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
```

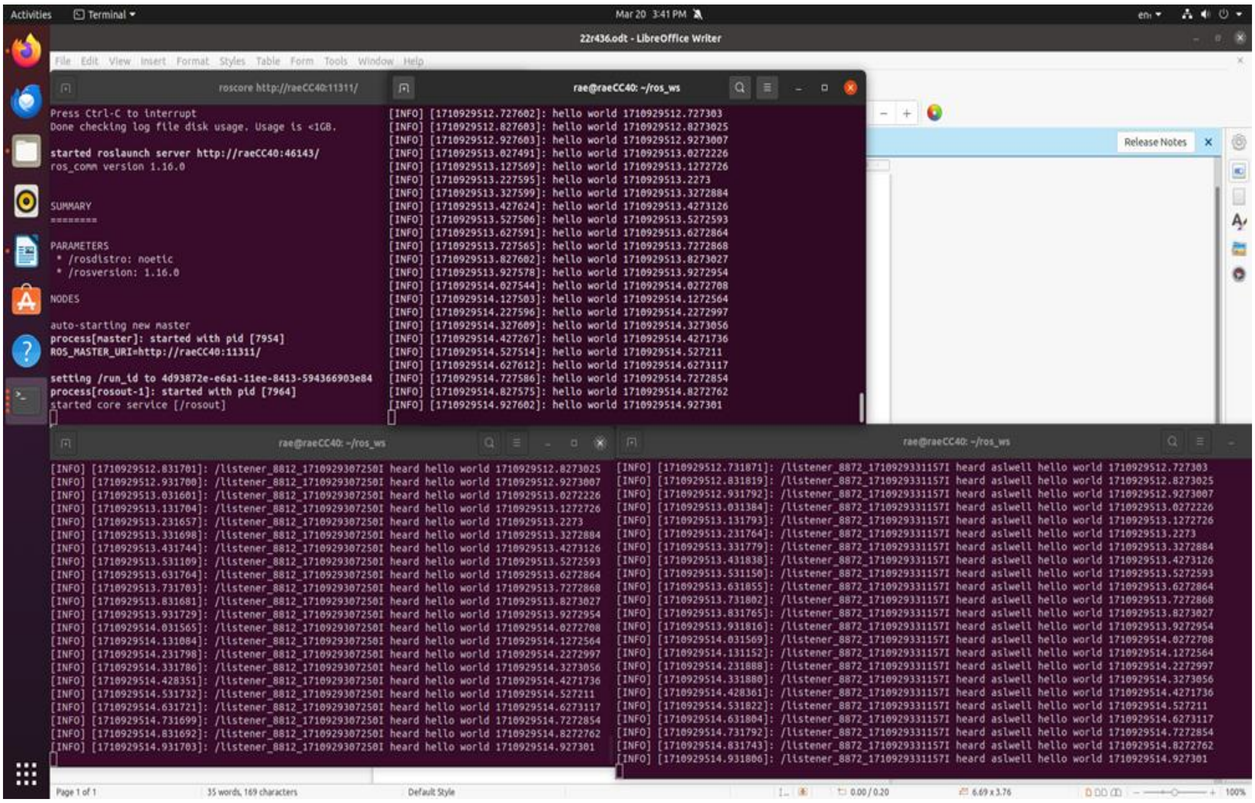
SUBSCRIBER 1 CODE:

```
#!/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:

