Nanyang Polytechnic

School of Engineering

Diploma in Mechatronics Engineering

Final Year Project Report

Development of ROS Navigation Components for an Omni-directional Social Robot

Submitted by: LAI ZEYU(160081F)

Supervisor : Dr. Edwin Foo

Abstract

Now more and more robots are appearing in peoples live. Robots can do a lot of things instead of people. Social robot is one of the robot which can talk with people and take care of child or old people. In this report, I will mainly talk about the development on social Ruth move base part, use ROS to do navigation and a simple web side to control the navigation.

Before I started this project, the basic framework of navigation was basically completed, so in this report, I will talk more about tuning the move base navigation part and how to use web to do navigation.

- **Contents**

Background

The background from social robot ruth can see in other report by Chen Ting.

Hardware change

The Specifics	
Size/Dimensions	165 x 30 x 40 mm
Weight	0.3 kg
Range	0.4 – 2m
Depth Image Size	1280 *1024 (SXGA) @ 5FPS Windows Only
	640*480 (VGA) @ 30FPS
	320*240 (QVGA) @ 30FPS
	160*120 (QQVGA) @ 30FPS

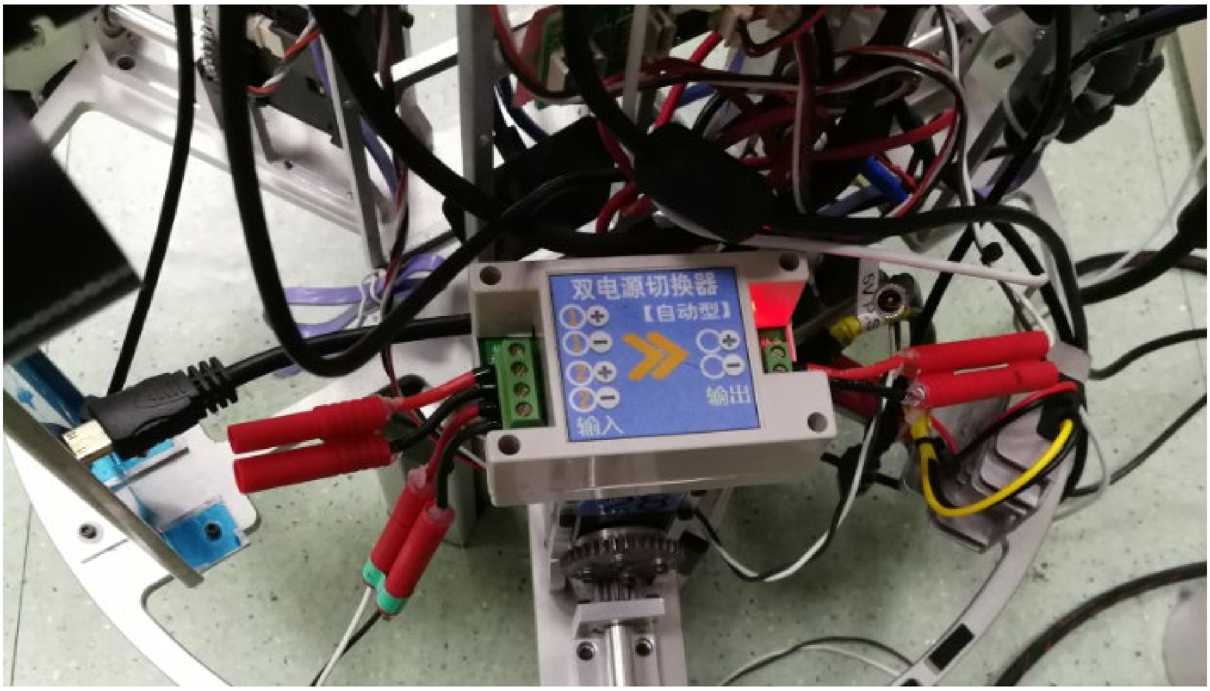
(Astra S technical specs)

The Specifics	
Size/Dimensions	165 x 30 x 40 mm
Weight	0.3 kg
Range	0.6 – 8.0m (Optimal 0.6 – 5.0m)
Depth Image Size	640*480 (VGA) @ 30FPS
	320*240 (QVGA) @ 30FPS
	160*120 (QQVGA) @ 30FPS

(Astra technical specs)

(RPlidar A2)

We use Astra S technical specs before, but we find the range is not enough for navigation.so we change to use Astra technical specs and rplidar A2 to do navigation scanning.



Dual power transfer switch

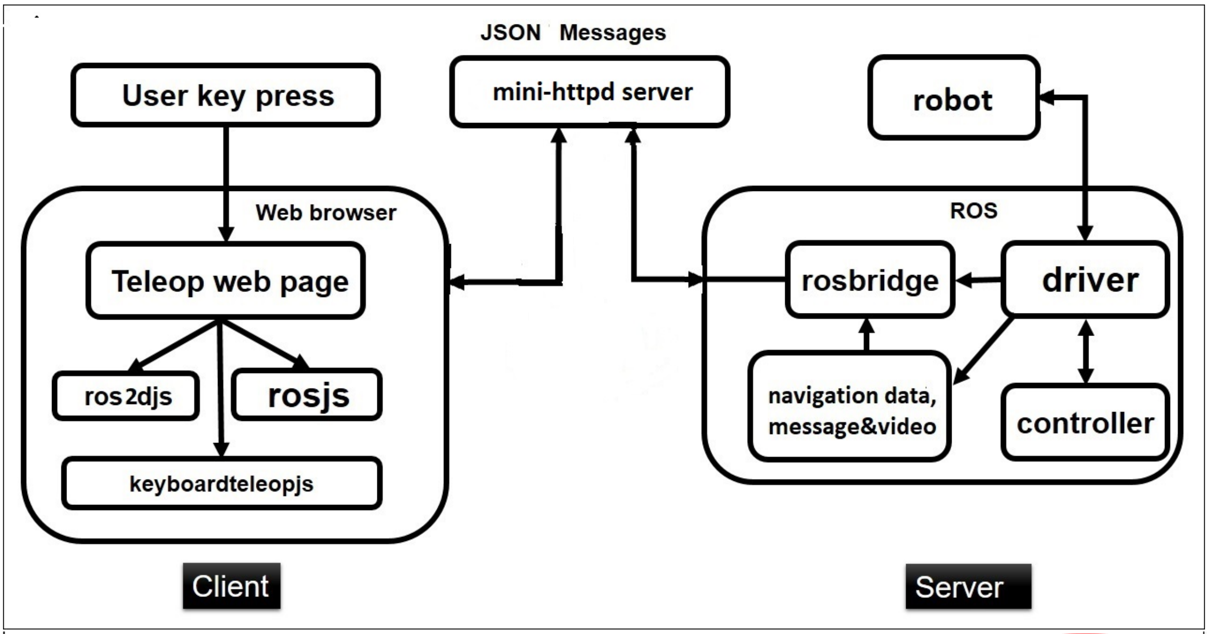
Fed by two independent power sources (a preferred and alternate), the transfer switch detects power failure and transfers to an alternate input source. This access to dual independent power sources offers unmatched reliability, availability and service ability the way no other solution do

