# 0. Pub/Sub to publish and receive name in ROS

1. **AIM**: To print the name via pub/sub
2. **METHOD**:

* Create catkin workspace: mkdir -p catkin\_ws/src
* Create catkin package: catkin\_create\_pkg exp0\_postlab rospy roscpp std\_msgs
* Create scripts: cd src/exp0\_postlab/src && mkdir scripts && cd scripts && touch publisher.py && touch subscriber.py
* Add the scripts in this directory
* Catkin Make: catkin\_make

1. **CODE**:

* publisher.py

#!/usr/bin/env python  
import rospy  
from std\_msgs.msg import String  
  
def publisher():  
 pub = rospy.Publisher('chatter', String, queue\_size=10)  
 rospy.init\_node('talker', anonymous=True)  
 rate = rospy.Rate(1) # 10hz  
 while not rospy.is\_shutdown():  
 name = "Aaryamann Challani"  
 rospy.loginfo(name)  
 pub.publish(name)  
  
 rate.sleep()  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 try:  
 publisher()  
 except rospy.ROSInterruptException:  
 pass

* subscriber.py

#!/usr/bin/env python  
import rospy  
from std\_msgs.msg import String  
  
def callback(data):  
 rospy.loginfo(rospy.get\_caller\_id() + "Name:%s", data.data)  
  
def subscriber():  
 rospy.init\_node('listener', anonymous=True)  
 rospy.Subscriber('chatter', String, callback)  
 rospy.spin()  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 subscriber()

1. **RESULT**: Successfully published and received messages via ROS’s pubsub transport

# 1. Move the turtle bot in D shape

1. **AIM**: Move the turtlebot in a D path
2. **METHOD**:

* Create catkin package: catkin\_create\_pkg exp1\_postlab rospy roscpp std\_msgs
* Create script: cd src/exp1\_postlab/src && mkdir scripts && cd scripts && touch turtlesim\_d.py
* Add the script in this directory
* Catkin Make: catkin\_make

1. **RESULT**: Successfully launched a turtlebot and moved it in the D shape

# 2. Move the turtle bot in Hexagon shape

1. **AIM**: Move the turtlebot in a Hexagon path
2. **METHOD**:

* Create catkin package: catkin\_create\_pkg exp2\_postlab rospy roscpp std\_msgs
* Create script: cd src/exp2\_postlab/src && mkdir scripts && cd scripts && touch turtlesim\_hex.py
* Add the script in this directory
* Catkin Make: catkin\_make

1. **RESULT**: Successfully launched a turtlebot and moved it in the hexagon shape

# 3. 2-wheeled robot with caster wheel

1. **AIM**: To create a 2-wheeled robot with caster wheel using URDF
2. **METHOD**:

* Create catkin package: catkin\_create\_pkg exp3\_postlab
* Create urdf files: cd src/exp3\_postlab/src && mkdir urdf && mkdir launch && cd urdf && touch bot.urdf && cd ../launch && touch gazebo.launch
* Add the launch file and urdf definition in this directory
* Launch using roslaunch exp3\_postlab gazebo.launch

1. **RESULT**: Successfully simulated/defined parameters for a 2 wheeled robot with a caster wheel via URDF/ROS. Observed output in Gazebo.

# 4. 3 Joint Manipulator with gripper as end effector

1. **AIM**: To Create a manipulator using URDF
2. **METHOD**:

* Create catkin package: catkin\_create\_pkg exp4\_postlab
* Create urdf/launch files: cd src/exp4\_postlab/src && mkdir urdf && mkdir launch && cd urdf && touch manipulator.urdf && cd ../launch/ && touch gazebo.launch
* Add the files from this directory
* Catkin Make: catkin\_make
* Launch the bot: roslaunch exp4\_postlab gazebo.launch
* Note: From ROS Textbook

1. **RESULT**: Successfully created a 3 joint manipulator, and simulated in Gazebo

# 5. Image Processing using OpenCV + ROS

1. **AIM**: To find the distance of image from origin in ROS.
2. **METHOD**:

* Save your ros distribution in a variable export ROS\_DIST=<noetic|melodic|kinetic>
* Install USB Camera support in ROS sudo apt-get install ros-$ROS\_DIST-usb-cam
* Install OpenCV sudo apt-get install opencv-python
* Verify Installations are in $PATH
* Start roscore: roscore
* Launch USB Camera: rosrun usb\_cam usb\_cam\_node
* Run the opencv script: rosrun exp5\_postlab image\_detect.py