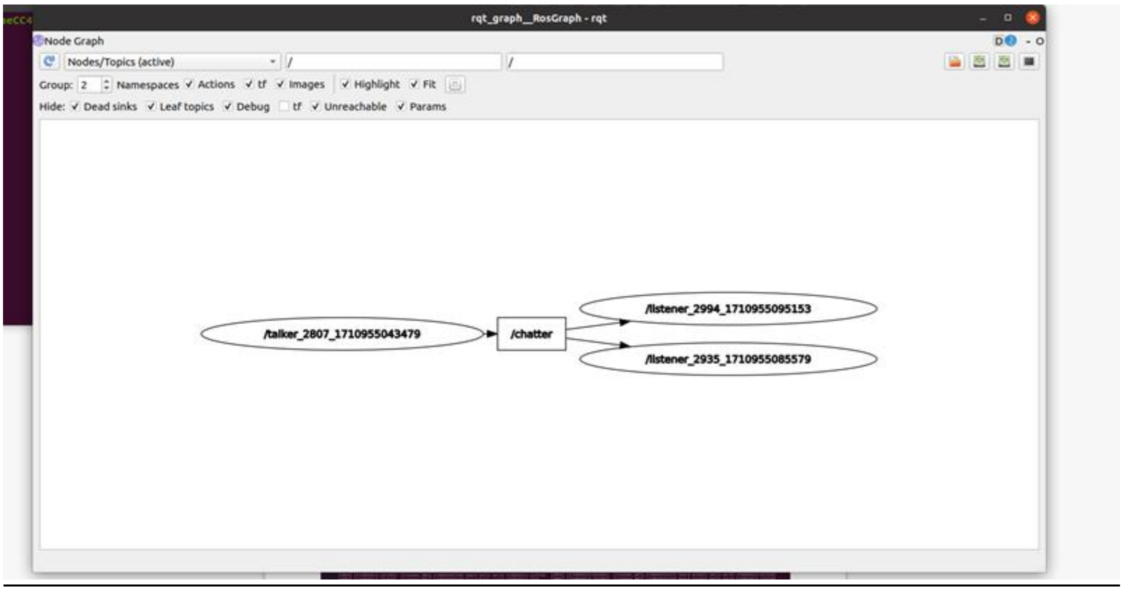


```
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://raeCC40:46143/
ros_comm version 1.16.0


SUMMARY
========

PARAMETERS
 * /rosdistro: noetic
 * /rosversion: 1.16.0

NODES

auto-starting new master
process[master]: started with pid [7954]
ROS_MASTER_URI=http://raeCC40:11311/

setting /run_id to 4d93872e-e6a1-11ee-8413-594366903e84
process[rosout-1]: started with pid [7964]
started core service [/rosout]
```

```
[INFO] [1710929512.727602]: hello world 1710929512.727303
[INFO] [1710929512.827603]: hello world 1710929512.8273025
[INFO] [1710929512.927603]: hello world 1710929512.9273007
[INFO] [1710929513.027491]: hello world 1710929513.0272226
[INFO] [1710929513.127569]: hello world 1710929513.1272726
[INFO] [1710929513.227595]: hello world 1710929513.2273
[INFO] [1710929513.327599]: hello world 1710929513.3272884
[INFO] [1710929513.427624]: hello world 1710929513.4273126
[INFO] [1710929513.527506]: hello world 1710929513.5272593
[INFO] [1710929513.627591]: hello world 1710929513.6272864
[INFO] [1710929513.727565]: hello world 1710929513.7272868
[INFO] [1710929513.827602]: hello world 1710929513.8273027
[INFO] [1710929513.927578]: hello world 1710929513.9272954
[INFO] [1710929514.027544]: hello world 1710929514.0272708
[INFO] [1710929514.127503]: hello world 1710929514.1272564
[INFO] [1710929514.227596]: hello world 1710929514.2272997
[INFO] [1710929514.327609]: hello world 1710929514.3273056
[INFO] [1710929514.427267]: hello world 1710929514.4271736
[INFO] [1710929514.527514]: hello world 1710929514.527211
[INFO] [1710929514.627612]: hello world 1710929514.6273117
[INFO] [1710929514.727586]: hello world 1710929514.7272854
[INFO] [1710929514.827575]: hello world 1710929514.8272762
[INFO] [1710929514.927602]: hello world 1710929514.927301
```

```
[INFO] [1710929512.831701]: /listener_8812_1710929307250I heard hello world 1710929512.8273025
[INFO] [1710929512.931700]: /listener_8812_1710929307250I heard hello world 1710929512.9273007
[INFO] [1710929513.031601]: /listener_8812_1710929307250I heard hello world 1710929513.0272226
[INFO] [1710929513.131704]: /listener_8812_1710929307250I heard hello world 1710929513.1272726
[INFO] [1710929513.231657]: /listener_8812_1710929307250I heard hello world 1710929513.2273
[INFO] [1710929513.331698]: /listener_8812_1710929307250I heard hello world 1710929513.3272884
[INFO] [1710929513.431744]: /listener_8812_1710929307250I heard hello world 1710929513.4273126
[INFO] [1710929513.531109]: /listener_8812_1710929307250I heard hello world 1710929513.5272593
[INFO] [1710929513.631764]: /listener_8812_1710929307250I heard hello world 1710929513.6272864
[INFO] [1710929513.731703]: /listener_8812_1710929307250I heard hello world 1710929513.7272868
[INFO] [1710929513.831681]: /listener_8812_1710929307250I heard hello world 1710929513.8273027
[INFO] [1710929513.931729]: /listener_8812_1710929307250I heard hello world 1710929513.9272954
[INFO] [1710929514.031565]: /listener_8812_1710929307250I heard hello world 1710929514.0272708
[INFO] [1710929514.131084]: /listener_8812_1710929307250I heard hello world 1710929514.1272564
[INFO] [1710929514.231798]: /listener_8812_1710929307250I heard hello world 1710929514.2272997
[INFO] [1710929514.331786]: /listener_8812_1710929307250I heard hello world 1710929514.3273056
[INFO] [1710929514.428351]: /listener_8812_1710929307250I heard hello world 1710929514.4271736
[INFO] [1710929514.531732]: /listener_8812_1710929307250I heard hello world 1710929514.527211
[INFO] [1710929514.631721]: /listener_8812_1710929307250I heard hello world 1710929514.6273117
[INFO] [1710929514.731699]: /listener_8812_1710929307250I heard hello world 1710929514.7272854
[INFO] [1710929514.831692]: /listener_8812_1710929307250I heard hello world 1710929514.8272762
[INFO] [1710929514.931703]: /listener_8812_1710929307250I heard hello world 1710929514.927301
```

```
[INFO] [1710929512.731871]: /listener_8872_1710929331157I heard aslwell hello world 1710929512.727303
[INFO] [1710929512.831819]: /listener_8872_1710929331157I heard aslwell hello world 1710929512.8273025
[INFO] [1710929512.931792]: /listener_8872_1710929331157I heard aslwell hello world 1710929512.9273007
[INFO] [1710929513.031384]: /listener_8872_1710929331157I heard aslwell hello world 1710929513.0272226
[INFO] [1710929513.131793]: /listener_8872_1710929331157I heard aslwell hello world 1710929513.1272726
[INFO] [1710929513.231764]: /listener_8872_1710929331157I heard aslwell hello world 1710929513.2273
[INFO] [1710929513.331779]: /listener_8872_1710929331157I heard aslwell hello world 1710929513.3272884
[INFO] [1710929513.431838]: /listener_8872_1710929331157I heard aslwell hello world 1710929513.4273126
[INFO] [1710929513.531150]: /listener_8872_1710929331157I heard aslwell hello world 1710929513.5272593
[INFO] [1710929513.631855]: /listener_8872_1710929331157I heard aslwell hello world 1710929513.6272864
[INFO] [1710929513.731802]: /listener_8872_1710929331157I heard aslwell hello world 1710929513.7272868
[INFO] [1710929513.831765]: /listener_8872_1710929331157I heard aslwell hello world 1710929513.8273027
[INFO] [1710929513.931816]: /listener_8872_1710929331157I heard aslwell hello world 1710929513.9272954
[INFO] [1710929514.031569]: /listener_8872_1710929331157I heard aslwell hello world 1710929514.0272708
[INFO] [1710929514.131152]: /listener_8872_1710929331157I heard aslwell hello world 1710929514.1272564
[INFO] [1710929514.231888]: /listener_8872_1710929331157I heard aslwell hello world 1710929514.2272997
[INFO] [1710929514.331880]: /listener_8872_1710929331157I heard aslwell hello world 1710929514.3273056
[INFO] [1710929514.428361]: /listener_8872_1710929331157I heard aslwell hello world 1710929514.4271736
[INFO] [1710929514.531822]: /listener_8872_1710929331157I heard aslwell hello world 1710929514.527211
[INFO] [1710929514.631804]: /listener_8872_1710929331157I heard aslwell hello world 1710929514.6273117
[INFO] [1710929514.731792]: /listener_8872_1710929331157I heard aslwell hello world 1710929514.7272854
[INFO] [1710929514.831743]: /listener_8872_1710929331157I heard aslwell hello world 1710929514.8272762
[INFO] [1710929514.931806]: /listener_8872_1710929331157I heard aslwell hello world 1710929514.927301
```