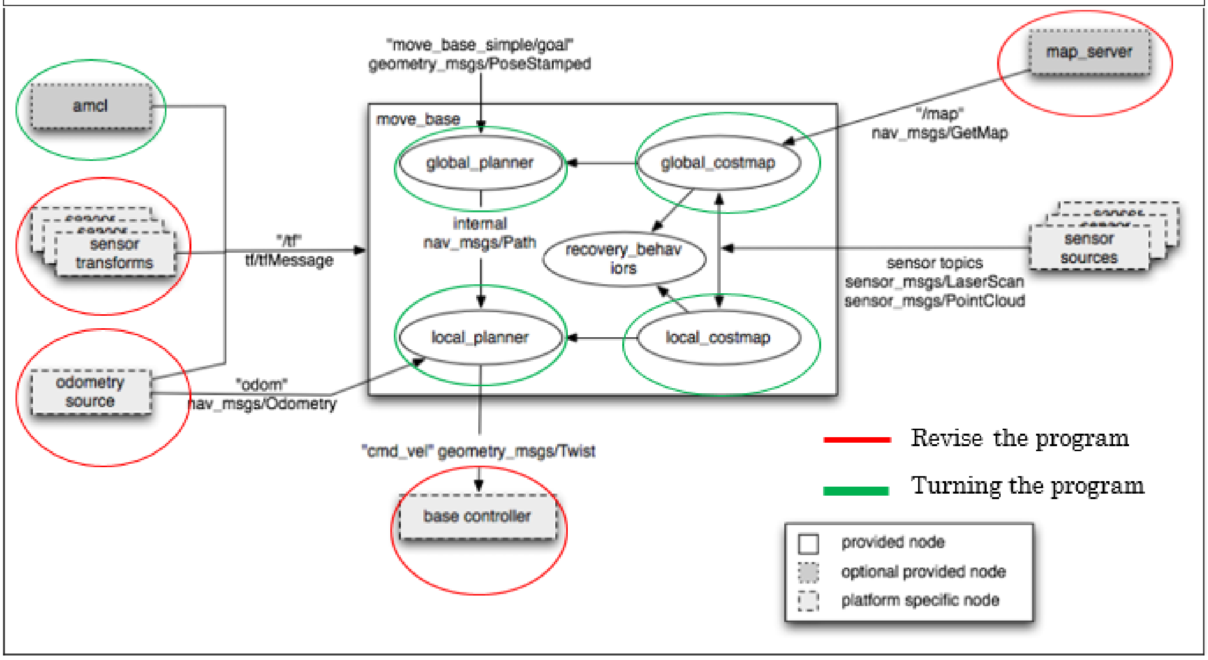
So can be converted directly between power supply and battery-powered, no need turn off computer.

*how to use will in guide.

Introduction

The introduction from social robot Ruth move base can see in other report by Chen Ting.





These two pictures show how to create a web navigation and how I adjust the ROS navigation part.

Tuning

Base controller & Odometry source

First ,I need tuning base controller & Odometry source to make sure the robot move in right velocity and can rotate to the correct angle. So I rearranged and adjusted the relevant formula to ensure accuracy.

Here is the formula for speed conversion.

```
in velocity_conversion  (vx,vy to wheel velocity)

           wheel_command.request.left_vel = (-((sqrt(3)) /
2)*(in_real_x_velocity) + (1.0/ 2)*(in_real_y_velocity) +
(L )*(in_angular)) /(1*R);
                wheel_command.request.right_vel = (((sqrt(3)) /
2)*(in_real_x_velocity) + (1.0/ 2)*(in_real_y_velocity) +
(L )*(in_angular)) /(1*R);
                wheel_command.request.center_vel = (-
(in_real_y_velocity)  + (L )*(in_angular)) /(1*R);
```

```
in velocity_control

wheel to motor :
int32_t VelocityControl::convertVelocity2Value(float velocity)
{
   return (int32_t) (velocity /3.0 *
multi_driver_->multi_dynamixel_[MOTOR]->velocity_to_value_ratio_);//
}
```

velocity:wheel velocity
3.0 :gear ratio
multi_driver_->multi_dynamixel_[MOTOR]->velocity_to_value_ratio_:velocity to value ratio

The above formula makes the wheel rotate at the correct speed.

The following formula ensures that the speed and position displayed on the map are the same as in reality.

motor feedback data(velocity)

```
  bool VelocityControl::controlLoop()
   {
     dynamixelStatePublish();
     ReadVelocity(); ///
   }
  void VelocityControl::ReadVelocity()
 {
  left_vel_c  =
multi_driver_->read_value_["present_velocity"]->at(0); //left_vel;
  right_vel_c =
multi_driver_->read_value_["present_velocity"]->at(1); //right_vel;
  center_vel_c =
multi_driver_->read_value_["present_velocity"]->at(2); //center_vel;

 }
```

send real motor velocity msg to tf (add gear ratio)

```
dynamixel_workbench_msgs::WheelCommandNew msg;
vel_ctrl.controlLoop();
msg.left_vel_new =left_vel_c*3.0; //left_vel;
msg.right_vel_new = right_vel_c*3.0; //right_vel;
msg.center_vel_new =center_vel_c*3.0; //center_vel;
dynamixel_wheel_vel_pub_.publish(msg);
```

in odo_pub (tf)

get real motor velocity(have gear ratio):

```
void velmessageCallback(const
dynamixel_workbench_msgs::WheelCommandNew::ConstPtr& msg)
{
     right_vel = msg->right_vel_new;
     left_vel = msg->left_vel_new;
     center_vel = msg->center_vel_new;
}
```

real motor velocity to real wheel velocity (add calibrate):

```
       double vl = left_vel * MOTOR_TO_WHEEL_VEL_RATIO *R;
       double vr = right_vel * MOTOR_TO_WHEEL_VEL_RATIO *R;
       double vc = center_vel * MOTOR_TO_WHEEL_VEL_RATIO *R;

       double linear_x_vel = 1.02*( -((vl ) / sqrt(3.0)) + ((vr ) /
sqrt(3.0)));
       double linear_y_vel = 1.02*(vr  + vl  -(2.0)*vc)/3.0;
       double angular = 1.115*((vl / L) + (vr / L) + (vc / L))/3.0;
```

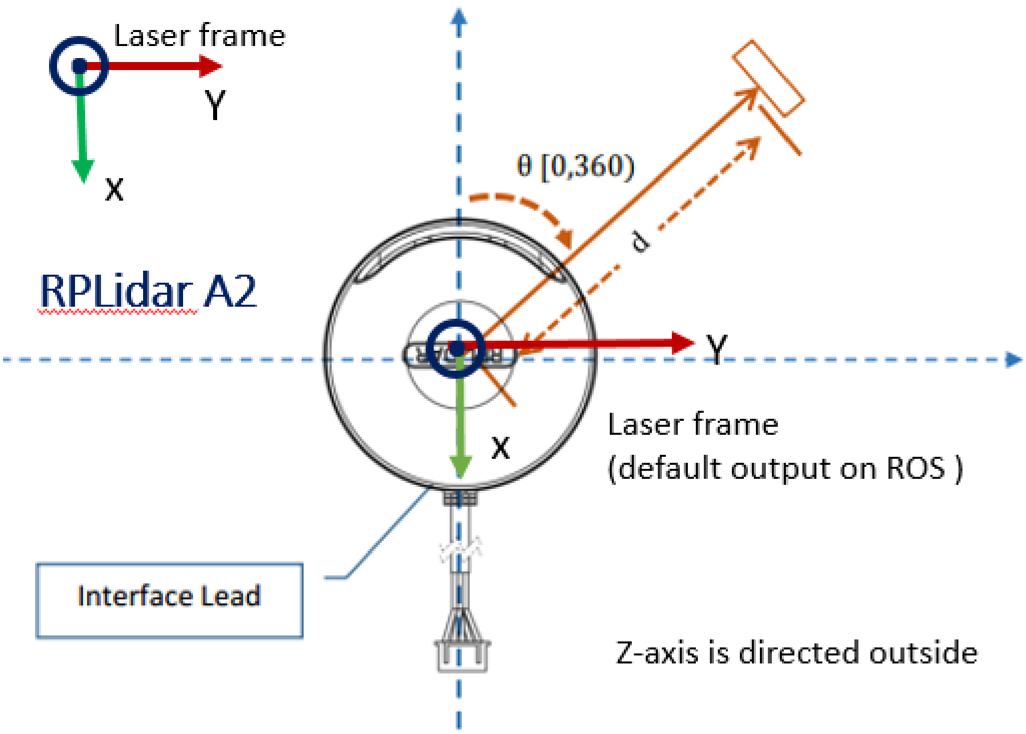
In the previous formula, the speed in the real environment is multiplied by a parameter that is used to correct errors in robot wheels and ground friction in the real environment.
In The guide will introduce how to get this parameter.

