CS378 - Autonomous Vehicles in Traffic
Spring 2012

Programming Assignment 2 (15 Points)
Due Date: Wednesday 2/22/12 11:59 PM

Instructions
- It is important that you work on this assignment independently as it will serve as a building block for the rest of the course.
- Ask the teaching staff first if you have any problems. It is OK to discuss the assignment with your friends and classmates as long as you only discuss programming concepts.
  
  Do not copy code from them!
- The assignment can be done in C++ or Python. The extra credit section is C++ only.
- The standard late policy as detailed in the syllabus will be followed
- Remember to follow the submission instructions at the end of the assignment

In this assignment, you will try and re-implement your code from programming assignment 1 as nodes within the ROS framework. You will have to perform the following steps:
1) Create a new ROS package called p2_<first_name>_<last_name> in the spr12 stack
2) Implement 2 nodes inside the package, called heating_unit and bang_bang_control.
A sample is provided in the ROS package p2_sample that demonstrates the file structure you should use for this assignment.

Inside the heating_unit node there should be 2 variables; current_temperature and heating_unit_on. Give these variables the default values of 70 and false.

The heating_unit node subscribes to the following messages:
1) A std_msgs/Empty message on the topic ‘turn_on’ - Turn the heating unit on when this message is received.
2) A std_msgs/Empty message on the topic ‘turn_off’ - Turn the heating unit off when this message is received.

The heating_unit node publishes the following message:
1) A std_msgs/Int32 message on the topic ‘current_temp’ - Publish current_temperature at 1Hz (i.e., once a second) using a rate loop as in the example in class. Increment/Decrement the temperature using the same policies as Assignment 1. (+1 when on, -1 when off, limited between 0 and 100). Every time you publish a message, print the following message on the screen using the ROS_INFO macro:
Heating Unit is currently ON (or OFF). Current temperature is XX.
The bang_bang_control node should have a variable to keep track of the current temperature being maintained (default value = 66).

The bang_bang_control node subscribes to the following messages:
1) A std_msgs/Int32 message on the topic ‘set_maintained_temp’ - Set the maintained temperature to value encapsulated by the message when it is received.
2) A std_msgs/Int32 message on the topic ‘current_temp’ - This message contains the current temperature. Perform the same checks as programming assignment 1 here:
   if (current_temp > temp_to_maintain + 2) {
     // publish std_msgs/Empty message on 'turn_off' topic
   } else if (current_temp < temp_to_maintain - 2) {
     // publish std_msgs/Empty message on 'turn_on' topic
   }

The bang_bang_control node publishes the following message:
1) A std_msgs/Int32 message on the topic ‘current_maintained_temp’ - Publish the current temperature being maintained on this topic. Instead of using a rate loop, publish this message every time you receive a message on the topic current_temp. Print the following message to the screen using the ROS_INFO macro when you publish this message:
   I am maintaining the temperature XX
2) A std_msgs/Empty message on the topic ‘turn_on’ - described above.
3) A std_msgs/Empty message on the topic ‘turn_off’ - described above.

Testing your code:
Build your code using the rosmake or make command. Run the following commands in separate windows (or tabs):
   roscore
   rosrun <your_package_name> heating_unit
   rostopic echo /current_temp
   This should start displaying the current temperature. Since you do not have the controller on, and defaulted to having the heating unit being off, the temperature should be steadily dropping.
Run the following command in a separate window to start the controller:
   rosrun <your_package_name> bang_bang_control
   You should see the temperature stabilizing in the terminal where you were printing the current temperature. To see the temperature being currently maintained, run the following command:
   rostopic echo /current_maintained_temp
   Finally, to change the maintained temperature, run the following command:
   rostopic pub /current_maintained_temp std_msgs/Int32 XX
   You should see the current temperature changing appropriately.
Submission instructions:
Make sure you have commit permission to our repository. Send an email to cs378-spr12-submit@utlists.utexas.edu to request it. To submit your code, you first need to remove all the build files that were generated when you compiled your code. Run the following commands:

```bash
roscd <your_package_name>
make clean
roscd spr12
svn add <your_package_name>
svn ci <your_package_name> -m "checking in my 2nd programming assignment"
```

Extra Credit (3 Points):
In this section, you will reconstruct the first part of the assignment through the use of libraries. There is a particular file structure that you should follow when working with header files in a ROS package. A sample package with some helpful comments named p2extra_sample has been provided.

Create a new package called p2extra_<first_name>_<last_name>. For the following steps, use p2extra_sample as a guideline.

1) Inside this package, you need to place the header files in the include/p2/ directory and the implementation in the src/impl/ directory. The nodes themselves need to reside in the src/ directory.
2) To include the header use `#include <p2/HeatingUnit.h>` or `#include <p2/BangBangControl.h>`. Follow the samples in p2extra_sample for this.
3) You can reuse your code from Programming Assignment 1 to write the headers and the implementations. I was able to use my HeatingUnit class and implementation code as it is. I had to make some changes to the BangBangControl class for this assignment.
   - You will need to remove the HeatingUnit object, and change the constructor accordingly.
   - The update() function should return the enumeration type ControllerAction instead of void. This enumeration type has been defined in the sample BangBangController.h.
   - The update() function should take the current temperature as an input argument.
   - All old calls to the HeatingUnit object inside the update function are now handled by ROS, as a result return the enumeration object with what action to take (NOOP - No operation, TURN_ON or TURN_OFF). This is required as you cannot tell the heating unit to perform an action from inside the BangBangControl class. Instead you need to publish the appropriate messages in the bang_bang_control node. Read the enumeration object when you call the update() function from the bang_bang_control node and publish the appropriate message.
4) Start with the existing code for the node that you created for the first part of this assignment, and use a HeatingUnit object in the heating_unit node instead of creating the variables for temperature and the status of the heating unit. Do the same for the bang_bang_control node.

You should be able to use the same instructions for testing your code and submitting it.