# Aim:

Experiment 10 Basics of RoS

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

# Software/ Package Used:

RoS

# Programs:

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

# CODE:

**PUBLISHER:**

#!/usr/bin/env python 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:

#!/usr/bin/env python import rospy

from std\_msgs.msg import String

def callback(data):

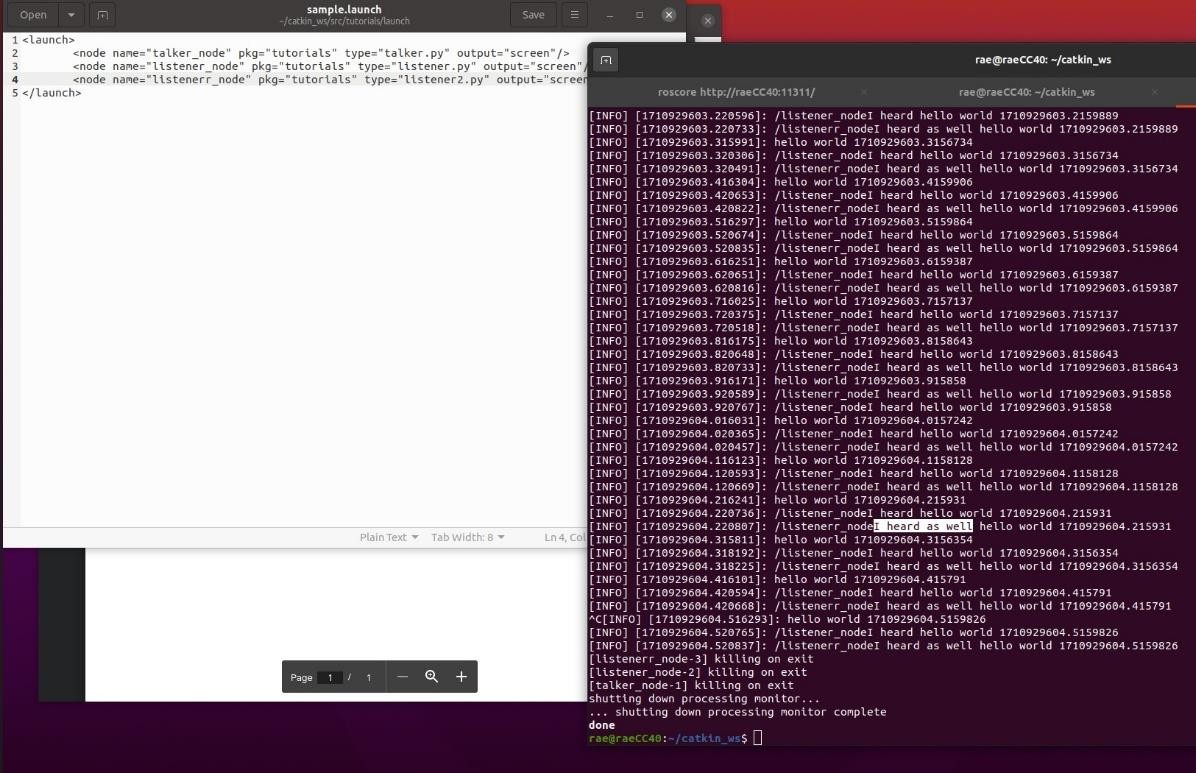
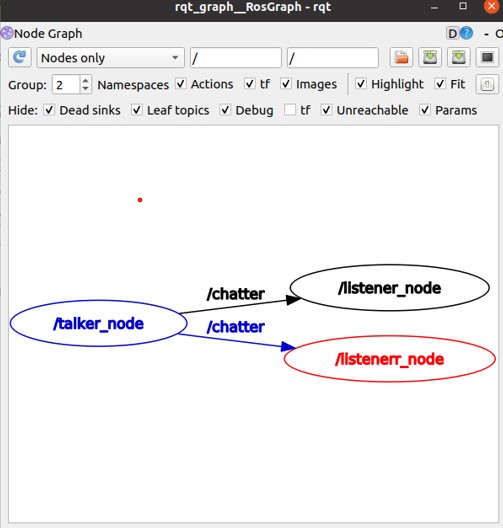
rospy.loginfo(rospy.get\_caller\_id() + 'I heard %s', data.data) def listener():

rospy.init\_node('listener', anonymous=True)

rospy.Subscriber('chatter', String, callback) rospy.spin()

if name == ' main ': listener()

