AA274A Section 8
Writeup

Josie Oetjen, Karthik Pythireddi, Gerry Della Rocca

Problem 1: To which topic are the robot's control input sent to? Hint: use the commands rostopic list, rosnode list, rosnode info, and rqt graph. What node is currently publishing these control inputs? In your writeup, include the commands that you run to check this, and their output.

The cmd_vel is the control input topic. Navigator is the node that publishes it.

Rostopic list
/camera/camera_info
/camera/image_raw
/camera/image_raw/compressed
/camera/image_raw/compressed/parameter_descriptions
/camera/image_raw/compressed/parameter_updates
/camera/image_raw/compressedDepth
/camera/image_raw/compressedDepth/parameter_descriptions
/camera/image_raw/compressedDepth/parameter_updates
/camera/image_raw/theora
/camera/image_raw/theora/parameter_descriptions
/camera/image_raw/theora/parameter_updates
/camera/parameter_descriptions
/camera/parameter_updates
/clicked_point
/clock
/cmd_nav
/cmd_smoothed_path
/cmd_smoothed_path_rejected
/cmd_vel
/controller/alpha
/controller/delta
/controller/rho
/gazebo/link_states
/gazebo/model_states
/gazebo/parameter_descriptions
/gazebo/parameter_updates
/gazebo/performance_metrics
/gazebo/set_link_state
/gazebo/set_model_state
/imu
/initialpose
/joint_states
/map

AA274A Section 8
Writeup

Josie Oetjen, Karthik Pythireddi, Gerry Della Rocca

/map_metadata
/map_updates
/move_base_simple/goal
/navigator/parameter_descriptions
/navigator/parameter_updates
/odom
/planned_path
/rosout
/rosout_agg
/rviz/compressed/parameter_descriptions
/rviz/compressed/parameter_updates
/scan
/tf
/tf_static
/turtlebot3_slam_gmapping/entropy

Rosnode list
/gazebo
/gazebo_gui
/goal_commander
/navigator
/robot_state_publisher
/rosout
/rviz
/turtlebot3_slam_gmapping

Rosnode info /navigator

group05@genbu:~$ rosnode info /navigator

Node [/navigator]
Publications:
 * /cmd_smoothed_path [nav_msgs/Path]
 * /cmd_smoothed_path_rejected [nav_msgs/Path]
 * /cmd_vel [geometry_msgs/Twist]
 * /controller/alpha [std_msgs/Float32]
 * /controller/delta [std_msgs/Float32]
 * /controller/rho [std_msgs/Float32]
 * /navigator/parameter_descriptions [dynamic_reconfigure/ConfigDescription]
 * /navigator/parameter_updates [dynamic_reconfigure/Config]
 * /planned_path [nav_msgs/Path]

AA274A Section 8
Writeup

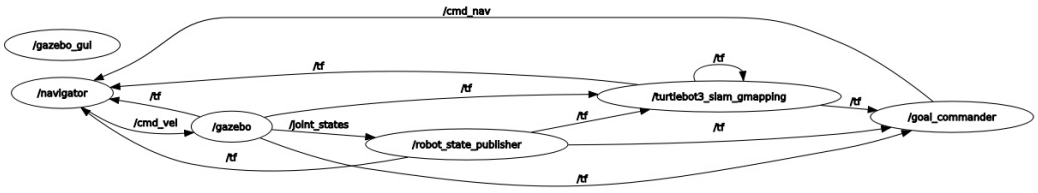
Josie Oetjen, Karthik Pythireddi, Gerry Della Rocca

```
 * /rosout [rosgraph_msgs/Log]

Subscriptions:
 * /clock [rosgraph_msgs/Clock]
 * /cmd_nav [geometry_msgs/Pose2D]
 * /map [nav_msgs/OccupancyGrid]
 * /map_metadata [nav_msgs/MapMetaData]
 * /tf [tf2_msgs/TFMessage]
 * /tf_static [tf2_msgs/TFMessage]

Services:
 * /navigator/get_loggers
 * /navigator/set_logger_level
 * /navigator/set_parameters


contacting node http://genbu.stanford.edu:39065/ ...
Pid: 1353826
Connections:
 * topic: /rosout
    * to: /rosout
    * direction: outbound (46173 - 172.24.66.14:57736) [9]
    * transport: TCPROS
 * topic: /cmd_vel
    * to: /gazebo
    * direction: outbound (46173 - 172.24.66.14:57744) [24]
    * transport: TCPROS
 * topic: /clock
    * to: /gazebo (http://genbu.stanford.edu:43045/)
    * direction: inbound
    * transport: TCPROS
 * topic: /tf
    * to: /robot_state_publisher (http://genbu.stanford.edu:37407/)
    * direction: inbound
    * transport: TCPROS
 * topic: /tf
    * to: /turtlebot3_slam_gmapping (http://genbu.stanford.edu:40791/)
    * direction: inbound
    * transport: TCPROS
 * topic: /tf
    * to: /gazebo (http://genbu.stanford.edu:43045/)
    * direction: inbound
```

AA274A Section 8
Writeup

Josie Oetjen, Karthik Pythireddi, Gerry Della Rocca

  * transport: TCPROS
* topic: /tf_static
  * to: /robot_state_publisher (http://genbu.stanford.edu:37407/)
  * direction: inbound
  * transport: TCPROS
* topic: /map
  * to: /turtlebot3_slam_gmapping (http://genbu.stanford.edu:40791/)
  * direction: inbound
  * transport: TCPROS
* topic: /map_metadata
  * to: /turtlebot3_slam_gmapping (http://genbu.stanford.edu:40791/)
  * direction: inbound
  * transport: TCPROS
* topic: /cmd_nav
  * to: /goal_commander (http://genbu.stanford.edu:46563/)
  * direction: inbound
  * transport: TCPROS



Problem 2: What command would you use to control the robot using your keyboard? Hint: Take a look at Section 3 and file /catkin ws/src/turtlebot3/turtlebot3 teleop/nodes/turtlebot3 teleop key. We do not suggest the one in /asl turtlebot/scripts which is extremely hard to use.

roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch

We have the ‘cmd_vel’ topic which is published by the publisher, through which we modify the linear and angular velocities using the below keyboard controls.

