Steps for Connecting to ROS via Arduino and terminal

Step 1) Make Sure that the Arduino IDE is setup for the ROS communication. You can look under File > Examples and if you find ros_lib there, you are good otherwise follow these steps for setting up the IDE.

STEPS FOR SETTING UP ARDUINO IDE

1) Install the rosserial libraries by opening a terminal and running these steps :

   sudo apt-get install ros-indigo-rosserial-arduino

   sudo apt-get install ros-indigo-rosserial

2) Now go under the sketchbook folder where the Linux Arduino environment saves your sketches and execute these commands:

`

   cd <sketchbook>/libraries

   rm -rf ros_lib

   rosrun rosserial_arduino make_libraries.py .

3) Now restart your IDE and you should see the ros_lib under the Files > Examples.

   For more information and errors, you can follow this tutorial on ROS wiki page:

   http://wiki.ros.org/rosserial_arduino/Tutorials/Arduino%20IDE%20Setup

Step 2) For running the ROS subscribers and publishers via Arduino ,open up the terminal and launch the roscore in a new terminal window:

```
roscore
```

Step 3) Now in a new terminal, run the rosserial client application that forwards your Arduino messages to the rest of ROS. Make sure to use the correct serial port(here it is USB0) :

```
rosrun rosserial_python serial_node.py /dev/ttyUSB0
```

Step 4) Now your Arduino should be able to communicate with the ROS nodes on your computer. To view all the topics that are being published by the ROS and Arduino, Run this command into a new terminal:

```
rostopic list
```

Step 5) To listen or see the values being published by a specific topic(“actual_speed” being the name of the topic here) run this command in to the terminal:

```
rostopic echo actual_speed
```

Whenever this topic receives a value, it will publish it in the terminal window.

Step 6) To publish a value to the topic or generating a fake topic, you can use the following command in a new terminal :

```
rostopic pub -r 10 /cmd_vel geometry_msgs/Twist  '{linear:  {x: 0.1, y: 0.0, z: 0.0}, angular: {x: 0.0,y: 0.0,z: 0.0}}'
```

This command here publishes the value to the topic named cmd_vel and the message type to be geometry_msgs/Twist and at a frequency of 10 hertz.

For any help related to the creating publishers and subscribers in Arduino and for the rostopic, you can follow these tutorials:

http://wiki.ros.org/rosserial_arduino/Tutorials/Hello%20World

http://wiki.ros.org/rosserial_arduino/Tutorials/Blink

http://wiki.ros.org/rostopic

COMMON ERRORS WHILE CONNECTING TO ARDUINO

1) While Uploading the sketch, you may face trouble connecting to the arduino board as the serial communication is taking place between the ROS and arduino, make sure to stop all commands running in the terminals mentioned above. Once the code is uploaded, you need to run them again to start the communication once again.

2) Another common error is : avrdude: ser_open(): can't open device "/dev/ttyACM0": Device or resource busy

For removing this error, follow these steps:- 1) Open up a terminal window and run these commands in order : sudo adduser <username> dialout

sudo chmod a+rw /dev/ttyACM0

Here replace the <username> with your own username which you are using for the Linux environment and replace the port name ACM0 with the correct Port name which is being used.

For more information, you can visit this page: https://stackoverflow.com/questions/40951728/avrdude-ser-open-cant-open-device-dev-ttyacm0-device-or-resource-busy

Steps for creating fake publishers and getting Rosbags Data

Step 1:- Make a Catkin workspace and set it up for Publishing and subscribing to the messages. For more help on this, you can folllow the ROS tutorials:- http://wiki.ros.org/ROS/Tutorials/CreatingPackage

Step 2 :- Under your workspace package , add the ramp and sin cpp files in the src folder and make the changes in the CmakeLists file accordingly. See the above mentioned tutorials for making the changes.

Step 3:- Now open up the terminal and run roscreo.In a Separate terminal, you can run your publisher/ subscriber by running this command:-

rosrun beginner_tutorials talker

Here begineer_tutorials being the name of the package and talker be the name of the publisher file.

Step 4:- Now your publisher would be up and running, you can run the rostopic list

command in a new terminal and see the value of the topic being published.

Step 5:- For recording the Rosbag data, open up a new terminal and Run the command : rosbag record -a

For playing a recorded rosbag, open up terminal and run the command to play the "bagfilename" file with all the topics running.

rosbag play bagfilename.bag

This command will record all the topics being published. Press Ctrl+ C to stop the recording in the rosbag.