from real wheel velocity get real point

```
       double dt = (current_time - last_time).toSec();
       double delta_x = (linear_x_vel * cos(th) - linear_y_vel *
sin(th)) * dt;
       double delta_y = (linear_x_vel * sin(th) + linear_y_vel *
cos(th)) * dt;
       double delta_th = angular * dt;


       x += delta_x;
       y += delta_y;
       th += delta_th;
```

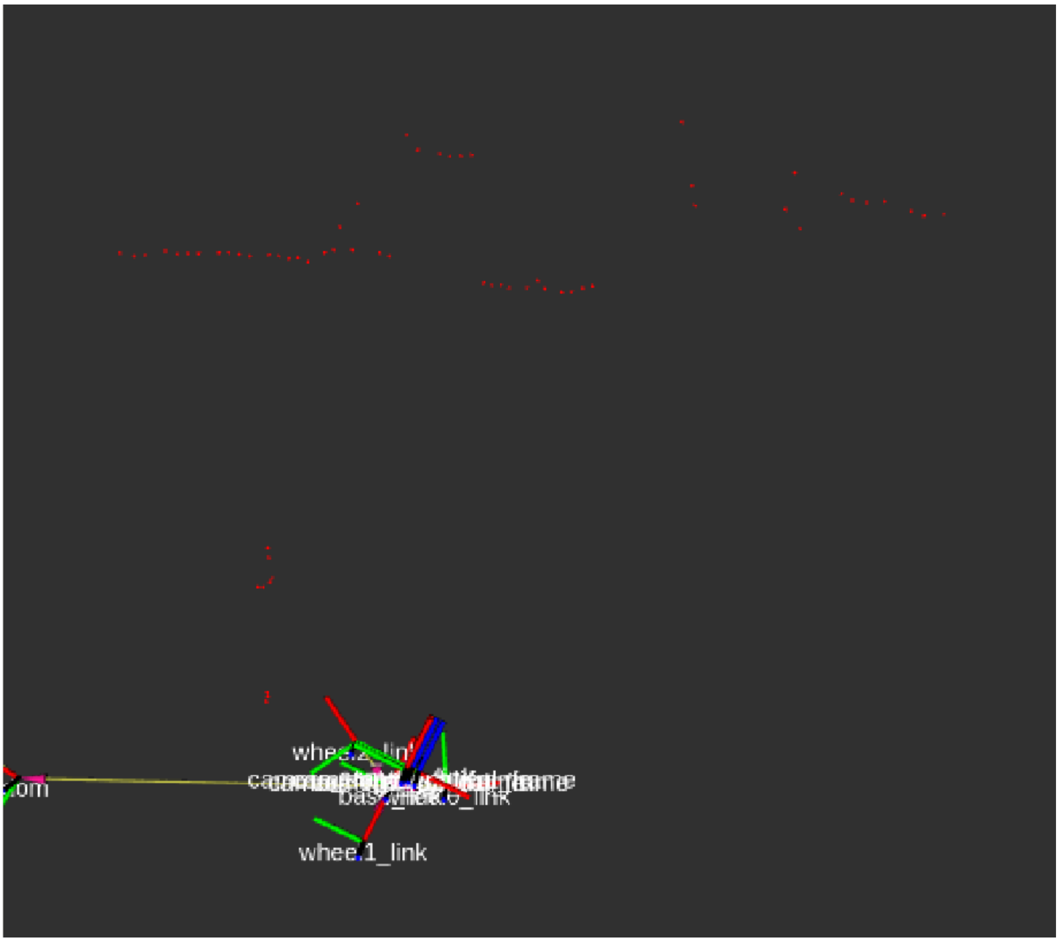
Now we can use these formulas to ensure that the robot does not have large systematic errors when it is moving. However, because the gear has some physical factors, the movement of the car is still very unstable. The following parameters adjustment for navigation is mainly to Reduce the impact of these issues on navigation.

Sensor source



The top is the default output of RPliadr A2. However, the direction of the radar parameters required by our robot is opposite to the default direction, and only the front 180 degrees of data are needed. So I modified the ridar output program.

```
in rplidar_ros
        change rplidar datadirection and only scan 90 degrees to 270
degrees
        if (i < right_degrees)
        {
           if (read_value == 0.0) scan_msg.ranges[2*degree_90 - i] =
std::numeric_limits<float>::infinity();
           else
              scan_msg.ranges[2*degree_90 - i] = read_value;
           scan_msg.intensities[2*degree_90 - i] = (float)
(nodes[i].sync_quality >> 2);
        }
        else if (i > left_degrees)
        {
           if (read_value == 0.0)
scan_msg.ranges[34:13:e8:34:ac:792*degree_270 - i] =
std::numeric_limits<float>::infinity();
           else
```

```
              scan_msg.ranges[2*degree_270 - i] = read_value;
           scan_msg.intensities[2*degree_270 - i] = (float)
(nodes[i].sync_quality >> 2);
        }
        else
        {
           //do nothing;
        }
```


This is the display image of RPridar's valid data in Rviz

Slam G-mapping

```
  <!-- <param name ="/use_sim_time" value="true"/> -->
   <include file="$(find my_dynamixel_workbench_tutorial)/launch/omniwheel_new.launch" />
<include file="$(find bringup)/launch/depthtolaser.launch"/>
   <!--<node pkg="tf" type="static_transform_publisher" name="base_link_2_camera_link" args="0.16
-0.0125 0.2850 0 0 0 /base_link /camera_link 100"/>  -->
   <node pkg="tf" type="static_transform_publisher" name="base_link_2_lider_link" args="0.16 -0.0125
0.2850 0 0 0 /base_link /laser 100"/>

  <node pkg="odom_tf_package" type="odo_pub" name="odo_pub" output="screen"/>


  <!-- gmapping node -->
  <node pkg="gmapping" type="slam_gmapping" name="slam_gmapping" output="screen">
    <param name="base_frame" value="base_link"/>
    <param name="odom_frame" value="odom"/>
    <param name="map_frame" value="map"/>
    <param name="map_update_interval" value="0.1"/>//
    <param name="maxUrange" value="4.0"/>  //
    <param name="maxRange" value="7.0"/>
    <param name="sigma" value="0.05"/>
    <param name="kernelSize" value="3"/>
    <param name="lstep" value="0.05"/>
    <param name="astep" value="0.05"/>
    <param name="iterations" value="5"/>
    <param name="lsigma" value="0.075"/>
    <param name="ogain" value="3.0"/>
    <param name="lskip" value="0"/>
    <param name="minimumScore" value="30"/>//
    <param name="srr" value="0.01"/>
    <param name="srt" value="0.02"/>
    <param name="str" value="0.01"/>
    <param name="stt" value="0.02"/>
    <param name="linearUpdate" value="0.05"/>//
    <param name="angularUpdate" value="0.0436"/>//
    <param name="temporalUpdate" value="-1.0"/>
    <param name="resampleThreshold" value="0.5"/>
    <param name="particles" value="20"/> //


    <param name="xmin" value="-10.0"/>
    <param name="ymin" value="-10.0"/>
    <param name="xmax" value="10.0"/>
    <param name="ymax" value="10.0"/>

    <param name="delta" value="0.05"/>
    <param name="llsamplerange" value="0.01"/>
    <param name="llsamplestep" value="0.01"/>
    <param name="lasamplerange" value="0.005"/>
    <param name="lasamplestep" value="0.005"/>
    <remap from="scan" to="/scan"/>
```

```
      <node pkg="tf" type="static_transform_publisher"
name="base_link_2_lider_link" args="0.16 -0.0125 0.2850 0 0 0
/base_link /laser 100"/>
```

```
        <param name="map_update_interval" value="0.1"/>
```

