Gmapping:

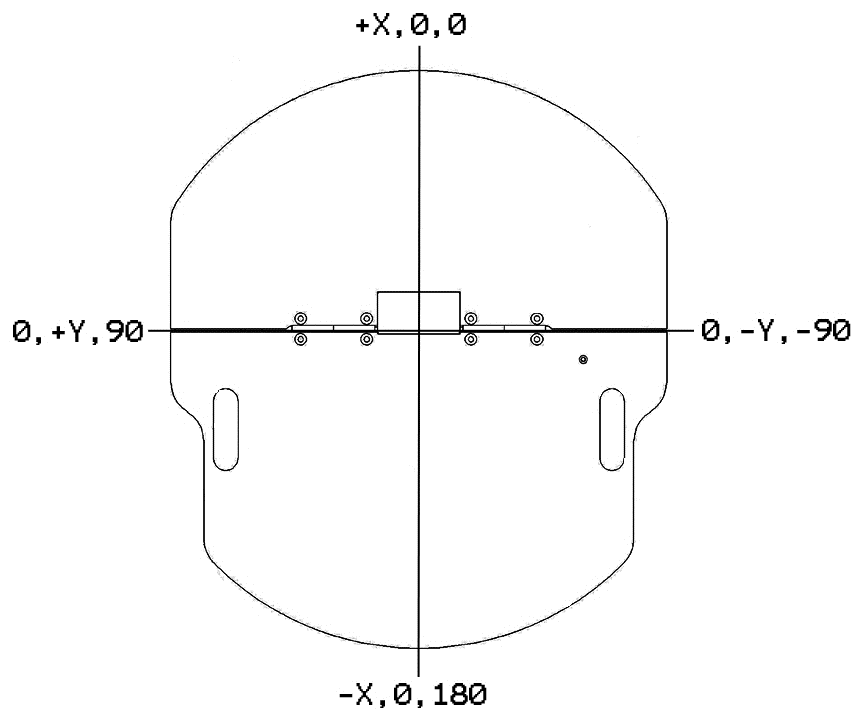
slam\_gmapping（Simultaneous Localization and Mapping） contains a wrapper around gmapping which provides SLAM capabilities.

Robot Slam\_gmapping can solve: At the same time positioning and map construction. the robot moves from an unknown position in an unknown environment, locates itself according to the position estimation and the map during the moving process, and simultaneously increases and establishes a map based on the positioning to realize independent positioning and navigation.

We installed the pioneer\_bringup package from the website by using the follow command.

**$ git clone https://github.com/amineHorseman/pioneer\_bringup.git.**

We use this package to get the data from pioneer. It can transfer data from rosaria. In the laser\_lms1xx.launch, we can find some information: call the minimal.launch to get the odom data by RosAria. Using the hokuyo to get the laser data. tf is a package that lets the user keep track of multiple coordinate frames over time. tf maintains the relationship between coordinate frames in a tree structure buffered in time, and lets the user transform points, vectors, etc between any two coordinate frames at any desired point in time. We can write some nodes about tf in this launch file to create the relationship between two coordinates. For example, the pioneer will create a local coordinate. The follow picture will show the coordinate. The hokuyo will also send a coordinate. We should connect these two coordinates by use the tf node. We should change the tf node according to the position of the hokuyo on the pioneer. We add hokuyo and Kinect on the deck. We write the tf for zero to simulate the hokuyo and Kinect are at the origin of coordinates, because the pioneer is a small robot, so there is a little error. If we get a big robot, we will change the tf data.



We also need to install Lms1xx for ros by using the follow command.