For more help on the Rosbag, you can see the following tutorial:-

http://wiki.ros.org/rosbag/Commandline

CONVERTING THE ANGULAR VELOCITY TO THE STEERING ANGLE

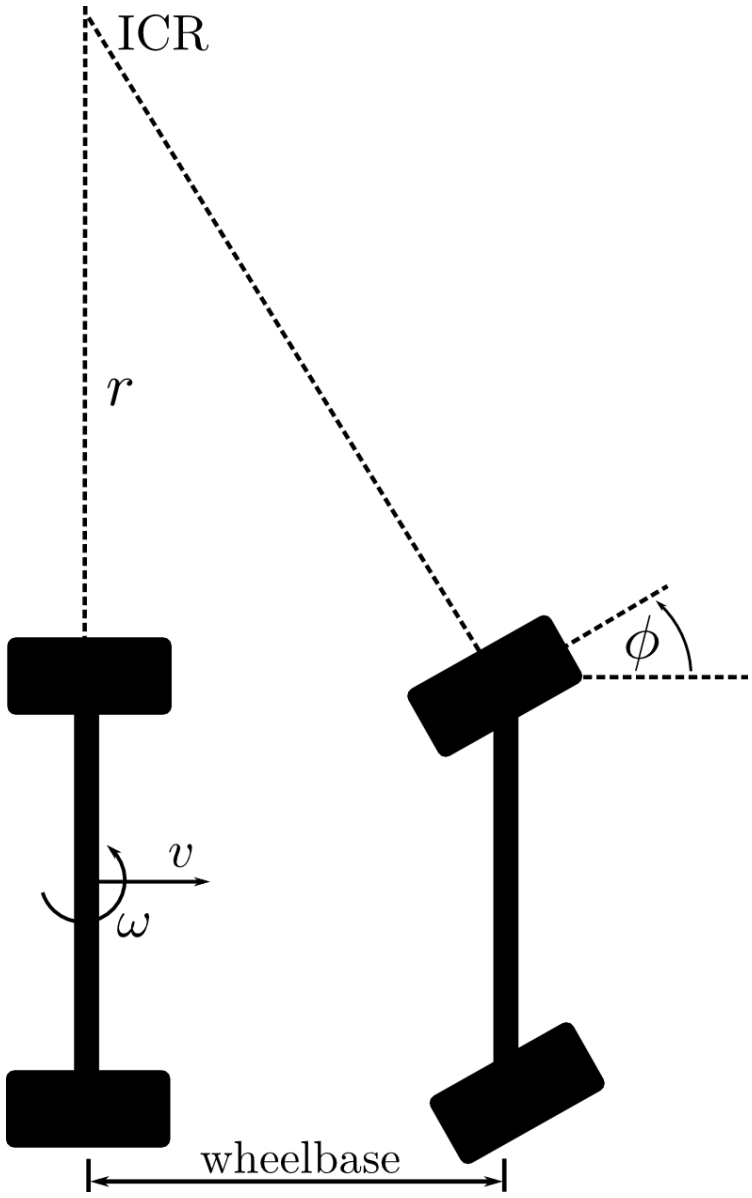
Translational and angular velocities

Our Planning module gives us the translational and angular velocities of the robot. In case of front-wheeled drives velocities are usually defined at the center of the rear axle. For this type, the published geometry_msgs/Twist message can be used directly.

Translational velocity and steering angle

The relevant variables including the translational velocity v and the steering angle phi are illustrated in the following figure. ICR denotes the instant centre of rotation.

For a vanishing angular velocity omega=0 the turning radius r tends to infinity which in turn leads to a zero steering angle phi=0. Otherwise the turning radius r might be computed by v/omega. Then, the steering angle is derived by phi=atan(wheelbase/r) (which indeed includes the first case). Note, the steering angle is not defined for v=0 using the equation introduced above. One could use the last known valid steering angle. However, in the following we assume that for a vanishing translational velocity the (desired) steering angle is set to zero.



So for our case, we calculated the wheelbase and did the mapping of the Steering angle with the wheel angle and we added a subscriber in the Arduino node which takes in the value of the angular velocity and taking the feedback from the Vectornav IMU, thus implementing the low level control on the angular velocity of the vehicle.

