AA274A Section 3: Turtlebot Hardware and Software

Brian Dobkowski bdobkows@stanford.edu
AA274A, Stanford University

1. Question 1

These are the active ROS topics on the Turtlebot:
battery_state
cmd_vel
cmd_vel_rc100
diagnostics
firmware_version
imu
joint_states
magnetic_field
motor_power
odom
reset
rosout
rosout_agg
sensor_state
sound
tf

2. Question 2

The type of message being published to the 'odom' topic is the nav_msgs/Odometry

Some information that's in this message:
seq
timestamp (seconds and nanoseconds)
frame_id
pose (position xyz, orientation xyz)
pose covariance
twist (linear, angular)
twist covariance

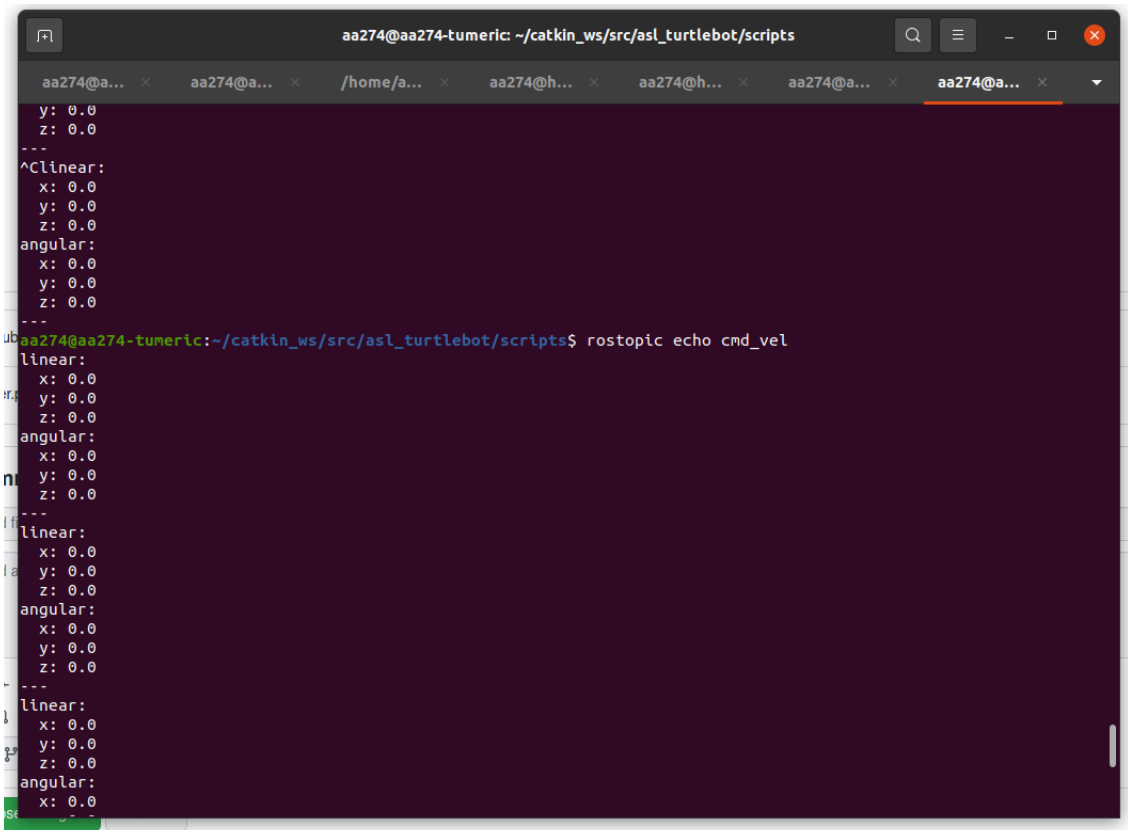
3. Question 3

Velocity command publisher code:

```
#!/usr/bin/env python3
```

```
import rospy
from geometry_msgs.msg import Twist

def publisher():
    pub = rospy.Publisher('/cmd_vel', Twist, queue_size=5)
    rospy.init_node('cmd_zero_vel_node', anonymous=True)

    while not rospy.is_shutdown():
        twist = Twist()
        twist.linear.x = 0
        twist.linear.y = 0
        twist.linear.z = 0
        twist.angular.x = 0
        twist.angular.y = 0
        twist.angular.z = 0
        pub.publish(twist)

    pass

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

Example of velocity command publisher output:



