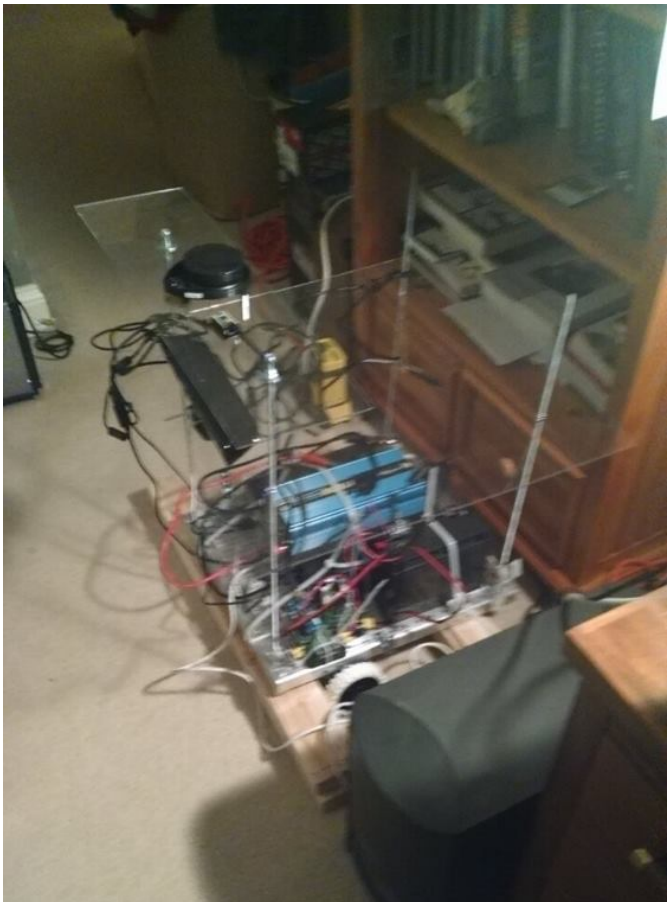
RoveME, the ROS based Autonomous Navigation Robot

by Trevor Sherrard



Electrical Hardware

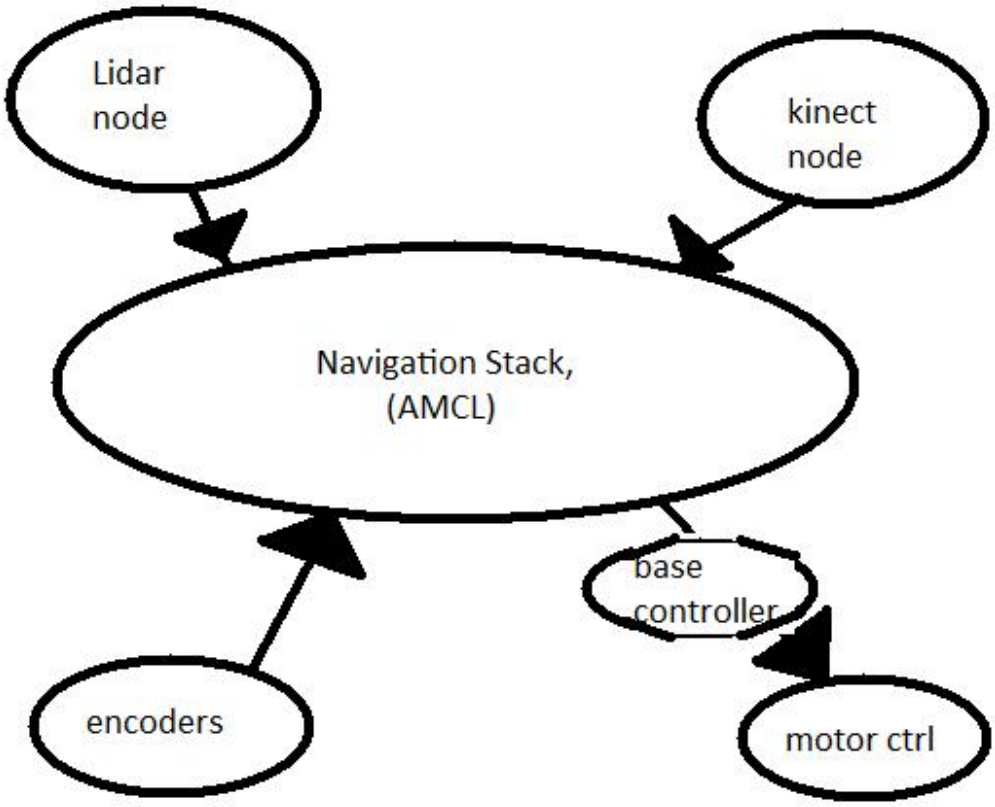
RoveME uses a standard micro ITX desktop computer as a controller. This computer is powered by a 1100 watt inverter coming off the battery. The battery also powers the Talon motor controllers that receive commands via an arduino connected to the PC. There is also a NeatoXV LIDAR onboard. This is powered by a board that also transmits the data from the LIDAR to the PC. The microsoft kinect provides RGBD images to aid in navigation. Encoders on the wheels allow for feedback for the ROS navigation stack to transform the robot into the odom frame. The LIDAR level is shown below.