# OUTPUT:



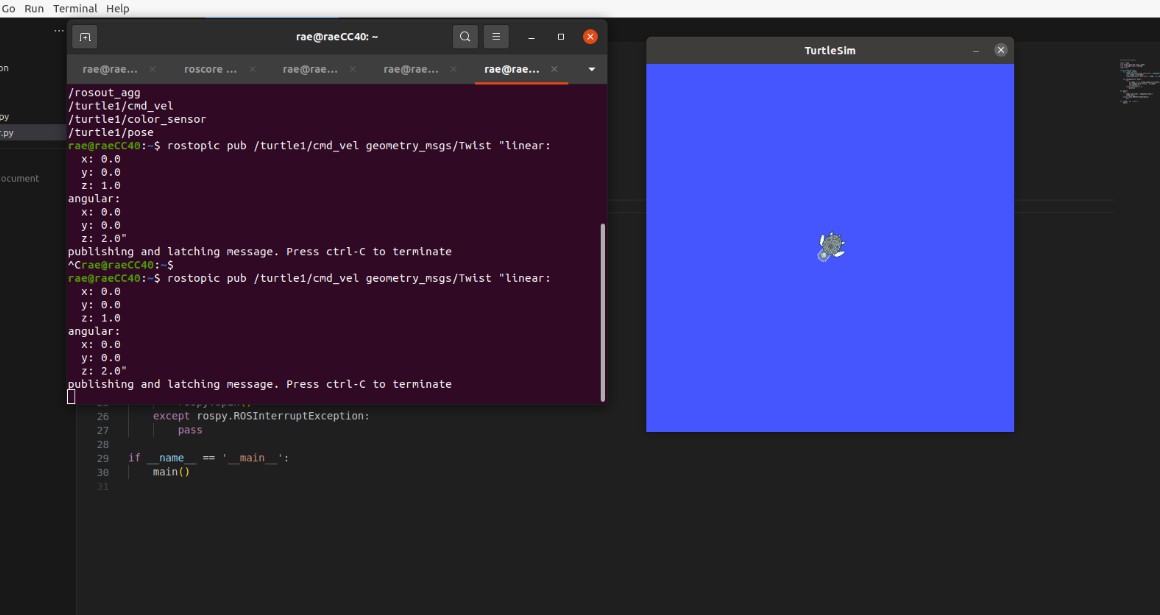
1. **Run turtlesim. Commands:**

# rosrun turtlesim turtlesim\_node

* + **In another terminal, rostopic list**

# Rostopic pub /cmd\_vel OUTPUT:



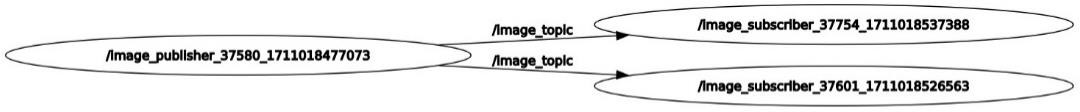


1. **RQT Graph. Command:**

# On running nodes on multiple terminal eg: publishing and subscribing an image

* + **On another terminal, type rqt\_graph**

# OUTPUT:



1. **Bridge RoS with openCV. Read an image into RoS and rotate the image. CODE:**