rqt_graph__RosGraph - rqt

Node Graph

Nodes/Topics (active) | / | /

Group: 2 ☑ Namespaces ☑ Actions ☑ tf ☑ Images ☑ Highlight ☑ Fit

Hide: ☑ Dead sinks ☑ Leaf topics ☑ Debug □ tf ☑ Unreachable ☑ Params

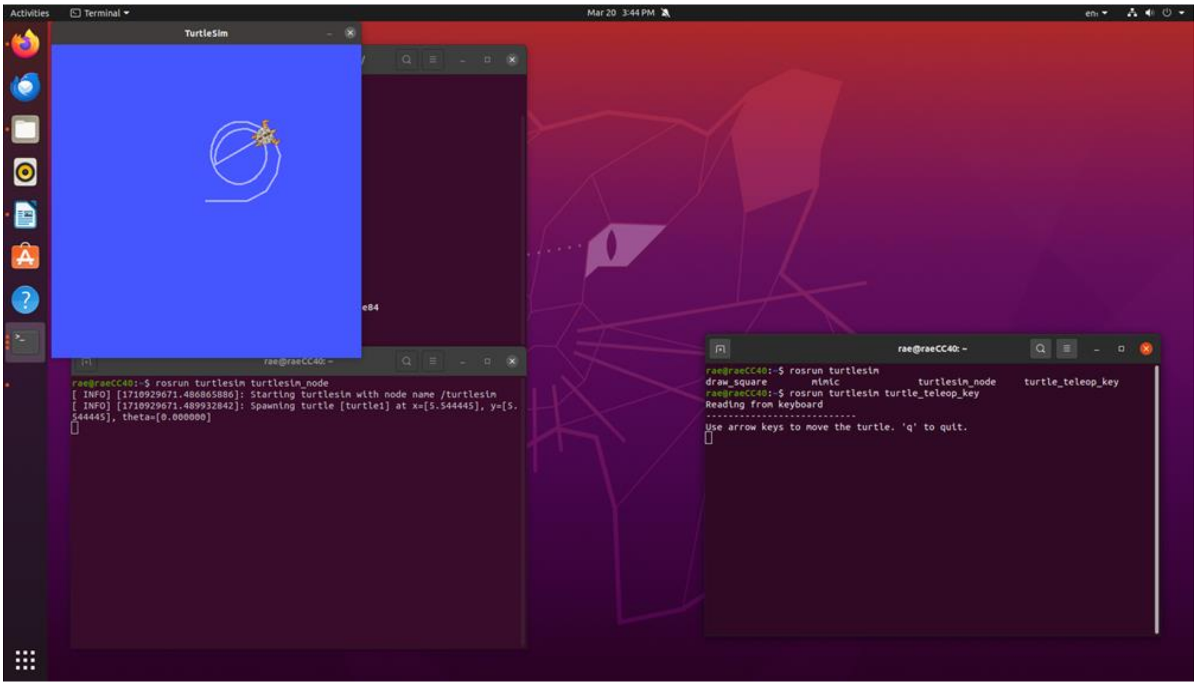
/talker_2807_1710955043479
/chatter
/listener_2994_1710955095153
/listener_2935_1710955085579