Mechanical Design

RoveME's chassis is based on the AndyMark barebones chassis. It consists of a differential drive, with a castor in the back for support. The additional levels are simply threaded rod with polycarbonate plastic attached.

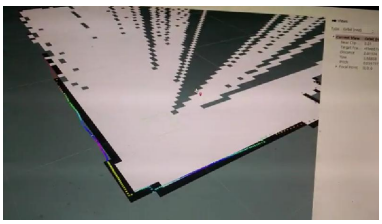
Software design

RoveME's software is designed around ROS (robot operating system). ROS is a messenger/subscriber platform that allows one to create nodes and topics that can communicate to one another. The way RoveME is set up topic-wise is as follows:



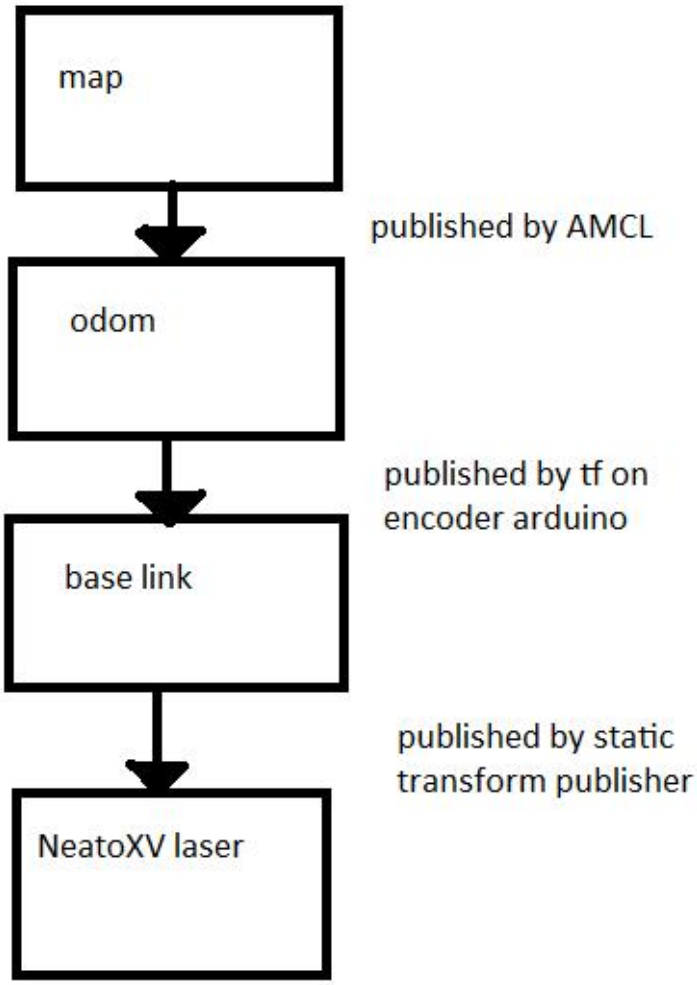
The LIDAR, Kinect and encoder topics all feed into the navigation stack and base controller. The navigation stack uses AMCL (adaptive monte carlo localization) to provide the dynamic transform between odom and map. It does this by using encoder data and laserscan data to localize against a static map. This static map must be obtained using gmapping. The base controller publishes the costmap topic. This allows the robot to decide, for example, that the cost of going through a small opening is too high to proceed is the opening is smaller than the robot's size. The base controller also handles path planning. Both a local and global path are published via base_controller.

Below is an example of what a static map looks like in Rviz:



Below one can find the complete (tf) transformation tree for RoveME.