# PUBLISHER:

#!/usr/bin/env python

import rospy

from sensor\_msgs.msg import Image import cv2

from cv\_bridge import CvBridge import os

class ImagePublisher: def init (self):

rospy.init\_node('image\_publisher', anonymous=True)

self.image\_pub = rospy.Publisher('image\_topic', Image, queue\_size=10) self.bridge = CvBridge()

self.rate = rospy.Rate(1) # Publish rate of 1 Hz self.image\_folder = "/home/rae/Downloads"

def publish\_image(self):

while not rospy.is\_shutdown():

for root, dirs, files in os.walk(self.image\_folder): for file in files:

if file.endswith(".jpeg") or file.endswith(".png"): image\_path = os.path.join(root, file)

cv\_image = cv2.imread(image\_path) if cv\_image is not None:

ros\_image = self.bridge.cv2\_to\_imgmsg(cv\_image, "bgr8") self.image\_pub.publish(ros\_image)

rospy.loginfo("Image published: {}".format(image\_path)) self.rate.sleep()

def main(): try:

image\_publisher = ImagePublisher() image\_publisher.publish\_image()

except rospy.ROSInterruptException: pass

if name == ' main ': main()

# SUBSCRIBER:

#!/usr/bin/env python

import rospy

from sensor\_msgs.msg import Image from cv\_bridge import CvBridge import cv2

class ImageSubscriber: def init (self):

rospy.init\_node('image\_subscriber', anonymous=True) self.bridge = CvBridge()

rospy.Subscriber('image\_topic', Image, self.image\_callback) self.cv\_window\_name = 'Subscribed Image'

def image\_callback(self, msg): try:

cv\_image = self.bridge.imgmsg\_to\_cv2(msg, desired\_encoding="bgr8") cv2.rectangle(cv\_image, (100, 100), (200, 200), (0, 255, 0), 2)

font = cv2.FONT\_HERSHEY\_SIMPLEX

cv2.putText(cv\_image, 'AI and vision', (20, 50), font, 1, (0, 0, 255), 2, cv2.LINE\_AA)

cv2.imshow(self.cv\_window\_name, cv\_image) cv2.waitKey(1)

except Exception as e: rospy.logerr(e)

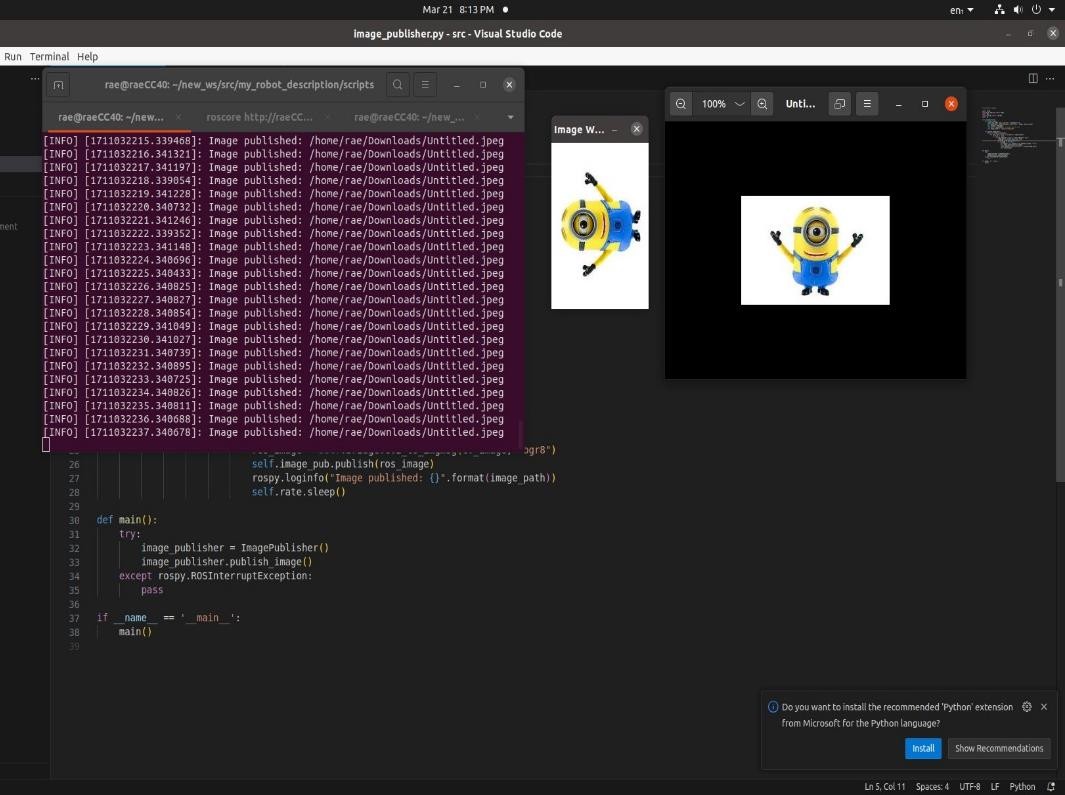
def run(self): rospy.spin()