How long (in seconds) between updates to the map. Lowering this number updates the occupancy grid more often, at the expense of greater computational load. (float, default: 5.0)

we choose because when we try to mapping the map_update very show,so change to 0.1 can make the map_update bring up to the requirement.

```
<param name="maxUrange" value="5.0"/>
```

The maximum usable range of the laser. A beam is cropped to this value. (choose the senser scan range value)

```
<param name="maxRange" value="7.0"/>
```

The maximum range of the sensor. If regions with no obstacles within the range of the sensor should appear as free space in the map, set maxUrange < maximum range of the real sensor <= maxRange.

```
<param name="minimumScore" value="30"/>
```

Minimum score for considering the outcome of the scan matching good. Can avoid jumping pose estimates in large open spaces when using laser scanners with limited range (e.g. 5m). Scores go up to 600+, try 50 for example when experiencing jumping estimate issues. (float, default: 0.0)

because our lidar just can scanner 8m(camera only 5m),so we have the jumping estimte issues, after test we find value 30 can solve the problom.

```
<param name="linearUpdate" value="0.05"/>(float, default: 0.5)
```

Process a scan each time the robot translates this far (float, default: 1.0)

```
<param name="angularUpdate" value="0.0436"/>
```

Process a scan each time the robot rotates this far (float, default: 0.5)

```
<param name="particles" value="20"/>
```

Number of particles in the filter (int, default: 30) !!!!!!

This parameter is very important, now because the robot's odom is vary unstable,so we only can use 20 ,if the robot become very stable need change the velue higher.

Navigation

navigation.launch

```
  <include file="$(find my_dynamixel_workbench_tutorial)/launch/omniwheel_new.launch" />
  <!--<include file="$(find bringup)/launch/depthtolaser.launch" />-->
  <!--<node name="robot_state_publisher" pkg="robot_state_publisher" type="state_publisher" />-->

   <!--<node pkg="tf" type="static_transform_publisher" name="map_2_odom" args="0.0 0.0 0.0 0 0 0 /
map /odom 100"/>  -->

 <node pkg="tf" type="static_transform_publisher" name="base_link_2_camera_link" args="0.16 -0.0125
0.2850 0 0 0 /base_link /camera_link 100"/>
    <node pkg="tf" type="static_transform_publisher" name="base_link_2_lider_link" args="0 0 0.0500 0
0 0 /camera_link /laser 100"/>




 <!-- <param name="pub_map_odom_transform" value="true" />-->

  <node pkg="odom_tf_package"    type="odo_pub" name="odo_pub" output="screen"/>
  <!--<arg name="odom_topic"        default="/dynamixel_workbench_velocity_conversion/odom" />-->

  <arg name="map_file" default="$(find nnaavvii)/map/Rlvl4.yaml"/>
  <node name="map_server" pkg="map_server" type="map_server" args="$(arg map_file)" />

 <include file="$(find rplidar_ros)/launch/rplidar.launch" />
 <include file="$(find astra_launch)/launch/astra.launch" />

  <include file="$(find nnaavvii)/launch/aammccll.launch">
    <arg name="use_map_topic" value ="true"/>
  </include>


  <include file="$(find nnaavvii)/launch/move_base.launch" />
```

```
        <node pkg="tf" type="static_transform_publisher"
name="base_link_2_camera_link" args="0.16 -0.0125 0.2850 0 0 0
/base_link /camera_link 100"/>
        <node pkg="tf" type="static_transform_publisher"
name="base_link_2_lider_link" args="0 0 0.0500 0 0 0 /camera_link
/laser 100"/>
```

(link sonser to base)

move_base.launch

```
        <param name="local_costmap/inscribed_radius" value="0.32"/>
        <param name="local_costmap/circumscribed_radius"
value="0.32"/>
```

This two value very important, if the value oversize, the robot can‘t find way can go.

If the value is too small, the robot will hit some obstacle.it can be calculated with the method in the diagram.

```
:!-- move_base node -->
 <node pkg="move_base" type="move_base" respawn="false" name="move_base" output="screen">
   <!--<param name ="/use_sim_time" value="true"/>-->
   <param name="controller_frequency"          value="10.0"/>
   <rosparam file="$(find nnaavvii)/parafiles/costmap_common_params.yaml" command="load"
ıs="global_costmap" />
   <rosparam file="$(find nnaavvii)/parafiles/costmap_common_params.yaml" command="load"
ıs="local_costmap" />
   <rosparam file="$(find nnaavvii)/parafiles/local_costmap_params.yaml" command="load" />
   <rosparam file="$(find nnaavvii)/parafiles/global_costmap_params.yaml" command="load" />
   <rosparam file="$(find nnaavvii)/parafiles/base_local_planner_params.yaml" command="load" />
   <rosparam file="$(find nnaavvii)/parafiles/dwa_local_planner_params.yaml" command="load" />
   <rosparam file="$(find nnaavvii)/parafiles/navfn_global_planner_params.yaml" command="load" />
   <rosparam file="$(find nnaavvii)/parafiles/globalplanner_params.yaml" command="load" />
:!--
   <rosparam file="$(find nnaavvii)/parafiles/move_base_params.yaml" command="load" />

->


   <param name="global_costmap/robot_base_frame" value="base_link"/>
   <param name="global_costmap/global_frame" value="/map"/>
   <param name="local_costmap/inscribed_radius" value="0.32"/>
   <param name="local_costmap/circumscribed_radius" value="0.32"/>
   <param name="base_global_planner" value="navfn/NavfnROS"/>
   <param name="base_local_planner" value="base_local_planner/TrajectoryPlannerROS"/>

   <remap from="cmd_vel"  to="/esther_velocity_controller/cmd_vel"/>
```