**$ sudo apt-get install ros-indigo-lms1xx.**

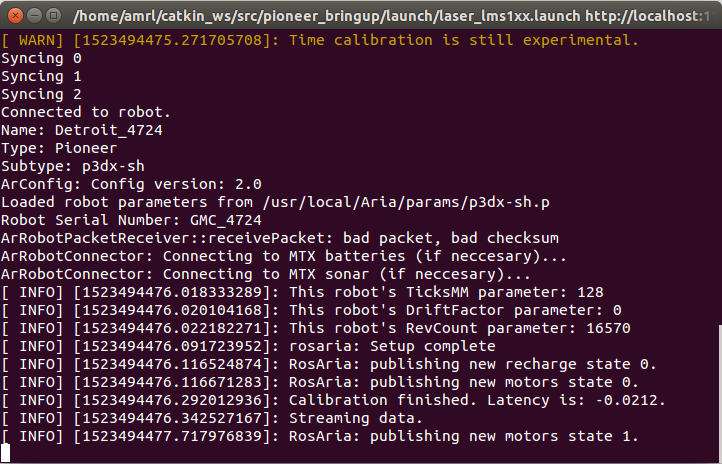
Then we installed the slam\_gmapping from the website by the follow command.

**$ git clone https://github.com/ros-perception/slam\_gmapping.**

Then we changed some data in the launch files, which are suit for the user. There are some information in the slam\_gmapping\_pr2.launch. we can change the speed of updating for the map and the size of the map. When we use gmapping, we should experiment repeatedly to find the suitable parameters.

After we install the slam\_gmapping successful, we run the follow commands in the terminals. First, open a new terminal. This command is for connecting the pioneer to the computer. When we receive streaming data. We connect successfully.

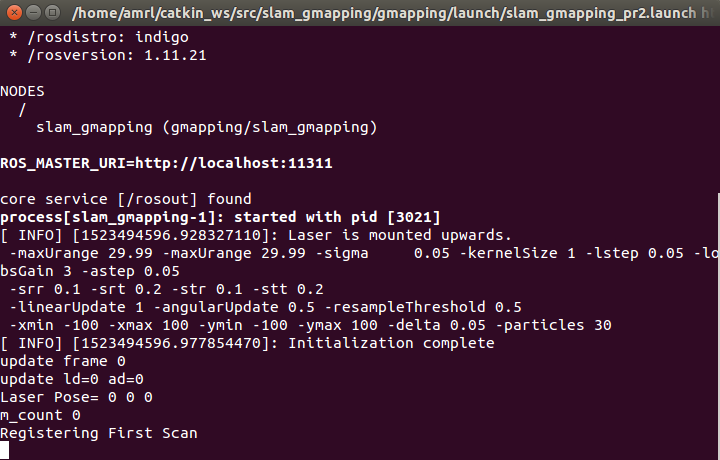
**$ roslaunch pioneer\_bringup laser\_lms1xx.launch.**