if name == ' main ': try:

rospy.init\_node('image\_subscriber', anonymous=True) image\_subscriber = ImageSubscriber() image\_subscriber.run()

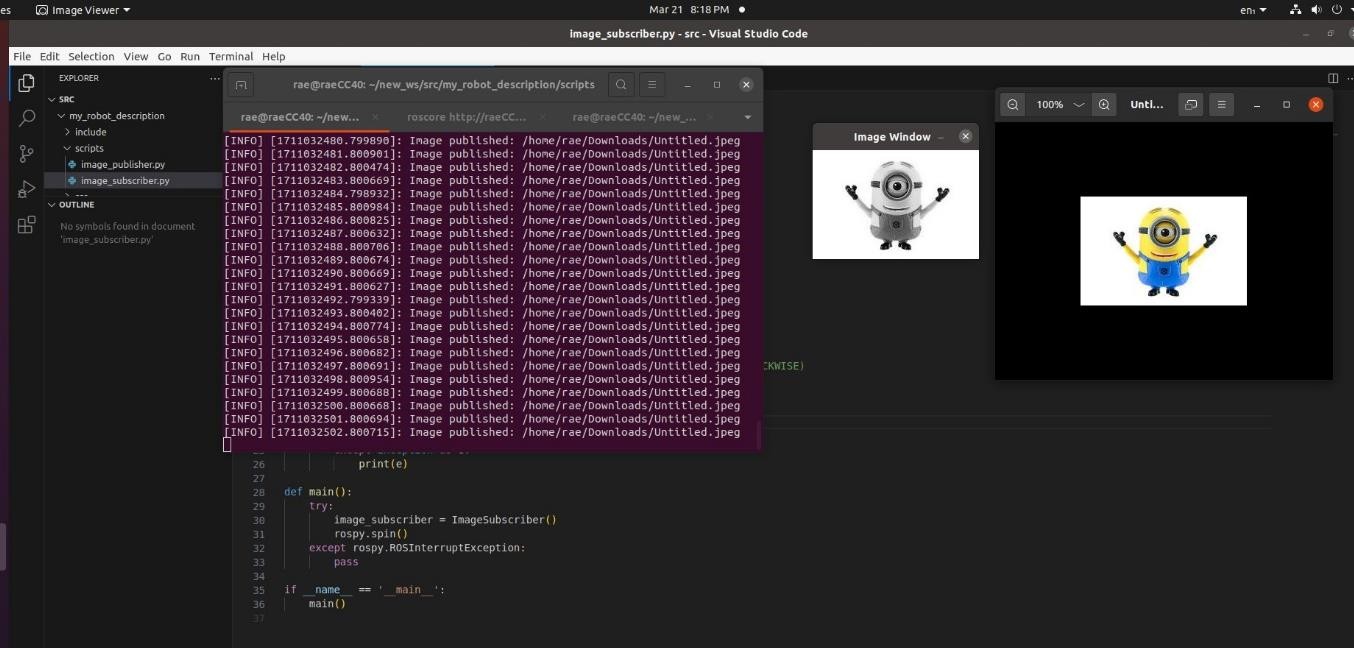
except rospy.ROSInterruptException: cv2.destroyAllWindows()