After getting the angular velocity from the Ros node, we made the corresponding changes as follows in the Arduino code:

radius = linear_velocity/angular_velocity; // calculates the radius of curvature of the robot

wheel_angle= atan(wheelbase/radius); // getting the wheel angle

desired_radians = wheel_angle*14.546; // converting wheel angle to radians

desired = desired_radians*57.2958; // converting the wheel angle to steering angle using the mapping done on the ground

PATH PLANNER- Using TEB PLANNER and ROS NAVIGATION

INSTALLING THE PACKAGES

Step 1:- Install ROS nodes required for the local and global planners, amcl, maps and motor control for the navigation stack.

```
$ sudo apt-get update
$ sudo apt-get install ros-kinetic-move-base
$ sudo apt-get install ros-kinetic-map-server
$ sudo apt-get install ros-kinetic-amcl
$ sudo apt-get install ros-kinetic-eband-local-planner
$ sudo apt-get install ros-kinetic-global-planner
$ sudo apt-get install ros-kinetic-teb-local-planner
$ sudo apt-get install ros-kinetic-teb-local-planner-tutorials
```

For more help in installing the teb planner follow this tutorial :
http://wiki.ros.org/teb_local_planner/Tutorials/Setup%20and%20test%20Optimization

Installing

Clone the repository in your catkin workspace 'src/' folder.

$ cd ~/catkin_ws/src/

Clone the repository here from either Github account or the bitbucket account.

Build the project:

$ cd ~/catkin_ws

$ catkin_make

If you haven't already, the following line can be added to your .bashrc to auto-source all new terminals

source ~/catkin_ws/devel/setup.bash

Run the Code

In a terminal window, type the following,

$ cd ~/catkin_ws

$ roslaunch skid_steer_bot udacity_world.launch

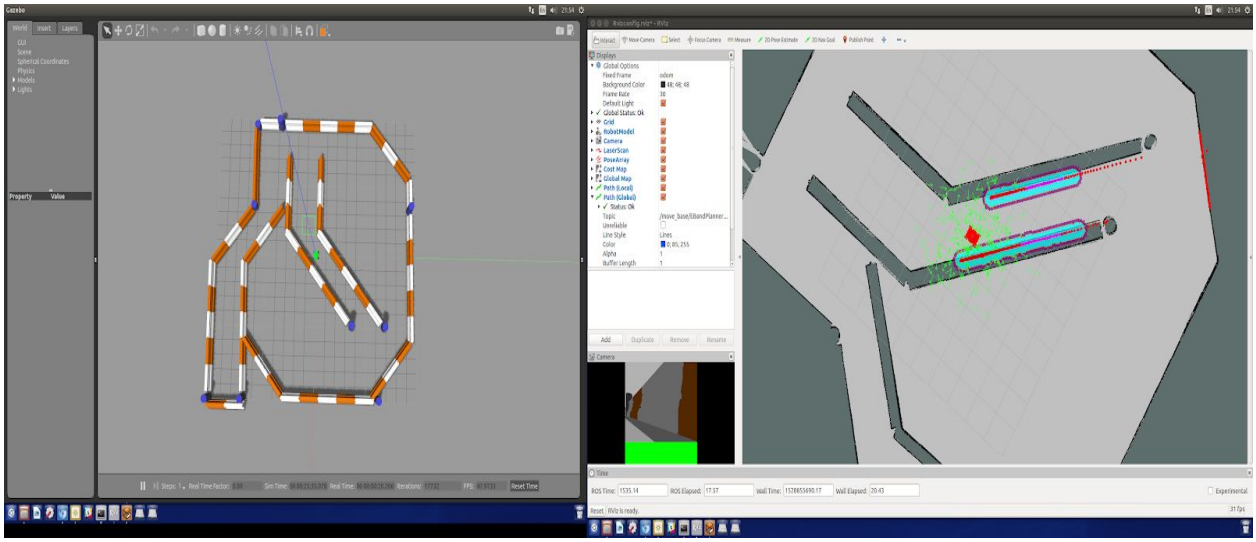
In a new terminal, run the 'amcl.launch' file.

$ cd ~/catkin_ws

$ source devel/setup.bash

$ roslaunch skid_steer_bot amcl.launch

Gazebo and Rviz will load and you should arrive at a result similar to the below.