aammccll.launch

```
<arg name="use_map_topic"   default="false"/>

<node pkg="amcl" type="amcl" name="amcl">
  <param name="use_map_topic"             value="$(arg use_map_topic)"/>
  <!-- Publish scans from best pose at a max of 10 Hz -->
  <param name="odom_model_type"           value="diff"/>

  <param name="gui_publish_rate"          value="10.0"/>
  <param name="laser_max_beams"           value="30"/>
  <param name="laser_max_range"           value="6.0"/>
  <param name="min_particles"             value="100"/>//500
  <param name="max_particles"             value="1000"/>//2000
  <param name="kld_err"                   value="0.05"/>
  <param name="kld_z"                     value="0.99"/>
  <param name="odom_alpha1"               value="0.2"/>
  <param name="odom_alpha2"               value="0.2"/>
  <!-- translation std dev, m -->
  <param name="odom_alpha3"               value="0.8"/>
  <param name="odom_alpha4"               value="0.2"/>
  <param name="odom_alpha5"               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_model_type" value="beam"/> -->
  <param name="laser_likelihood_max_dist" value="2.0"/>
  <param name="update_min_d"              value="0.2"/>
  <param name="update_min_a"              value="0.5"/>
  <param name="odom_frame_id"             value="odom"/>
  <param name="base_frame_id"             value="base_link"/>
  <param name="global_frame_id"           value="map"/>
  <param name="resample_interval"         value="2"/>
  <!-- Increase tolerance because the computer can get quite busy -->
  <param name="transform_tolerance"       value="0.1"/>///
  <param name="recovery_alpha_slow"       value="0"/>
  <param name="recovery_alpha_fast"       value="0"/>
  <param name="initial_pose_x" value="15.0"/>
  <param name="initial_pose_y" value="-7.6"/>
  <param name="initial_pose_a" value="3.14"/>
  <remap from="scan" to="/scan"/>
</node>
```

```
<param name="laser_max_range"          value="6.0"/>
```

Maximum scan range to be considered.(Maximum scan range use optimal rplidar)

```
<param name="min_particles"            value="100"/>(int, default: 100)
<param name="max_particles"            value="1000"/>(int, default: 5000)
```

Minimum allowed number of particles and maximum allowed number of particles. max_particles can’t be much because the robot need spend a lot of time to find the right pose.(But more partcles means the pose more accuracy)

```
<param name="odom_alpha1"              value="0.2"/>
```

Specifies the expected noise in odometry's rotation estimate from the rotational component of the robot's motion.

```
<param name="odom_alpha2"              value="0.2"/>
```

Specifies the expected noise in odometry's rotation estimate from translational component of the robot's motion.

```
<param name="odom_alpha3"              value="0.8"/>(double, default: 0.2)
```

Specifies the expected noise in odometry's translation estimate from the translational component of the robot's motion.

```
<param name="odom_alpha4"              value="0.2"/>
```

Specifies the expected noise in odometry's translation estimate from the rotational component of the robot's motion.

```
<param name="odom_alpha5"              value="0.2"/>
```

Translation-related noise parameter (only used if model is "omni").

Also because the odom problem, we set odom_alpha3 value very higher, so can reduce the expected noise from the translational component of the robot's motion.

base_local_planner

```
max_vel_x: 0.5
min_vel_x: 0.3
min_vel_y: 0.3
max_vel_y: 0.2
```

```
max_vel_theta: 0.5
min_in_place_vel_theta: 0.2
acc_lim_theta: 0.1
acc_lim_x: 0.02
acc_lim_y: 0.02
```

(Setting the velocity and acceleration)

```
sim_time: 3.0
```

The amount of time to forward-simulate trajectories in seconds.(double, default: 1.0)

Longer sim_time can see more simulate trajectories.

```
vx_samples: 6
vy_samples: 6
vtheta_samples: 20
```

The number of samples to use when exploring the x ,y,theta velocity space .

Because the odom is very unstable, so we use more samples.

```
meter_scoring: true
```

Whether the gdist_scale and pdist_scale parameters should assume that goal_distance and path_distance are expressed in units of meters or cells. Cells are assumed by default.

We use meter scoring bucause cell well change size in different map but meter always same, so use meter_scoring can become higher adaptability.

```
holonomic_robot: false
```

Now, we can't use holonomic mode because difficulties to reach the goal, winding trajectories, endless rotations...

```
TrajectoryPlannerROS:
  max_vel_x: 0.5
  min_vel_x: 0.3
  min_vel_y: 0.3
  max_vel_y: 0.2
  max_vel_theta: 0.5
  min_in_place_vel_theta: 0.2
  yaw_goal_tolerance: 0.05
  xy_goal_tolerance: 0.10

  acc_lim_theta: 0.1
  acc_lim_x: 0.02
  acc_lim_y: 0.02

  sim_time: 3.0
  sim_granularity: 0.025
#angular_sim_granularity:
  vx_samples: 6
  vy_samples: 6
  vtheta_samples: 20
  pdist_scale: 0.6
  gdist_scale: 0.8
  occdist_scale: 0.01
  meter_scoring: true
  heading_lookahead: 0.325
  oscillation_reset_dist: 0.05
  escape_reset_dist: -0.1
  escape_vel:-0.07
  holonomic_robot: false
  y_vels: [-0.6, -0.1, 0.1, 0.6]
```

Some tuning suggest can see ROSNavigationGuide in https://github.com/zkytony/ROSNavigationGuide/blob/master/main.pdf

globalpanner_params

```
cost_factor:  0.55
```

bast line when the value is 0.55

```
publish_scale: 100
planner_costmap_publish_frequency: 10.0
```

Can publish better than other value.

```
GlobalPlanner:

  lethal_cost:  120 #253
  neutral_cost: 50
  cost_factor:  0.55  #3.0
  publish_potential: true

  planner_window_x: 0.0
  planner_window_y: 0.0
  default_tolerance: 0.0

  old_navfn_behavior: false
  use_quadratic: true
  use_dijkstra: true
  use_grid_path: false
  allow_unknown: true

  publish_scale: 100

  planner_costmap_publish_frequency: 10.0
```

Some other navigation params

```
local_costmap:
 global_frame: odom
 robot_base_frame: base_link
 update_frequency: 5.0
 publish_frequency: 2.0
 static_map: false  # cost_map will always update==>false

 rolling_window: true
 width: 4.0
 height: 4.0
 resolution: 0.03
 transform_tolerance: 10.0
 plugins:
     - {name: obstacle_layer,      type: "costmap_2d::VoxelLayer"}
     - {name: inflation_layer,     type: "costmap_2d::InflationLayer"}
     - {name: voxel_layer,         type: "costmap_2d::VoxelLayer"}


shutdown_costmaps: false

controller_frequency: 5.0
controller_patience: 3.0


planner_frequency: 1.0
planner_patience: 5.0

oscillation_timeout: 0.0
oscillation_distance: 0.5

# local planner - default is trajectory rollout
base_local_planner: "dwa_local_planner/DWAPlannerROS"

base_global_planner: "navfn/NavfnROS" #alternatives: global_planner/GlobalPlanner, carrot_planner/
CarrotPlanner

global_costmap:
 global_frame: /map
 robot_base_frame: base_link
 update_frequency: 5.0
 static_map: true
 transform_tolerance: 10.0


 resolution: 0.03
 transform_tolerance: 10.0
 plugins:
      - {name: static_layer,           type: "costmap_2d::StaticLayer"}
      - {name: obstacle_layer,         type: "costmap_2d::VoxelLayer"}
      - {name: inflation_layer,        type: "costmap_2d::InflationLayer"}
```

```
robot_radius: 0.236
obstacle_layer:
  enabled:              true
  track_unknown_space:  false    #true needed for disabling global path planning through unknown space
  obstacle_range: 2.5
  raytrace_range: 3.0
  observation_sources: laser_scan_sensor
  laser_scan_sensor: {data_type: LaserScan, topic: /scan, marking: true, clearing: true,
min_obstacle_height: 0.08, max_obstacle_height: 0.4, expected_update_rate: 0.0}
#transform_tolerance: 0.5
#publish_voxel_map: false
#point_cloud_sensor: {data_type: PointCloud2, topic: /camera/depth/points, marking: true, clearing:
true, expected_update_rate: 0.4}


inflation_layer:
  enabled:              true
  cost_scaling_factor:  10.0  # exponential rate at which the obstacle cost drops off (default: 10)
  inflation_radius:     0.5 #0.5  # max. distance from an obstacle at which costs are incurred for
planning paths.

static_layer:
  enabled:              true

voxel_layer:
  enabled:              true
  max_obstacle_height:  0.6
  origin_z:             0.0
  z_resolution:         0.2
  z_voxels:             2
  unknown_threshold:    15
  mark_threshold:       8
#combination_method:   1
```

```
DWAPlannerROS:

  sim_time: 1.0
  sim_granularity: 0.025
  angular_sim_granularity: 0.1

  path_distance_bias: 64.0  # 32.0
  goal_distance_bias: 24.0
  occdist_scale: 0.5       # 0.01

  stop_time_buffer: 0.2
  oscillation_reset_dist: 0.05
  oscillation_reset_angle: 0.2

  forward_point_distance: 0.325
  scaling_speed: 0.25
  max_scaling_factor: 0.2

  vx_samples: 6
  vy_samples: 6
  vtheta_samples: 20

  use_dwa: true
  restore_defaults: false

  publish_traj_pc : true
  publish_cost_grid_pc: true
  global_frame_id: odom
```