2. Run turtlesim.

COMMANDS:

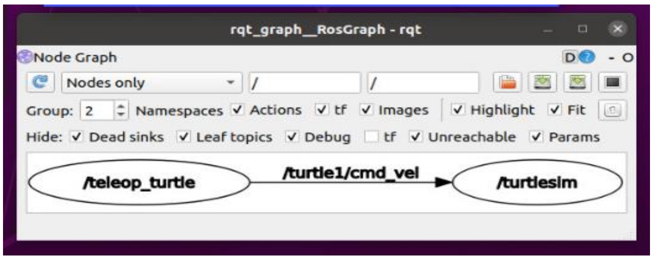
- rosrun turtlesim turtlesim_node
- rosrun turtlesim turtle_teleop_key
- rqt_graph

OUTPUT:



```
rae@raeCC40:~$ rosrun turtlesim turtlesim_node
[ INFO] [1710929671.486865886]: Starting turtlesim with node name /turtlesim
[ INFO] [1710929671.489932842]: Spawning turtle [turtle1] at x=[5.544445], y=[5.
544445], theta=[0.000000]
```

```
rae@raeCC40:~$ rosrun turtlesim
draw_square         mimic               turtlesim_node      turtle_teleop_key
rae@raeCC40:~$ rosrun turtlesim turtle_teleop_key
Reading from keyboard
---------------------------
Use arrow keys to move the turtle. 'q' to quit.
```



rqt_graph__RosGraph - rqt

Node Graph

Nodes only | / | /

Group: 2 ☑ Namespaces ☑ Actions ☑ tf ☑ Images ☑ Highlight ☑ Fit

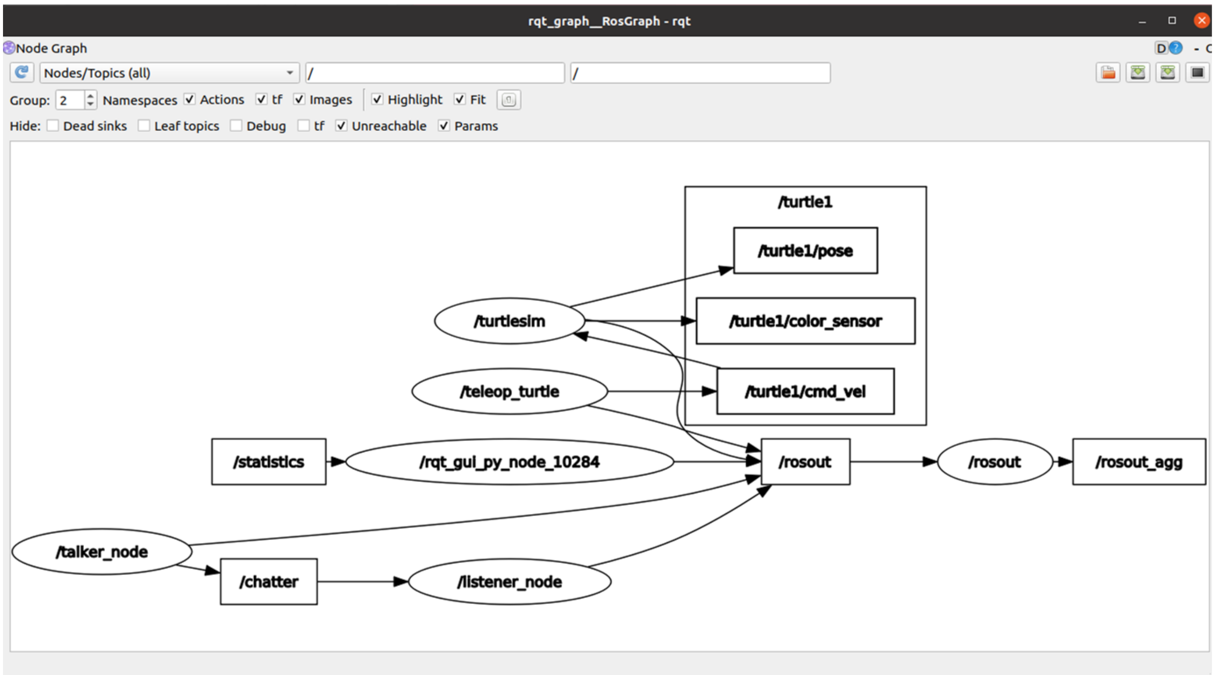
Hide: ☑ Dead sinks ☑ Leaf topics ☑ Debug □ tf ☑ Unreachable ☑ Params