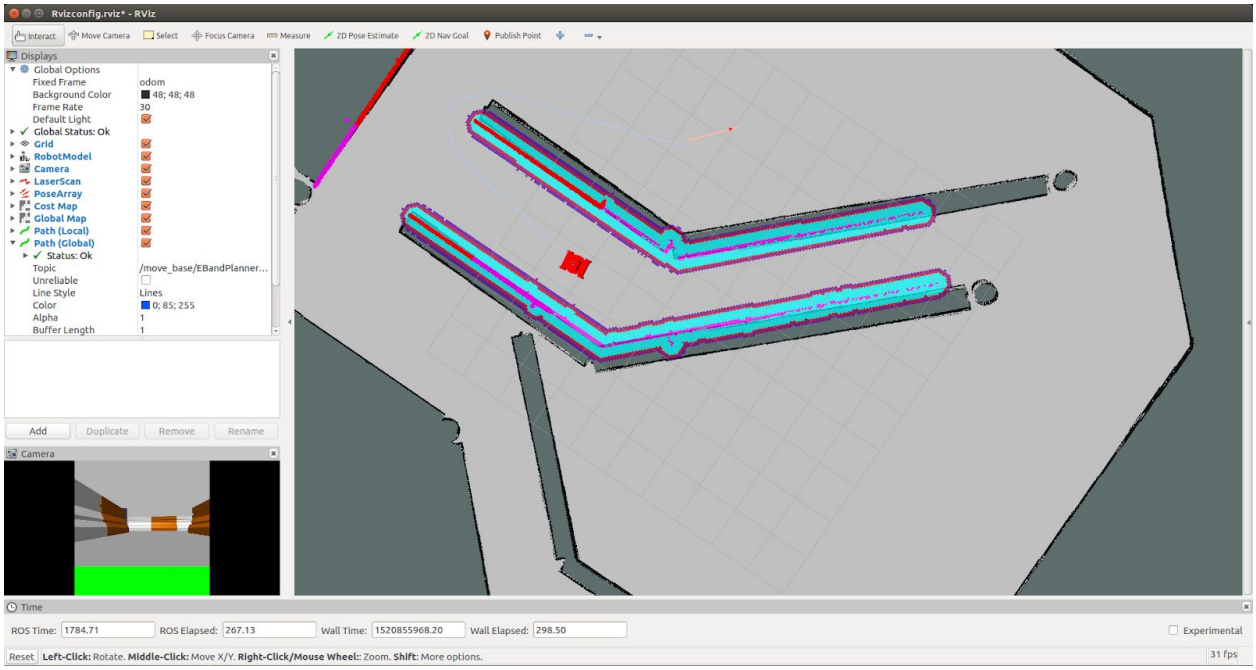


Testing

In Rviz, click on the 2D Nav Goal in the top menu.

Click on the Rviz map where you want the robot to navigate too.

You should arrive at a result similar to the below.



Creating a Robot Model in Gazebo

You can see how to create a model in gazebo by following these tutorials:

http://gazebosim.org/tutorials?tut=build_model

http://gazebosim.org/tutorials?tut=model_editor

You can see the robot model being created in the urdf folder in the repository

Creating a Virtual World in Gazebo

You can see a virtual world being created in the Worlds folder in the repository. You can open the file jackal_race.world to modify the changes in the world. Few of the tags that have been used are explained here.

1) Most of the links which are being used are the drc_practice_white_jersey_barrier for creating the obstacles in the world. You can change their positions and the collision behavior using the pose and the collision tabs in the code.

2) The other type of links used are the blue cylinders at the end of the links which can also be changed in the similar way as that of the above links.

3) For creating the effect of Dynamic obstacles, I have used the rectangular white boxes and imparted them a velocity in random directions. You can find them in the <model name="box2"> tabs. You can alter the size and the pose of them using the visual and pose tags. I have used the Random Velocity Plugin for imparting the boxes velocities in random directions.

For more help on creating the worlds in gazebo, you can refer to : http://gazebosim.org/tutorials?tut=build_world

Configuring the Parameters for the Planner

Configuring the parameters involved tuning and setting up a lot of different things in different files. Here are the parameters which can be varied according to the requirement:

1) Go to the launch Folder in the Package. There are three different launch files for launching all the nodes required for the simulation.

The Udacity_world.launch file is used to launch the environment and the robot in there with all the sensors i.e. Laser Scan and camera right now . Under this file, you will see

different node launch names which can be altered if you want to use a different world and a different model of the robot.