After setting these parameters, the robot can navigate through rviz. The specific navigation steps are below the guide.

Web set up components

Web server

We use mini-httpd to build a web server in NUC linux system.

mini_httpd is a small HTTP server. Its performance is not great, but for low or medium traffic sites it's quite adequate. It implements all the basic features of an HTTP server. The main reason why this project currently uses mini-httped as a server is that it is very easy to use.

Internet connection

We use rosbridge to connection and server, and use web video server to send camera video to web.

The tutorial can see in:

Rosbridge: http://wiki.ros.org/rosbridge_suite

Web video server: http://wiki.ros.org/web_video_server

Web production

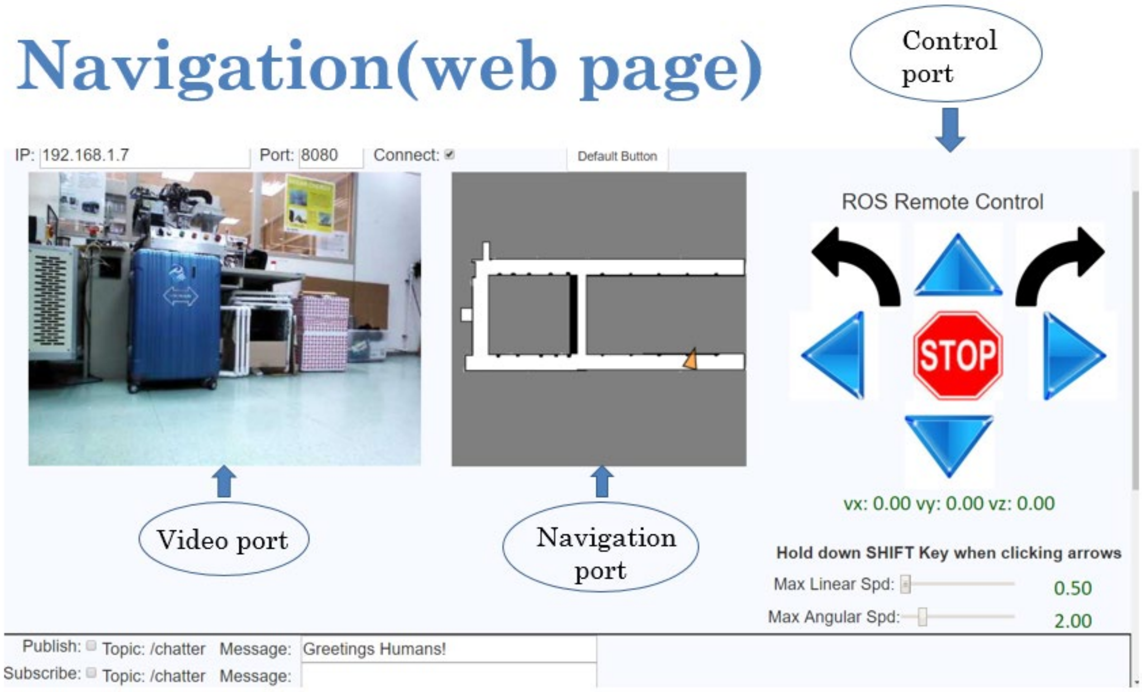
We get a web page template from：

https://github.com/technext/Office

And some Ros javescript programming in rbx2_gui:

https://github.com/pirobot/rbx2

Based on this change, and then made a basic navigation interface.



*How to use the web page will in guide.

Conclusion

Accomplishment

Completed mobile base set up for web navigation.

Tuning the ros navigation part.

Use G-mapping get map.

Suggestions and further enhancements

Omni-directional still have problem need check local planner.

guide

Hardware

How to turn on Ruth move base

How to use Dual power transfer switch

Software

How to setting up the ROS environment

How to import old navigation files

How to make the robot move

How to get motor velocity parameter

How to run sensor and see camera video

How to use rviz

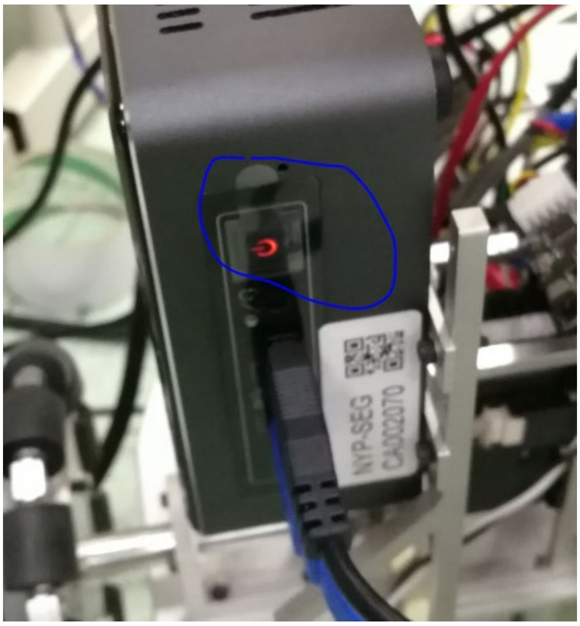
How to do slam G-mapping

How to do ros navigation

How to use web control the robot

How to turn on Ruth move base

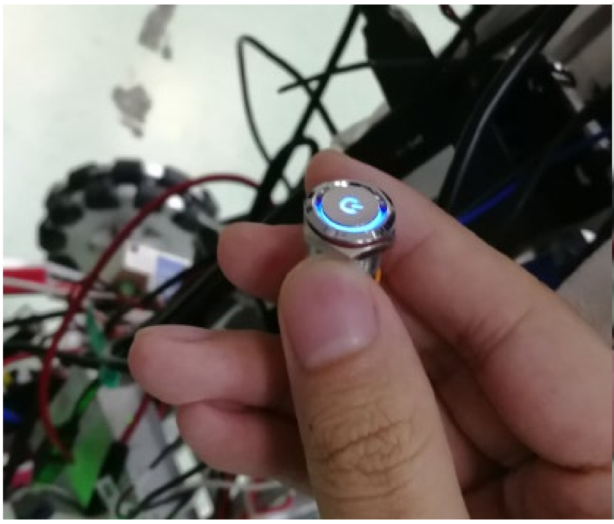
First turn on the main power switch



If can see the red light means power on. Now Press the red button to turn on the NUC. Can use lidar line light to check the NUC, if the green light is on means NUC connect to the lidar and work will.

