Gregory Cooke

Partner: Connor Napolitano

Learning Module 8: Autonomous Vehicles/Navigation

Module 1: Wandering

Week 3: A Safety Finite Machine

Homework Question: You should roughly reiterate the activity, discuss how it was accomplished, demonstrate through images (if possible) the functionality, then briefly note any observations (including cases where it may not work so faithfully, or what is needed to work very well):

In this activity we paired basic sensing and actuation together. The end goal was to create a state machine that would move forward until either a bumper was hit or the robot was lifted off the ground. Then, once that condition (hit bumper or lifted robot) had been corrected, the robot was to wait two seconds then being moving forward again. To do this we made use of the Kobuki Bumper Events and the Kobuki Wheel Drop Events. When one of those sensor is hit or dropped in the Kobuki base, it sends a ROS message that we can interpret to know what happened to the robot. Then, based off of those messages, we can change the state of the state machine to where it is supposed to go.

To accomplish this, we change a variable, bhit, in the callback functions for the /mobile_base/events/bumper topic and the /mobile_base/events/wheel_drop topics. This bhit variable was a 5-bit binary number which was defined as follows: Left Bumper | Center Bumper | Right Bumper | Left Wheel | Right Wheel. A 1 meant the sensor was in the "hit" state for the bumpers or the "lowered" state for the wheels, whereas a 0 meant the sensor was in the "unhit" state for the bumpers or the "raised" state for the wheels. The callback functions changed the individual bit in this number based on the message received.

The state machine had 3 states – 0, 1, and 2. The 0 state is the robot moving forward continuously. The 0 state moved to the one state when the bhit variable had a value greater than 0, because that means that at least one of the sensors had been hit. The state machine then moved to the last state after bhit was zero again, then waited two seconds and proceeded to state 0.

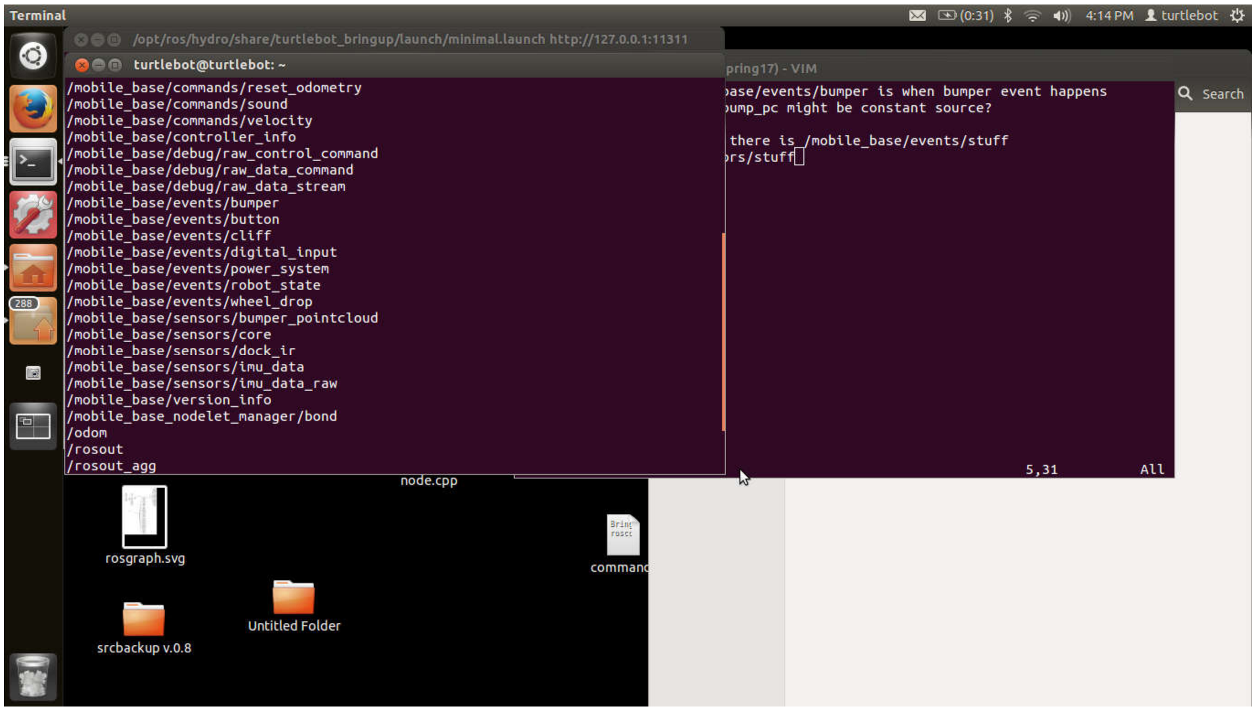
We ran into difficult in one specific place with this week's assignment – we were using the "and" and "or" keywords in Python to do our bitwise manipulations. They were not acting as we expected them too, thus we found out that the bitwise binary operators in Python are actually "|" and "&". Our original callbacks also, if a bumper was released or a wheel raised, would set the whole bhit value to 0. This caused an issue where, if the robot was lifted then had a bumper hit while it was still in the air, it would move forward before being put down. This is because the bumper callback was resetting the wheel values to 0 even though they had not been lowered. This was fixed by changing to doing bitwise operations that only changed the specific bumper or wheel that the callback happened for.

go_forward_bumpers_wheels.py has been included to show the code that we ran for our demo.

Turtlebot Adventures Answers:

1. What are the ROS topics that must be subscribed to in order to get these sensor measurements?

    a. The ROS topics are the /mobile_base/(events or sensors/* topics in the below image.



```
/mobile_base/commands/reset_odometry
/mobile_base/commands/sound
/mobile_base/commands/velocity
/mobile_base/controller_info
/mobile_base/debug/raw_control_command
/mobile_base/debug/raw_data_command
/mobile_base/debug/raw_data_stream
/mobile_base/events/bumper
/mobile_base/events/button
/mobile_base/events/cliff
/mobile_base/events/digital_input
/mobile_base/events/power_system
/mobile_base/events/robot_state
/mobile_base/events/wheel_drop
/mobile_base/sensors/bumper_pointcloud
/mobile_base/sensors/core
/mobile_base/sensors/dock_ir
/mobile_base/sensors/imu_data
/mobile_base/sensors/imu_data_raw
/mobile_base/version_info
/mobile_base_nodelet_manager/bond
/odom
/rosout
/rosout_agg
```

```
ase/events/bumper is when bumper event happens
ump_pc might be constant source?

 there is /mobile_base/events/stuff
rs/stuff
                                          5,31          All
```

The /mobile_base/events/bumpers and the /mobile_base/events/wheel_drop topic were subscribed to in order to create the finite state machine. An example that we didn't use is the gyroscopic information, and that is obtained via the /mobile_base/sensors/imu_data topic.

2. Which of these publish constant data, and which only publish as needed
    a. The /mobile_base/events/ topics are only published to when something happens, such as a bumper being hit or released, or a wheel being raised or dropped. The /mobile_base/sensors/ topics are published to continuously with data.

3. How many different bumpers does the kobuki have?
    a. The kobuki has a left, center, and right bumper. They are all somewhat attached physically so that they all move sort of together, but hitting the on physical bumper does not trigger each side.