2) The robot_description.launch file is used to launch the model of the robot and setup fake joint values and send robot states to tf for the simulation to respond to the planner outputs.

3) THE AMCL.launch file is the most important file here. It contains all the launch and configuration files for the Map Server, placing map frame at odometry frame , Localization, transforms, laser parameters, Odometry Model Parameters and then the ROS parameters files for the global and the local planners.

For the Planner to be working, You need to have Rosparamter files for each one of these things:

```
 <rosparam file="$(find skid_steer_bot)/config/costmap_common_params.yaml" command="load" ns="global_costmap" />
   <rosparam file="$(find skid_steer_bot)/config/costmap_common_params.yaml" command="load" ns="local_costmap" />
   <rosparam file="$(find skid_steer_bot)/config/costmap_converter_params.yaml" command="load" />
   <rosparam file="$(find skid_steer_bot)/config/local_costmap_params.yaml" command="load" />
   <rosparam file="$(find skid_steer_bot)/config/global_costmap_params.yaml" command="load" />
   <rosparam file="$(find skid_steer_bot)/config/base_local_planner_params.yaml" command="load" />
```

For shifting from one Planner type to another for the local planner, just go to this line command : `<param name="base_local_planner" value="teb_local_planner/TebLocalPlannerROS" />` and change the value to any of the following you lwant to use : base_local_planner/TrajectoryPlannerROS,

eband_local_planner/EBandPlannerROS, teb_local_planner/TebLocalPlannerROS

You can also change the Planner and the Controller frequency from these commands :

```
<param name="controller_frequency" value="30.0" />
   <param name="planner_frequency" value="10.0" />
```

For more help on understanding and learning of Launch files, you can go through this tutorial :
http://wiki.ros.org/roslaunch/Tutorials/Roslaunch%20tips%20for%20larger%20projects

For the Teb Planner Parameters Setup, go to the Config file in the Repository and there are five different yaml files there to setup the Parameters.

Under the Global_costmap_params.yaml and Local_costmap_params. Yaml, we have the parameters for the global and local costmaps which include the update frequency and the publish frequency of the maps as well as the height and width of the map you want to use in metres.

Under the Costmap_common_params.yaml, we have the parameters for the costmap such as the obstacle range ,raytrace range , footprint and transform tolerance.

The MOst important of all is the base_local_palnner_params.yaml which contains all the parameters for the local planner making it follow the constraints of the cart.
In this, we have several tabs starting with a # symbol .
1) Under the Trajectory tab, you can alter the parameters like teb_autosize: True
dt_ref, dt_hysteresis, global_plan_overwrite_orientation, allow_init_with_backwards_motion, max_global_plan_lookahead_dist. Make sure you select appropriate lookahead distance for the planner to be working good. Too less or large values can lead to the slow motion of the robot.

2) Under the Robot tab, You can alter the maximum velocity of the robot in the forward and the backward direction by the max_vel_x and max_vel_x_backwards parameters. You can also change the maximum angular velocity and acceleration limits on both velocity and angular velocity.
Make sure you are using correct value of angular velocity as the angular velocity is also bounded by min_turning_radius in case of a car like robot (r = v / omega).

3) Under the car like robot parameters, you can alter the minimum turning radius of the vehicle and select the wheelbase and use the option to take command angle instead of angular velocity.

4) in the footprint tab, you can change the type of the footprint you want to use for your robot. You can select it to be a point or a line or a polygon. Selecting the polygon is the best way to use, you can specify the coordinates of the robot in the polygon by measuring the exact dimensions of the golf cart.

5) under the Goal Tolerance and the Obstacles tab, you can alter the minimum obstacle distance and the weight values of the velocities and the accelerations.

Installing the Vectornav package

This assumes that you have a VectorNav device connected to your computer via a USB cable and that you have already created a catkin workspace.

Build:

```
cd ~/catkin_ws/src
git clone https://github.com/dawonn/vectornav.git
cd ..
```

catkin_make

Run:

(Terminal 1) roscore

(Terminal 2) roslaunch vectornav vectornav.launch

(Terminal 3) rostopic list

(Terminal 3) rostopic echo /vectornav/IMU

(Terminal #) ctrl+c to quit

Overview

vnpub node

This node provides a ROS interface for a vectornav device. It can be configured via ROS parameters and pubishes sensor data via ROS topics.

vectornav.launch

This launch file contains the default parameters for connecting a device to ROS. You will probably want to copy it into your own project and modify as required.