3. RQT Graph.

COMMANDS:

- rqt_graph

OUTPUT:



rqt_graph__RosGraph - rqt

Node Graph

Nodes/Topics (all) / /

Group: 2 Namespaces ☑ Actions ☑ tf ☑ Images ☑ Highlight ☑ Fit

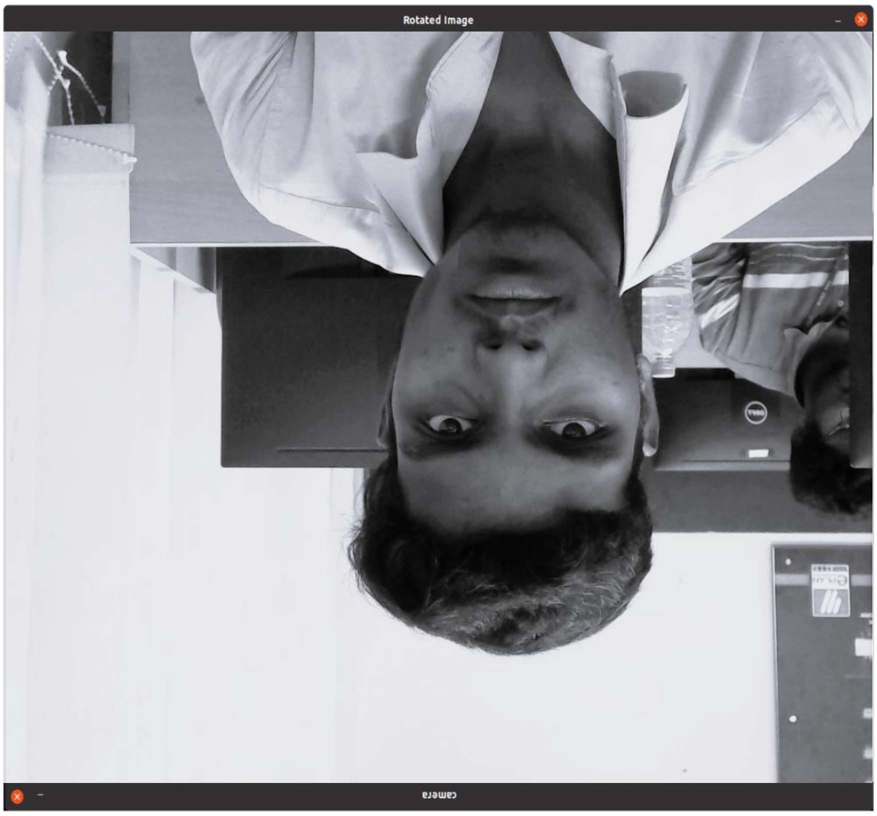
Hide: □ Dead sinks □ Leaf topics □ Debug □ tf ☑ Unreachable ☑ Params

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:

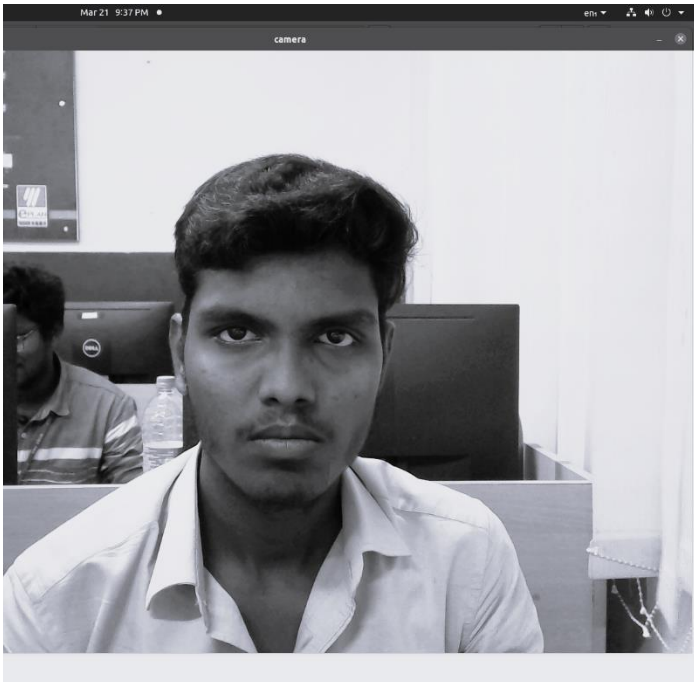


5. Read an image into ROS and perform color conversions on an image.

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:



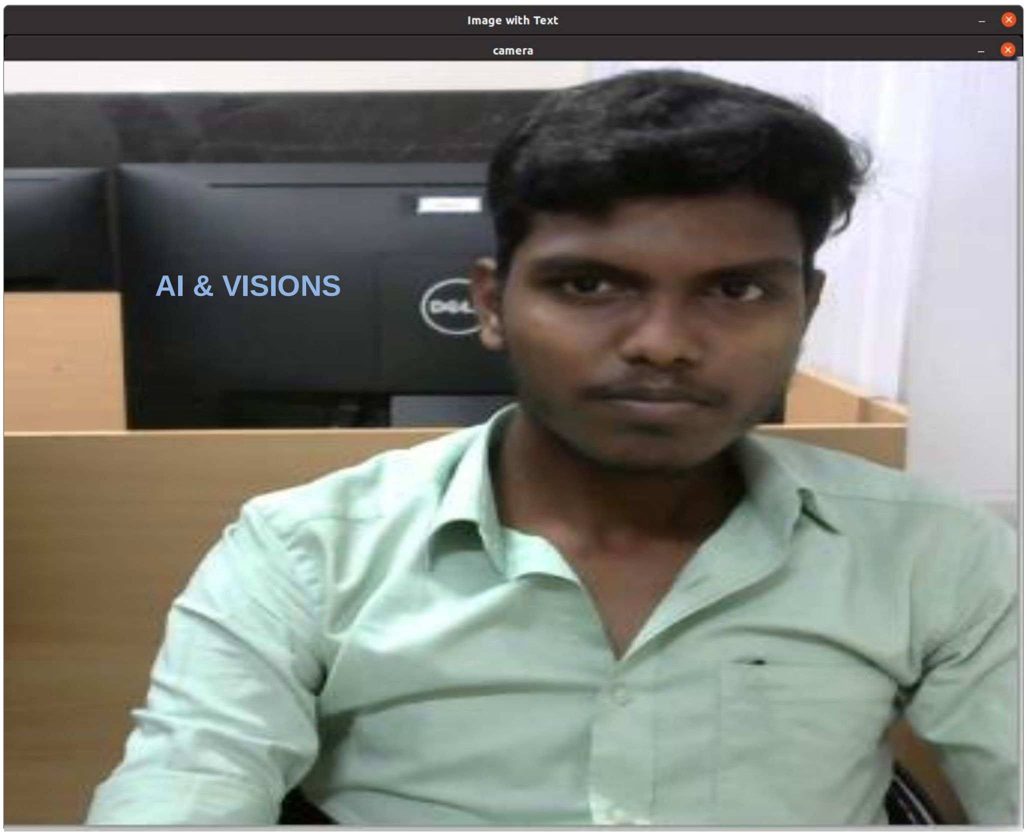
6. Write AI and vison on an image.

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:



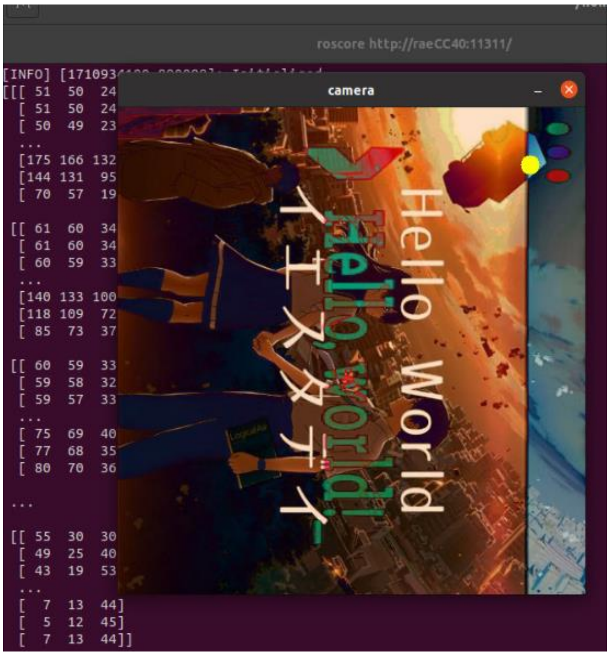
7. Find the difference between the two images.