AA274A Section 8
Writeup

Josie Oetjen, Karthik Pythireddi, Gerry Della Rocca

Control Your TurtleBot3!

Moving around:
```
      w
  a   s   d
      x
```

w/x : increase/decrease linear velocity (Burger : ~ 0.22, Waffle and Waffle Pi : ~ 0.26)
a/d : increase/decrease angular velocity (Burger : ~ 2.84, Waffle and Waffle Pi : ~ 1.82)

space key, s : force stop

CTRL-C to quit

Problem 3: Propose a method to efficiently switch from the controller used in the previous section (i.e. with the navigator) to the teleoperated controller. Different methods are possible.

Hint: One method is to write a node that subscribes to both control topics (one from navigator.py, and one from the teleop controller) and publishes to the topic with the control inputs that are sent to the robot.

We go with the hint and proceed as follows:

We would first modify navigator.py to publish to /cmd_vel_nav instead of /cmd_vel and teleoperator to publish to /cmd_vel_tel instead of /cmd_vel. We can then create a new file multiplexer.py which subscribes to both /cmd_vel_nav and /cmd_vel_tel and publishes the selected option to /cmd_vel.

Problem 4: Implement your idea. For now, you can choose which controller is in charge by hard-coding some parameter. The rest of this section will look at different ways to switch between controllers. Make sure that control commands are only published to the robot from one source.

```
#!/usr/bin/env python3
import rospy
import time
from geometry_msgs.msg import Twist
from nav_msgs.msg import Odometry

class Switch:
```

AA274A Section 8
Writeup

Josie Oetjen, Karthik Pythireddi, Gerry Della Rocca

```
    def __init__(self):
        rospy.init_node('switch_node',anonymous=True)
        self.pub=rospy.Publisher('cmd_vel',Twist,queue_size=10)
        self.sub1=rospy.Subscriber("cmd_vel_nav", Twist, self.callback_nav)
        self.sub2=rospy.Subscriber("cmd_vel_tel", Twist, self.callback_tel)
        #Switching variable
        self.switch=1

    def callback_nav(self,msg: Twist):
        if self.switch:
            self.pub.publish(msg)


    def callback_tel(self,msg: Twist):
        if not self.switch:
            self.pub.publish(msg)

def subscriber():
    switch=Switch()
    rospy.spin()


if __name__ == '__main__':
    try:
        subscriber()
    except rospy.ROSInterruptException:
        Pass
```

Problem 5 : Write a launch file that starts both the main software stack (project.launch) and your backup keyboard controller

```
<launch>
  <arg name="sim" default="true"/>

  <include file="$(find asl_turtlebot)/launch/root.launch">
    <arg name="world" value="project_city" />
    <arg name="x_pos" default="3.15"/>
    <arg name="y_pos" default="1.6"/>
    <arg name="z_pos" default="0.0"/>
    <arg name="rviz" default="section4"/>
    <arg name="model" default="asl_turtlebot"/>
```

AA274A Section 8
Writeup

Josie Oetjen, Karthik Pythireddi, Gerry Della Rocca

```
  <arg name="sim" default="$(arg sim)"/>
 </include>
 <param name="switch" value="1"/>
 <node pkg="asl_turtlebot" type="navigator.py" name="navigator" output="screen" />
 <node pkg="asl_turtlebot" type="switching_controller.py" name="switching_controller"
output="screen" />
</launch>
```

Problem 6.1: Make it possible to switch between controllers by publishing a message to a ROS topic. Test this out on the command line by using

Problem 6.2: Make the parameter that enables switching between controllers available as a ROS parameter. It should be visible when running the command rosparam list and can be set using the rosparam set command.

```
Rosparam list
Rosparam set switch 0
Rosparam get switch
#!/usr/bin/env python3
import rospy
import time
from geometry_msgs.msg import Twist
from nav_msgs.msg import Odometry

class Switch:
   def __init__(self):
      rospy.init_node('switch_node',anonymous=True)
      self.pub=rospy.Publisher('cmd_vel',Twist,queue_size=10)
      self.sub1=rospy.Subscriber("cmd_vel_nav", Twist, self.callback_nav)
      self.sub2=rospy.Subscriber("cmd_vel_tel", Twist, self.callback_tel)

   def callback_nav(self,msg: Twist):
      switch=rospy.get_param("switch",1)
      if switch:
         self.pub.publish(msg)


   def callback_tel(self,msg: Twist):
      switch=rospy.get_param("switch",1)
      if not switch:
```

AA274A Section 8
Writeup

Josie Oetjen, Karthik Pythireddi, Gerry Della Rocca

```
        self.pub.publish(msg)

def subscriber():
    switch=Switch()
    rospy.spin()


if __name__ == '__main__':
    try:
        subscriber()
    except rospy.ROSInterruptException:
        Pass
```