Autonomous Cars Hackathon : Level 0/1 and beyond

Step 1 has been shared before and contains the most important installation steps. We hope that everyone has completed Step 1. Now, please go through Step 2 for further basic setup requirements

Step 1
Installation instructions for Ros and Gazebo:

1. Log into your Ubuntu 16.04 system, and open the terminal (Ctrl+Alt+T by default)
2. We first need to install RoS, which stands for Robot Operating System onto our machines. This software will help us completely control the robot in both simulated as well as real world environments.
3. Please follow the instructions given at http://wiki.ros.org/kinetic/Installation/Ubuntu to install RoS onto your Ubuntu system. Do not forget to install the dependencies given at the end. You just have to copy the command, paste it in your terminal window and press enter each time. Do this for all the commands given on the page.
   Note: **Be careful to install just the FULL DESKTOP version of RoS** (1st option under the 1.4 Installation section on the page). Do not copy any other commands in this section.
4. After RoS has been installed, we need to install Gazebo for RoS. This is a simulator which can be used to visualize the movements of our robot.
   Turtlebot, which is the what our robot is called, already has a packages available for download and install. This package will enable us to visualize and play with the turtlebot in its simulated environment.
   To install the packages type the following in your terminal (Again, Ctrl+Alt+T by default):
   sudo apt-get install ros-kinetic-turtlebot-* -y
5. That's it! You have successfully finished installing RoS and Gazebo onto your Ubuntu 16.04 system. You can explore the Gazebo environment if you like. Just open up a terminal and type "gazebo". The simulator should start up an empty world which you can modify yourself for fun.

Step 2 (Total Time to set up ~20 minutes)

Please follow the instructions below to upgrade your packages so that they work coherently with our code.

1. First, you will need to upgrade your turtlebot package to the latest version. Just open up a terminal window and type
   sudo apt-get install ros-kinetic-turtlebot3-* -y
   This will install all the required packages in your system to run the code that we will provide for you. After this just type "**sudo updatedb**" to update the cache on the system with the package. To check if this step worked type the following in your terminal:

roscd turtlebot3-

And press tab 2 or more times. The turtlebot3 packages should show up. In case nothing shows up on pressing tab, restarting your computer will resolve the issue.

2. Next, we need to upgrade Gazebo to the latest compatible version, i.e. 7.12.0. Please follow the **alternate instructions** (**DO NOT** run the automatic setup with the curl command) given at:
   http://gazebosim.org/tutorials?cat=install&tut=install_ubuntu&ver=7.0.
   Go line by line and continue pasting commands in your terminal.
3. Once the update is complete type "**gazebo --version**" into your terminal and press enter. If the version number displayed is 7.12.0 or higher, the update was successful and you can proceed to the next step.
4. Adding the models to gazebo:
   It is necessary for you to add the models that we have provided you with to the right path so that everything functions smoothly. The model files basically define the properties of the objects you can view in the Gazebo environment.
   The instructions below will guide you through this process.
   a. Navigate to the hackathon folder provided.
   b. Open the models directory.
   c. Select everything in the directory (Ctrl + A), right click and press copy.
   d. Browse back to your home folder.
   e. Press "Ctrl + H" to view hidden files and folders.
   f. Open a hidden folder named ".gazebo"
   g. Navigate into the models folder.
   h. Right click, and press paste in the models folder. All the models are now ready to be loaded and used in the Gazebo environment.
5. Now, let's set up the whole package after which we can run our codes and the launch files
   a. Open a Terminal (Ctrl+Alt+T)
   b. Install Catkin by running the following commands sequentially:
      i. **sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu `lsb_release -sc` main" > /etc/apt/sources.list.d/ros-latest.list'**
      ii. **wget http://packages.ros.org/ros.key -O - | sudo apt-key add -**
      iii. **sudo apt-get update**
      iv. **sudo apt-get install python-catkin-tools**
      v. Installation is done. Run **catkin --version** to check if it has been installed successfully. You should get a version not older than 0.4.4
   c. Make a catkin directory to build our package:
      i. **cd ~/**
      ii. **mkdir catkin_workspace**
      iii. **cd catkin_workspace**
      iv. **mkdir src**
      v. **cd src**
      vi. Run **catkin_init_workspace**
      vii. Copy the hackathon folder inside src folder