# OUTPUT:

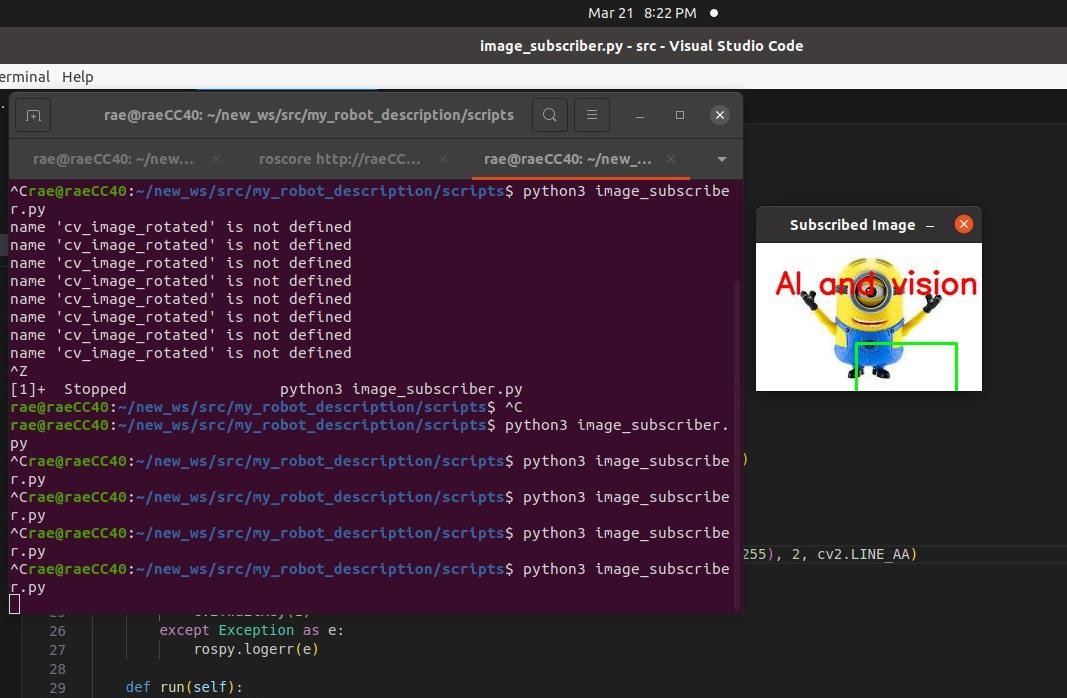


1. **Read an image into RoS and perform colour conversions on an image.**

# OUTPUT:



1. **Write AI and vison on an image. OUTPUT:**



# Find the difference between the two images. CODE:

**PUBLISHER:**

#!/usr/bin/env python3

import rospy

from sensor\_msgs.msg import Image from cv\_bridge import CvBridge import cv2

def publish\_image\_diff(): rospy.init\_node('image\_diff\_publisher', anonymous=True)

image\_pub = rospy.Publisher('image\_difference', Image, queue\_size=10) rate = rospy.Rate(10) # 10 Hz

bridge = CvBridge()

cap = cv2.VideoCapture(0)

if not cap.isOpened():

rospy.logerr("Error: Failed to open camera.") return

prev\_frame = None

while not rospy.is\_shutdown(): ret, frame = cap.read()

if not ret:

rospy.logerr("Error: Failed to capture frame.")

break

if prev\_frame is not None:

frame\_diff = cv2.absdiff(frame, prev\_frame)

diff\_msg = bridge.cv2\_to\_imgmsg(frame\_diff, "bgr8") image\_pub.publish(diff\_msg)

prev\_frame = frame.copy() rate.sleep()

cap.release()

if name == ' main ': try:

publish\_image\_diff()

except rospy.ROSInterruptException: pass

# SUBSCRIBER:

#!/usr/bin/env python3

import rospy

from sensor\_msgs.msg import Image from cv\_bridge import CvBridge import cv2

class ImageSubscriber: def init (self):

rospy.init\_node('image\_subscriber', anonymous=True) self.bridge = CvBridge()

self.sub = rospy.Subscriber('image\_difference', Image, self.callback)

def callback(self, data): try:

cv\_image = self.bridge.imgmsg\_to\_cv2(data, "bgr8") cv2.imshow("Image Difference", cv\_image) cv2.waitKey(1)

except Exception as e: print(e)

