frame")
  current_frame = br.imgmsg_to_cv2(data)
current_frame=cv2.putText(current_frame,"hiiiii",(50,50),cv2.FONT_HERSHEY_SI
MPLEX,1,(255,0,255),2,cv2.LINE_AA)
  current_frame=cv2.circle(current_frame,(60,60),10,(0,255,255),-1)
  cv2.imshow("camera", current_frame)
  cv2.waitKey(1)
def receive_message():
   rospy.init_node('video_sub_py', anonymous=True)
   rospy.Subscriber('video_frames', Image, callback)
   rospy.spin()
  cv2.destroyAllWindows()
if _name_ == '_main_':
  receive_message()
```

OUTPUT:



10. Simulate a world of your own in Gazebo and Rviz and spawn a turtlebot on it. Environment with turtlebot 3 has been setup:

SETUP:

Download link:
https://github.com/SakshayMahna/Robotics-Playground/tree/main/turtlebot3_ws

Unzip into home dir.
Open terminal
    roscore
Open another terminal
    sudo apt-get install ros-noetic-navigation
Open another terminal
    cd turtlebot3_ws/
    catkin_make

source devel/setup.bash

TO RUN TURTLEBOT3:

Terminal 1

roscore

Terminal 2

cd turtlebot3_ws/
catkin_make
source devel/setup.bash
roslaunch ros_world turtlebot3_world.launch

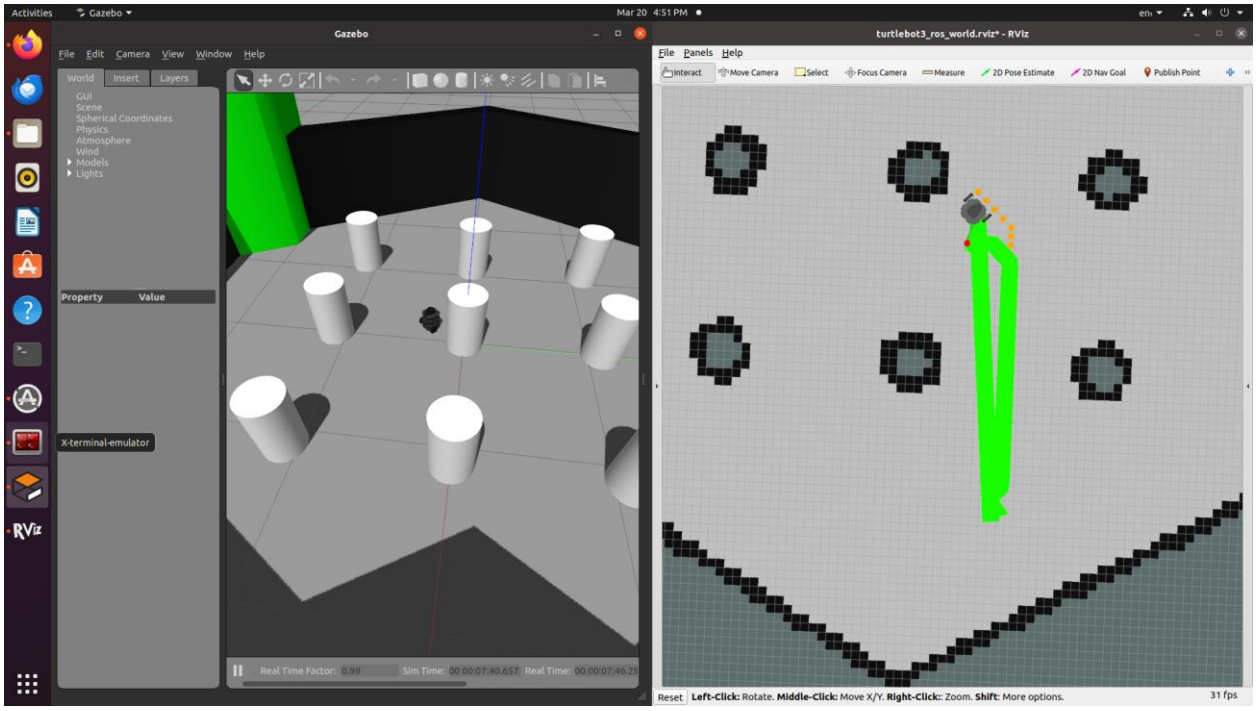
Terminal 3

cd turtlebot3_ws/
catkin_make
source devel/setup.bash
roslaunch global_path_planning turtlebot3_ros_world.launch

Terminal 4

cd turtlebot3_ws/
catkin_make
source devel/setup.bash
rosrun global_path_planning path_planning_server.py

OUTPUT:



Department of RAE			
Criteria	**Excellent (75% - 100%)**	**Good (50 – 75%)**	**Poor (<50%)**
Preparation (30)			
Performance (30)			
Evaluation (20)			
Report (20)			
Sign:		**Total (100)**	

Result:

Thus, the ROS basics has been successfully implemented.