Finally check if the motor emergency button is turned on.

(The button on the left picture is not open, the motor cannot work, the button on the right picture is already open, and the motor can work.)



If all correct, now you can start setting up the ROS environment.

How to setting up the ROS environment.

First, use this tutorials to install ROS.

http://wiki.ros.org/kinetic/Installation/Ubuntu

I will put the code which we need use under this sentence.

Open a terminal

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" >
/etc/apt/sources.list.d/ros-latest.list'
```

```
sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key
421C365BD9FF1F717815A3895523BAEEB01FA116
```

```
sudo apt-get update
```

```
sudo apt-get install ros-kinetic-desktop-full
```

```
apt-cache search ros-kinetic
```

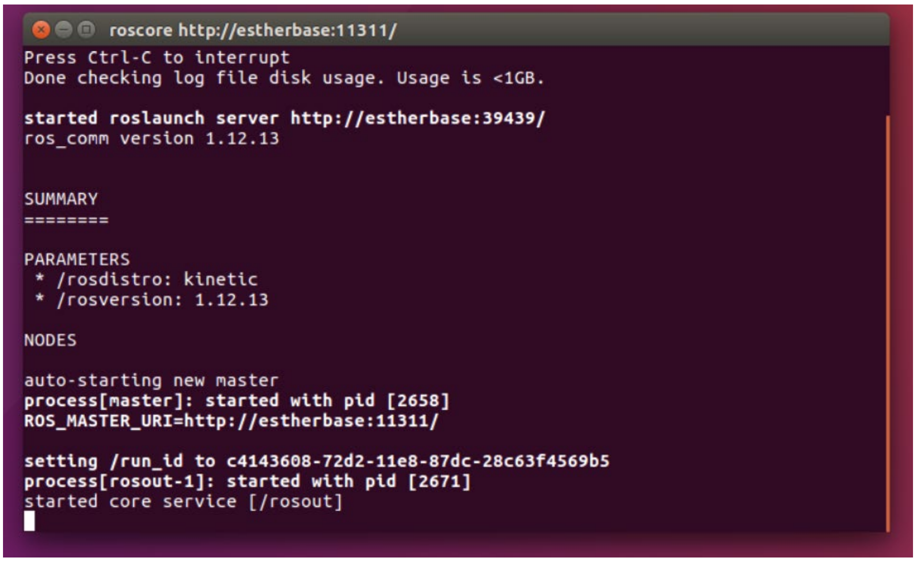
```
sudo rosdep init
rosdep update
```

```
echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
source ~/.bashrc
source /opt/ros/kinetic/setup.bash
```

```
sudo apt-get install python-rosinstall python-rosinstall-generator python-wstool build-essential
```

Now run roscore make sure Ros can work now.

If you see this means ROS can use now.



Now, let’s create and build a catkin workspace:

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/
catkin_make
```

Additionally, if you look in your current directory you should now have a 'build' and 'devel' folder. Inside the 'devel' folder you can see that there are now several setup.*sh files. Sourcing any of these files will overlay this workspace on top of your environment. To understand more about this see the general catkin documentation: catkin. Before continuing source your new setup.*sh file:

```
source devel/setup.bash
```

Now can add in my src to the catkin_ws.

How to import old navigation files.

First, put the src file in the new catkin_ws file. Before catkin_make you need do something to make every file can work and download some packages.

Open a terminal

```
sudo apt-get update

sudo apt-get upgrade

cd catkin_ws/

source devel/setup.bash

sudo apt-get install libbullet-dev

sudo apt-get install libsdl-image1.2-dev

sudo apt-get install libsdl-dev

sudo apt-get install libspnav-dev

sudo apt-get install ros-kinetic-serial
```

```
sudo apt-get install ros-kinetic-pointcloud-to-laserscan

sudo apt-get install ros-kinetic-navigation

sudo apt-get install ros-kinetic-rosbridge-server
```

Now delete catkin_wa/src/dynamic_reconfigure

After delete the file

```
cd src/
git clone https://github.com/ros/dynamic_reconfigure.git
cd ..
```

```
go to
home/nudesktop/catkin_ws/src/control_toolbox/cfg/parameters.cfg
choose permissions and
check out (Allow executing file as program)
same
on(/home/nudesktop/catkin_ws/src/nodelet_topic_tools/cfg/NodeletThro
ttle.cfg
/home/nudesktop/catkin_ws/src/rbx2/rbx2_utils/cfg/BatterySimulator.c
fg
/home/nudesktop/catkin_ws/src/rbx2/rbx2_utils/cfg/Pub3DTarget.cfg
/home/nudesktop/catkin_ws/src/ros_astra_camera/cfg/Astra.cfg
/home/nudesktop/catkin_ws/src/depthimage_to_laserscan/cfg/Depth.cfg
/home/nudesktop/catkin_ws/src/teraranger/cfg/TerarangerOne.cfg
/home/nudesktop/catkin_ws/src/teraranger/cfg/TerarangerDuo.cfg
/home/nudesktop/catkin_ws/src/teraranger_array/cfg/(all)
/home/nudesktop/catkin_ws/src/navigation/amcl/cfg/AMCL.cfg
/home/nudesktop/catkin_ws/src/rbx1/rbx1_nav/cfg/(all)
/home/nudesktop/catkin_ws/src/navigation/costmap_2d/cfg/(all)
/home/nudesktop/catkin_ws/src/navigation/base_local_planner/cfg/(all
)
/home/nudesktop/catkin_ws/src/navigation/dwa_local_planner/cfg/DWAPl
anner.cfg
/home/nudesktop/catkin_ws/src/navigation/global_planner/cfg/GlobalPl
anner.cfg
/home/nudesktop/catkin_ws/src/navigation/move_base/cfg/MoveBase.cfg)
```

Now can start catkin_make

```
catkin_make
```

Waiting 20~30min.