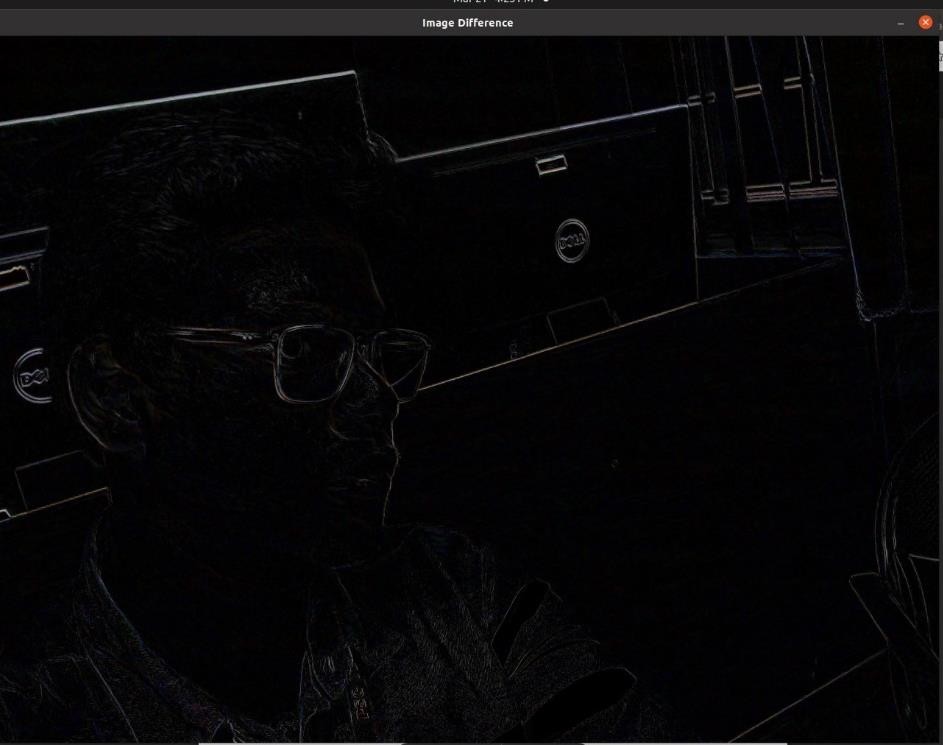
def main(): try:

image\_subscriber = ImageSubscriber() rospy.spin()

except rospy.ROSInterruptException: pass

if name == ' main ': main()

# OUTPUT:



1. **Write a python program in RoS to sort a given set of numbers. CODE:**

# PUBLISHER:

#!/usr/bin/env python3

import rospy

from std\_msgs.msg import Int32MultiArray import random

def publish\_numbers(): rospy.init\_node('sorting\_publisher', anonymous=True)

numbers\_pub = rospy.Publisher('numbers\_to\_sort', Int32MultiArray, queue\_size=10)

rate = rospy.Rate(1) # 1 Hz

while not rospy.is\_shutdown():

# Generate a list of random numbers

numbers = [random.randint(0, 100) for \_ in range(10)]

# Publish the numbers numbers\_pub.publish(Int32MultiArray(data=numbers))

rate.sleep()

if name == ' main ': try:

publish\_numbers()

except rospy.ROSInterruptException: pass

# SUBSCRIBER:

#!/usr/bin/env python3

import rospy

from std\_msgs.msg import Int32MultiArray

def sort\_numbers(numbers): sorted\_numbers = sorted(numbers) return sorted\_numbers

def numbers\_callback(msg): numbers = msg.data

sorted\_numbers = sort\_numbers(numbers) rospy.loginfo("Received numbers: {}".format(numbers)) rospy.loginfo("Sorted numbers: {}".format(sorted\_numbers))

def subscribe\_to\_numbers(): rospy.init\_node('sorting\_subscriber', anonymous=True)