```
  y: 0.0
  z: 0.0
---
^Clinear:
  x: 0.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0
---
aa274@aa274-tumeric:~/catkin_ws/src/asl_turtlebot/scripts$ rostopic echo cmd_vel
linear:
  x: 0.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0
---
linear:
  x: 0.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0
---
linear:
  x: 0.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
```

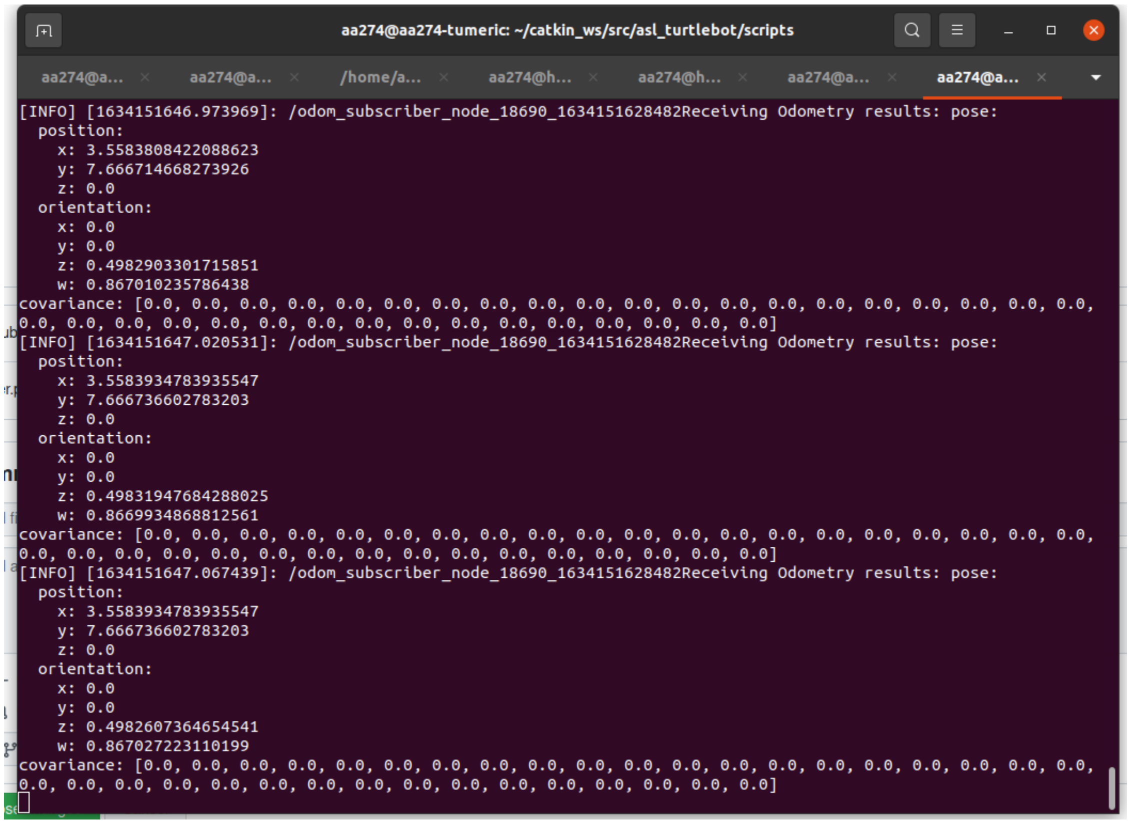
Figure 1: Output from Velocity Command (all zeros)

4. Question 4

Code for odometry message subscriber:

```python
#!/usr/bin/env python3

import rospy
from nav_msgs.msg import Odometry

def callback(data):
    #import pdb;pdb.set_trace()
    rospy.loginfo(rospy.get_caller_id() + "Receiving Odometry results: {}".
    format(data.pose))

def subscriber():
    rospy.init_node('odom_subscriber_node', anonymous=True)
    rospy.Subscriber('/odom', Odometry, callback)
    rospy.spin()

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

Example output from our subscriber of Odometry message Pose sub-structure:



```
[INFO] [1634151646.973969]: /odom_subscriber_node_18690_1634151628482Receiving Odometry results: pose:
  position:
    x: 3.5583808422088623
    y: 7.666714668273926
    z: 0.0
  orientation:
    x: 0.0
    y: 0.0
    z: 0.4982903301715851
    w: 0.867010235786438
covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[INFO] [1634151647.020531]: /odom_subscriber_node_18690_1634151628482Receiving Odometry results: pose:
  position:
    x: 3.5583934783935547
    y: 7.666736602783203
    z: 0.0
  orientation:
    x: 0.0
    y: 0.0
    z: 0.49831947684288025
    w: 0.8669934868812561
covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[INFO] [1634151647.067439]: /odom_subscriber_node_18690_1634151628482Receiving Odometry results: pose:
  position:
    x: 3.5583934783935547
    y: 7.666736602783203
    z: 0.0
  orientation:
    x: 0.0
    y: 0.0
    z: 0.4982607364654541
    w: 0.867027223110199
covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
```

Figure 2: Subscriber output of Odometry readings