viii. After copying hackathon inside src: go inside the folder
cd hackathon

ix. Run **cmake**

x. **cd ../..** (You should be in catkin_workspace folder now)

xi. Run **catkin_make**. This should have built the project. But we still need to update paths

xii. Run **echo "source ~/catkin_workspace/devel/setup.bash" >> ~/.bashrc**

xiii. Run **source ~/.bashrc.** The setup process is now complete.

xiv. To check the setup run **roscd hackathon**, this should take you to the hackathon directly if the above steps were followed correctly.

xv. To further check if the launch files are working, run
roslaunch hackathon fin.launch
This should open up gazebo with the level 1 map loaded in it and turtlebot spawned near the start of the map. Note: When running this for the first time, it might take 2-5 minutes to load. Please be patient as this happens only 1-2 times and your simulations will load fast thereafter.

6. The code file where you will be coding all your modules is contained in the hackathon folder by the name "**movebot.py**". The file is very well commented and we request you to please go through the file and all its functions (especially laser_callback and image_callback) before starting. To run the file:
    a. Open Terminal
    b. roscd hackathon
    c. python movebot.py

    This will launch the map with the bot and deploy the code on it.
7. If you have reached this step successfully: CONGRATS, you are done with Level 0. Now, in the code provided, you will find a function called **threaded_image_handling**. This function takes the image seen by the bot and sets the global variable **turn** to the corresponding label: right/left/u/stop, etc. You can call your model in this function and thus integrate your model with the code. To check how your model performs, run "**movebot.py**" and track how the turtlebot moves.
8. The given code turns according to wall distance in front of bot. In cases where the wall is at the side, and in cases where wall is both at the side and front, you will have to use lidar data to judge at what distance should you start your turn. That is also an exercise left to you so that you can get a better understanding of Lidar data and it will help you move to level 2 also.

Details Regarding Next Levels

The next 2 levels include the following tasks:
Obstacle Avoidance
Dynamic Object Handling (Traffic Light)
Processing Multiple Signs seen in the same bot view image

Given 3 types of live data: image, odometry and 2D Lidar, there are multiple ways you can solve these tasks and we recommend that you brainstorm about the same and possibly implement one or more of the above tasks.
Also, we highly recommend reading about the very interesting and highly popular yolonet model and how it may be relevant in such a setting as that of an autonomous car.

Points to Note

1. In case someone finishes levels 0/1 early (before Saturday), we have provided a version (without dynamically changing traffic lights) of the full map too. All you need to do to access it is change the word “level1.world” on line 20 in file fin.launch (present inside hackathon/launch/ in catkin workspace) to “all_levels.world”.
2. Lidar data is read as a 2D laser scan. In laser_callback function, the msg.ranges list gives 360 values representing 2D depth at each degree. The 0th index represents the angle right in front of bot and the 359th index represents the angle 1 degree to the right of the 0th index and the 1st index represents the angle 1 degree to the left of the 0th index. The range of the Lidar is 3.5 meters (can be configured)) and beyond it it counts all distances as inf.
3. The bot movement can be only controlled via the velocity_publisher which controls the linear x/y/z and angular x/y/z speed of the car. Of these we are mostly interested only in linear x and angular z speeds.
4. Please go through all the functions and documentation in the code in order to fully understand and thus be able to tweak and change it.
5. If you are able to complete all the levels on the simulation, your team gets to run the code in a real world setting on a turtlebot.
6. The marking scheme is as follows:
   Level 0 + Level 1 = 110
   Level 2 = 40
   Level 3 = 50
7. Bonus: Can you control the bot speed dynamically according to surrounding?
8. More hints about what approach you can use for the next levels will be provided to you on the day of the Hackathon.