TF tree for RoveME the indoor autonomous robot



Below is the RoveME bringup launch file. it launches all nodes needed to start navigating with RoveME. as stated at the top of the file, do not forget to launch the laserscan node, the map

server, and the rosserial_python node to communicate with the arduinos controlling the motors and encoders.

```
<launch>

 <!-- Don't forget to launch laser node, ros_serial server node and map server -->

 <!-- set up tf's -->
 <node pkg="tf" type="static_transform_publisher" name="link1_broadcaster" args="0 0 0 0 0 0  base_link neato_laser 100" />
 <!-- amcl wil provide the tf from odom to map -->

 <!-- start AMCL -->
 <node pkg="amcl" type="amcl" name="amcl" output="screen">
   <!-- Publish scans from best pose at a max of 10 Hz -->
   <param name="odom_model_type" value="diff"/>
   <param name="transform_tolerance" value="0.05" />
   <param name="gui_publish_rate" value="10.0"/>
   <param name="laser_max_beams" value="500"/>
   <param name="min_particles" value="500"/>
   <param name="max_particles" value="1000"/>
   <param name="kld_err" value="0.01"/>
   <param name="kld_z" value="0.99"/>
   <param name="odom_alpha1" value="0.8"/>
   <param name="odom_alpha2" value="0.8"/>
   <!-- translation std dev, m -->
   <param name="odom_alpha3" value="0.2"/>
   <param name="odom_alpha4" value="0.2"/>
   <param name="laser_z_hit" value="0.95"/>
   <param name="laser_z_short" value="0.05"/>
   <param name="laser_z_max" value="0.05"/>
   <param name="laser_z_rand" value="0.05"/>
   <param name="laser_sigma_hit" value="0.2"/>
   <param name="laser_lambda_short" value="0.1"/>
   <param name="laser_model_type" value="likelihood_field"/>
   <param name="laser_likelihood_max_dist" value="0.3"/>
   <param name="update_min_d" value="0.2"/>
   <param name="update_min_a" value="0.2"/>
   <param name="odom_frame_id" value="odom"/>
   <param name="resample_interval" value="1"/>
   <param name="recovery_alpha_slow" value="0.0"/>
   <param name="recovery_alpha_fast" value="0.0"/>
 </node>

 <!-- launch move base -->
 <node pkg="move_base" type="move_base" respawn="false" name="move_base" output="screen">
   <rosparam file="costmap_common_params.yaml" command="load" ns="global_costmap" />
   <rosparam file="costmap_common_params.yaml" command="load" ns="local_costmap" />
   <rosparam file="local_costmap_params.yaml" command="load" />
   <rosparam file="global_costmap_params.yaml" command="load" />
   <rosparam file="base_local_planner_params.yaml" command="load" />
 </node>

</launch>
```

This launches the amcl node and move base node.
Below the yaml files for move base node can be found.

local_costmap_params.yaml

```yaml
local_costmap:
  global_frame: odom
  robot_base_frame: base_link
  update_frequency: 2.0
  publish_frequency: 1.0
  static_map: false
  rolling_window: true
  width: 2.5
  height: 2.5
  resolution: 0.025
```

global_costmap_params.yaml

```yaml
global_costmap:
  global_frame: /map
  robot_base_frame: base_link
  update_frequency: 1.0
  publish_frequency: 0.0
  static_map: true
```

base_local_planner_params.yaml

```yaml
controller_frequency: 5.0
TrajectoryPlannerROS:
  max_vel_x: 1
  min_vel_x: .01
  max_rotational_vel: .25
  min_in_place_rotational_vel: .1
  acc_lim_th: 0.2s
  acc_lim_x: 0.1
  acc_lim_y: 0.0
  holonomic_robot: false
  yaw_goal_tolerance: 0.05
  xy_goal_tolerance: 0.1
  goal_distance_bias: 0.8
  path_distance_bias: 0.6
  sim_time: 1.5
  heading_lookahead: 0.325

  vx_samples: 6
  vtheta_samples: 20
  dwa: false


```

costmap_common_params.yaml

```
obstacle_range: 2.5
raytrace_range: 5.0
footprint: [[0.30, 0.30], [0.30, -0.30], [-0.30, -0.30], [-0.30, 0.30]]
robot_radius: 0.30
inflation_radius: 0.42
transform_tolerance: 1.0

observation_sources: base_scan
base_scan: {sensor_frame: neato_laser, data_type: LaserScan, topic: scan, marking: true, clearing: true}
```

These files outline various parameters needed by the move_base node including robot footprint maximum speeds and accelerations, etc.

All of the up to date code can be found on my Github

Future Plans for RoveME

In the near future I plan to get an IMU and GPS unit for RoveME to improve localization. Some long term goals include getting a more rugged, outdoor capable chassis. This is possible as i have recently found out that the NeatoXV Lidar works outside.