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:



```
[INFO] [171093
[[[ 51  50  24
  [ 51  50  24
  [ 50  49  23
  ...
  [175 166 132
  [144 131  95
  [ 70  57  19

 [[ 61  60  34
  [ 61  60  34
  [ 60  59  33
  ...
  [140 133 100
  [118 109  72
  [ 85  73  37

 [[ 60  59  33
  [ 59  58  32
  [ 59  57  33
  ...
  [ 75  69  40
  [ 77  68  35
  [ 80  70  36

 ...

 [[ 55  30  30
  [ 49  25  40
  [ 43  19  53
  ...
  [  7  13  44]
  [  5  12  45]
  [  7  13  44]]
```

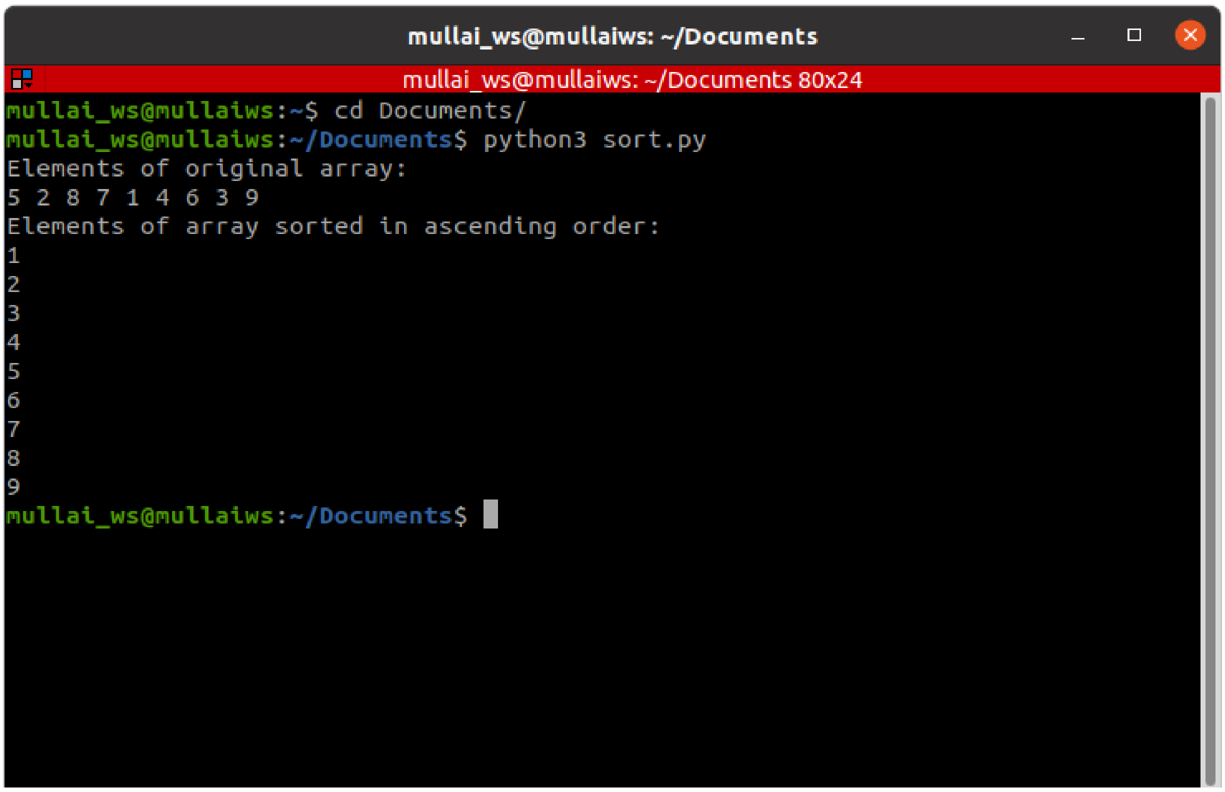
8. Write a python program in ROS to sort a given set of numbers.

CODE:

```
#Initialize array
arr = [5, 2, 8, 7, 1];
temp = 0;
#Displaying elements of original array
print("Elements of original array: ");
for i in range(0, len(arr)):
    print(arr[i], end=" ");
```

```
#Sort the array in ascending order
for i in range(0, len(arr)):
    for j in range(i+1, len(arr)):
        if(arr[i] > arr[j]):
            temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
print();
#Displaying elements of the array after sorting
print("Elements of array sorted in ascending order: ");
for i in range(0, len(arr)):
    print(arr[i], end=" ");
```

OUTPUT:



mullai_ws@mullaiws: ~/Documents
mullai_ws@mullaiws: ~/Documents 80x24

```
mullai_ws@mullaiws:~$ cd Documents/
mullai_ws@mullaiws:~/Documents$ python3 sort.py
Elements of original array:
5 2 8 7 1 4 6 3 9
Elements of array sorted in ascending order:
1
2
3
4
5
6
7
8
9
mullai_ws@mullaiws:~/Documents$
```