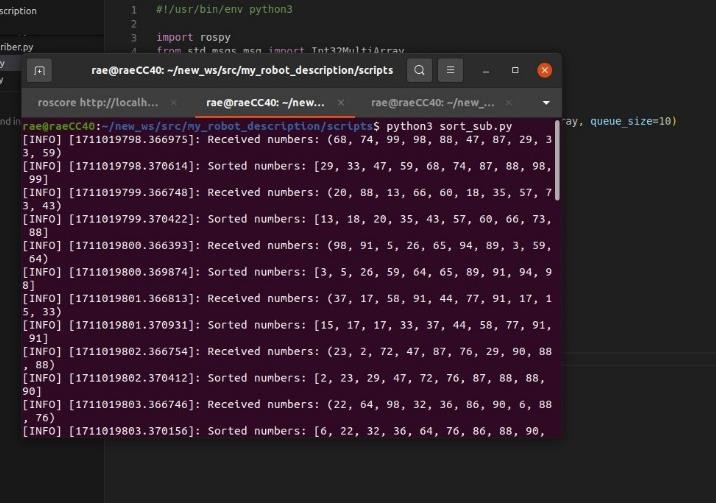
rospy.Subscriber('numbers\_to\_sort', Int32MultiArray, numbers\_callback) rospy.spin()

if name == ' main ': try:

subscribe\_to\_numbers()

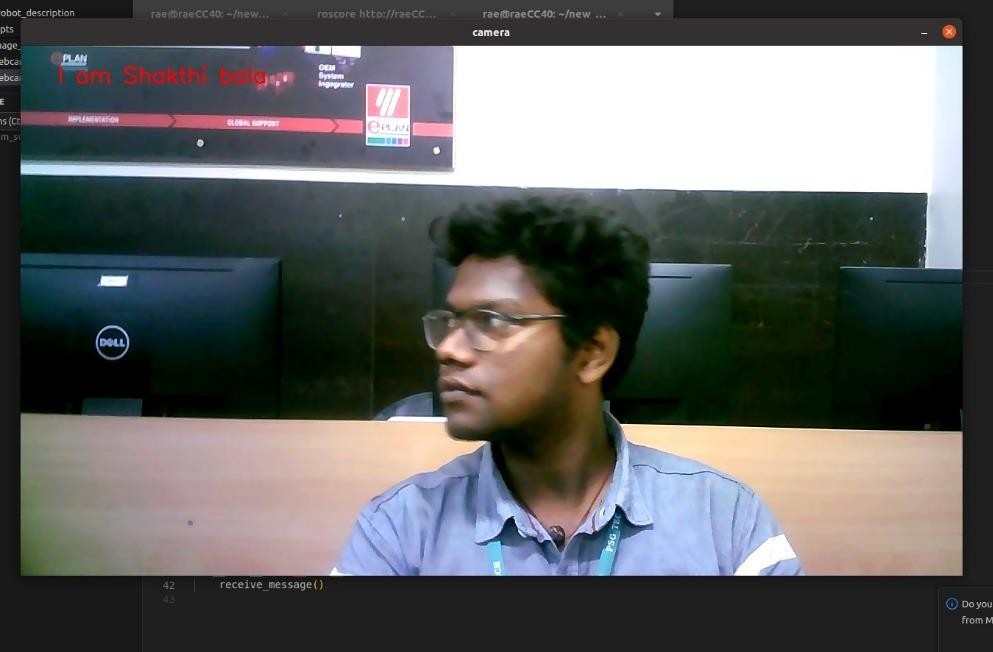
except rospy.ROSInterruptException: pass