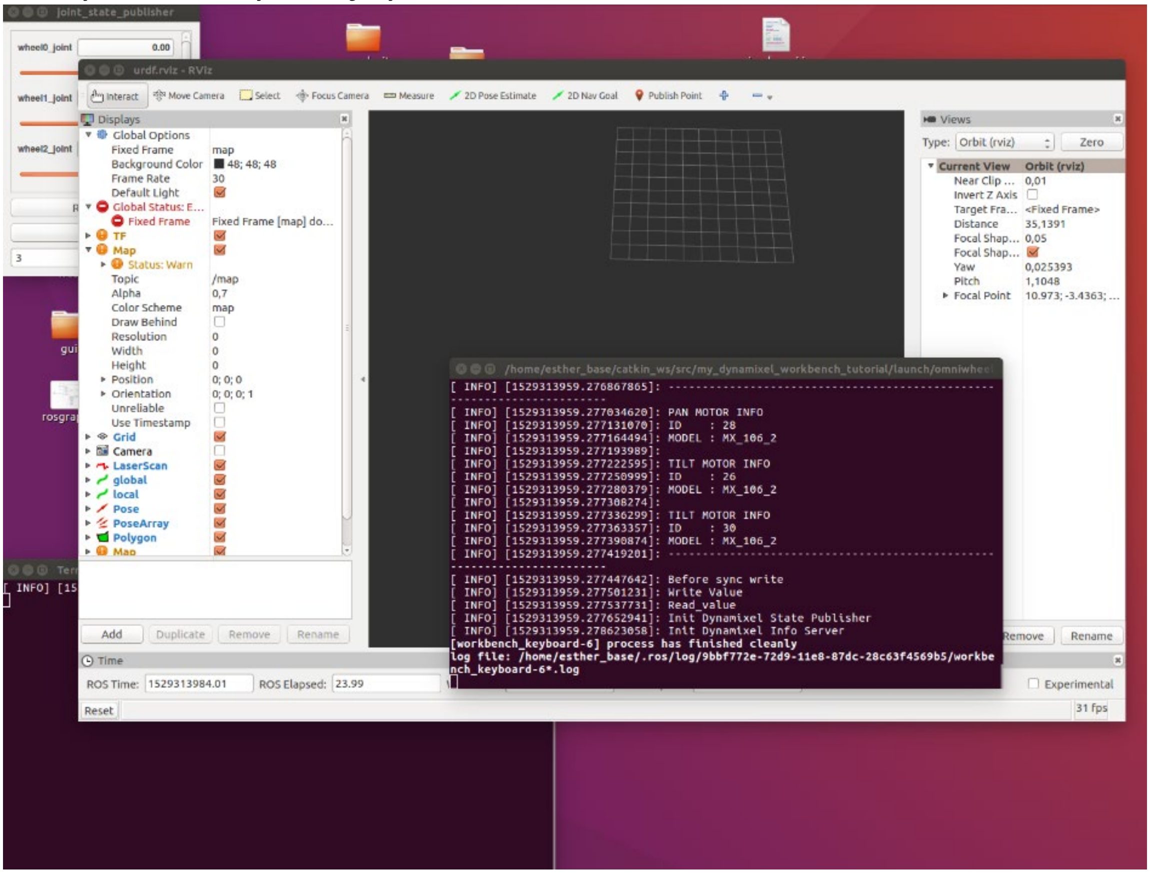
If there is no error as shown in the figure below, it is ready for use and you can proceed to the next step.

How to make the robot move

Open a terminal

```
cd catkin_ws/
source devel/setup.bash
sudo chmod a+rw /dev/ttyACM0
roslaunch my_dynamixel_workbench_tutorial omniwheel_new.launch
```

If there is no error as shown in the figure below, it is ready for use and you can try use joyestick.



For the joystick, you need hold LB key when you what to control with joystick. And the left controller of the handle controller the direction of movement and the right controller control the rotation left or right.

How to get motor velocity parameter

Open a terminal

```
cd catkin_ws/
source devel/setup.bash
sudo chmod a+rw /dev/ttyACM0
roslaunch nnaavvii navigation.launch
```

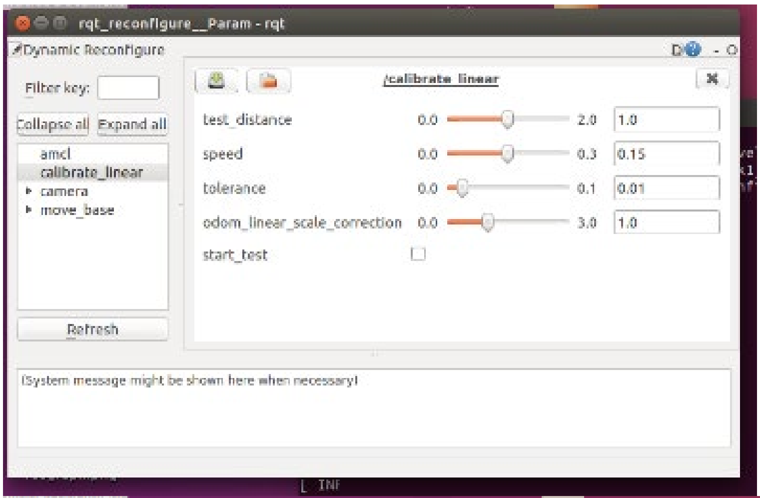
Open a new terminal

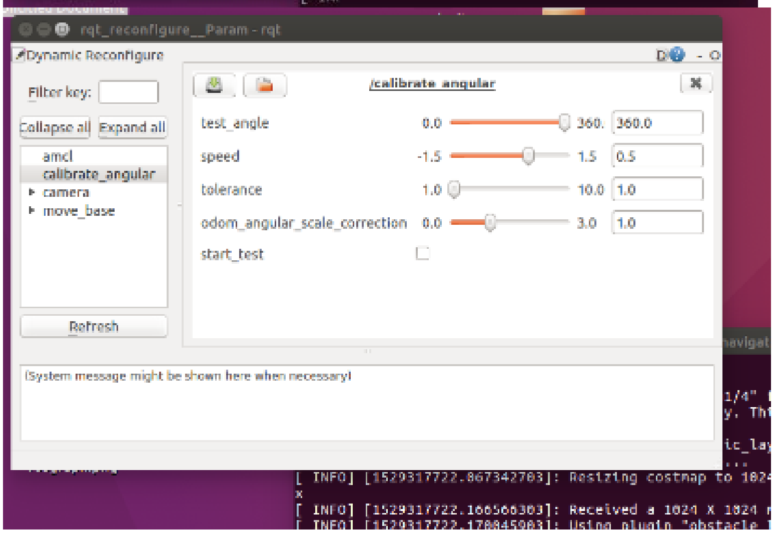
```
cd catkin_ws/
source devel/setup.bash

rosrun rbx1_nav calibrate_linear.py                   (calibrate_linear)
or
rosrun rbx1_nav calibrate_angular.py
```

Open a new terminal

```
rosrun rqt_reconfigure rqt_reconfigure
```

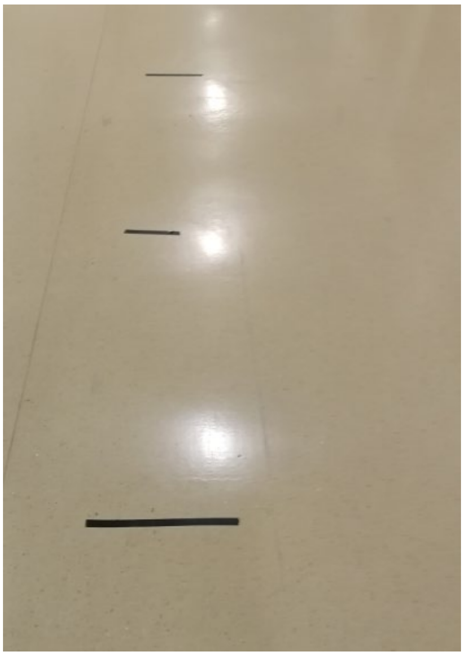
We will see like picture





Now can start calibrate.

Mark the beginning of the ground and the one-meter and two-meter positions.



Adjust the desired speed and distance in rqt, then click start-test.

After the robot stops, measure the actual distance and adjust odom_linear_scale_correction until the distance is very close to the set value. Then change the desired distance or speed until you can reach the set value, and then put this parameter in the formula introduced earlier. The method of adjusting the angular velocity is the same, and it will not_be stated here.

How to run sensor and see camera video

Run rplidar

Open a terminal

```
cd catkin_ws/
source devel/setup.bash
sudo chmod a+rw /dev/ttyACM1
roslaunch rplidar_ros rplidar.launch
```

Now you will find that the radar begins to rotate, which means that the radar has started to work.

Run Astra technical specs(camera) and see camera video and see camera video

Open a terminal

```
cd catkin_ws/
source devel/setup.bash
roslaunch astra_launch astra.launch
```

Open a new terminal

```
cd catkin_ws/
source devel/setup.bash
```