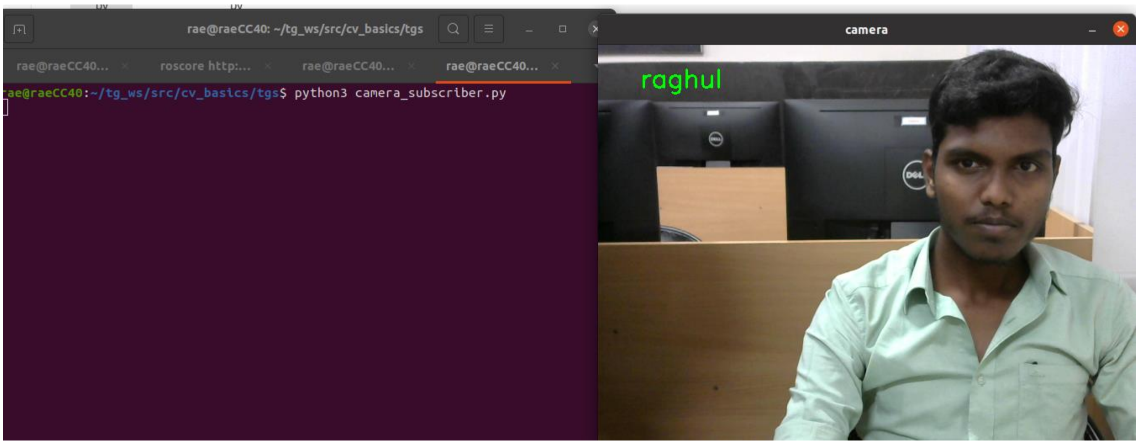
9. Stream the video from USB camera in RoS and write your name on the Stream.

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:



```
rae@raeCC40:~/tg_ws/src/cv_basics/tgs$ python3 camera_subscriber.py
```

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
```

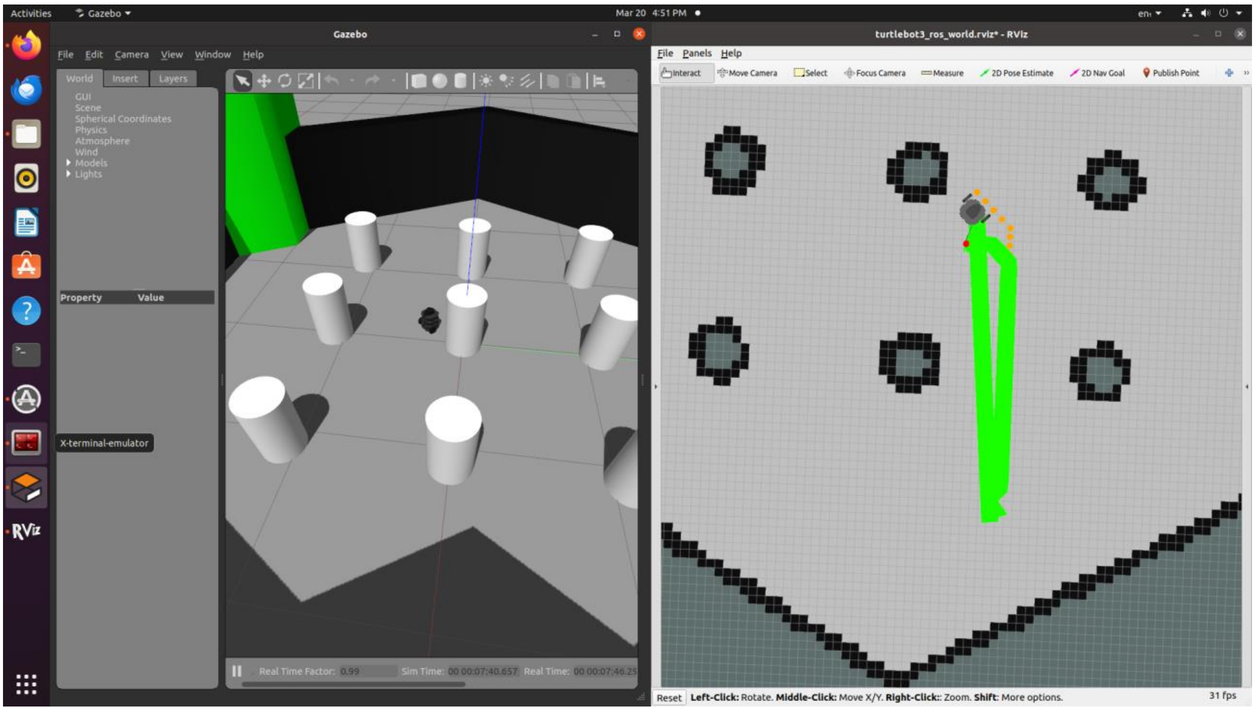
Terminal 3

```
        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.