# OUTPUT:



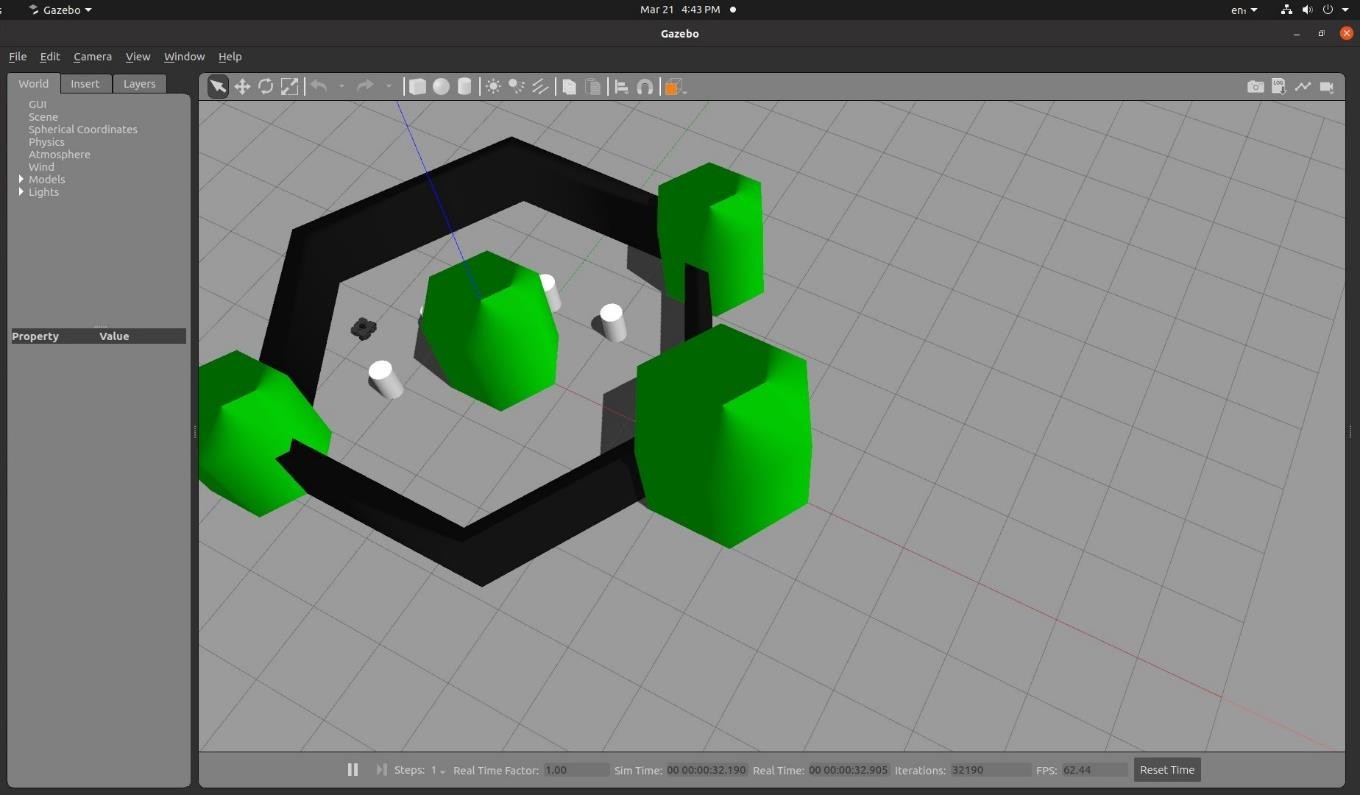
1. **Stream the video from USB camera in RoS and write your name on the stream.**

# OUTPUT:

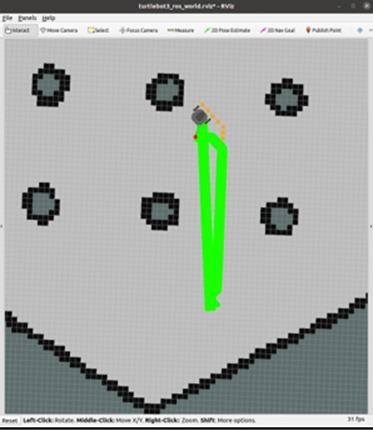


1. **Simulate a world of your own in Gazebo and Rviz and spawn a turtlebot on it.**

# OUTPUT:



**RVIZ 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.