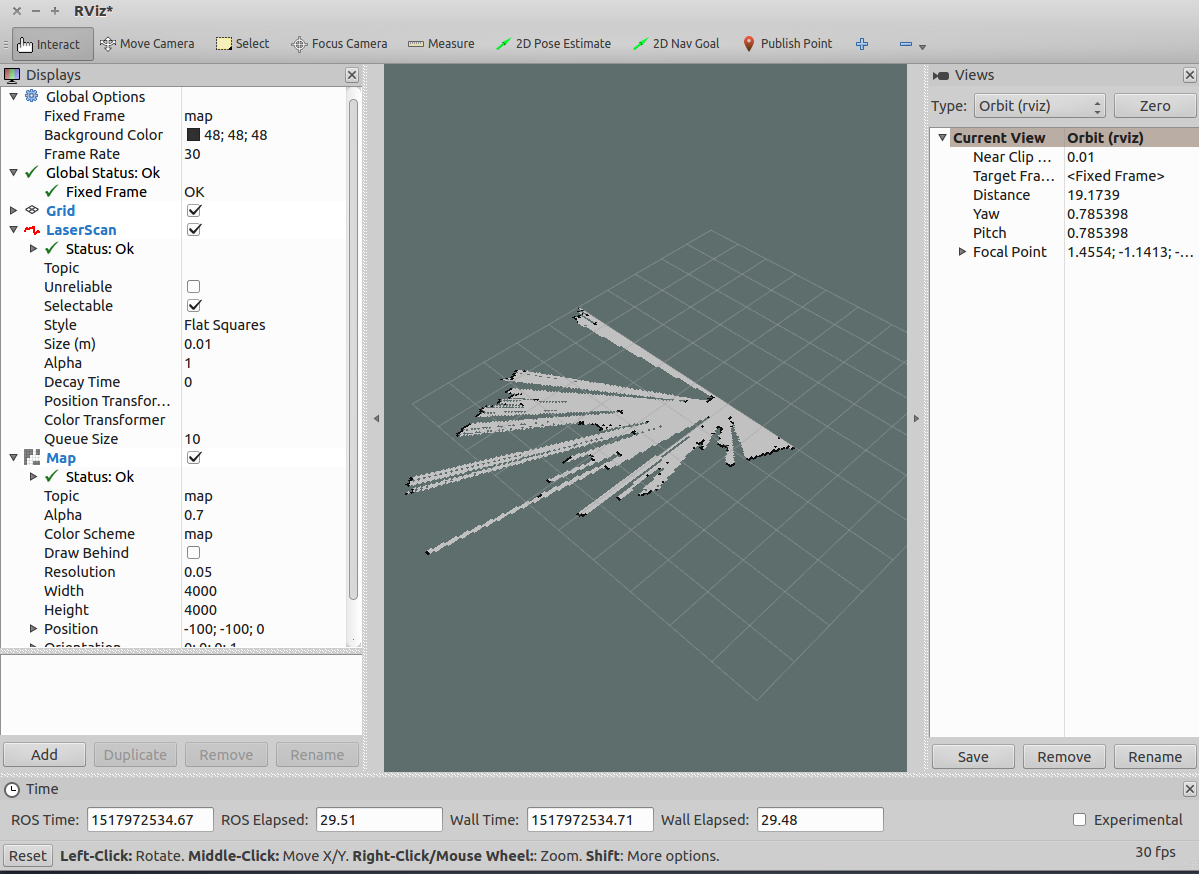
Secondly, write the follow command in a new terminal.

**$ roslaunch gmapping slam\_gmapping\_pr2.launch.**

when we receive the first data, we run successfully. We need to run the pioneer, then we can receive the follow data, because, the procedure shows that the pioneer need to run some distance in linear or turn some angle to receive the data from hokuyo.



Then open **rviz**. We can observe the map in the rviz. Create visualization by add map. If we want to see the coordinates, we will add tf. The fixed frame in the global options we choose map.

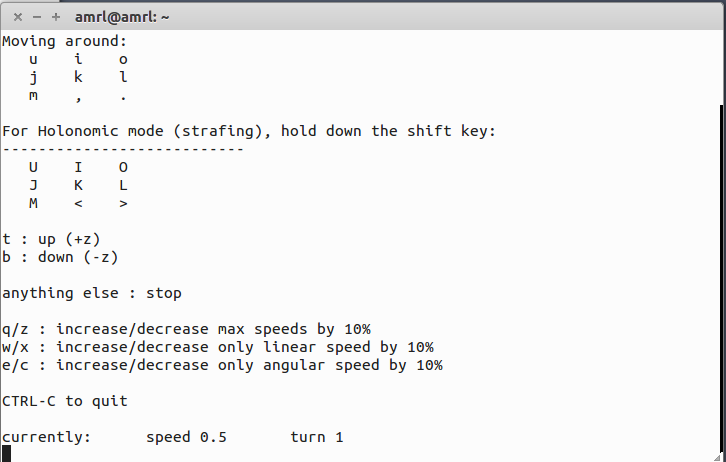


After those steps, we can find a part of the map in the rviz. The pioneer need to run to make a full map. We need to install a package to control the pioneer by keyboard by using command.

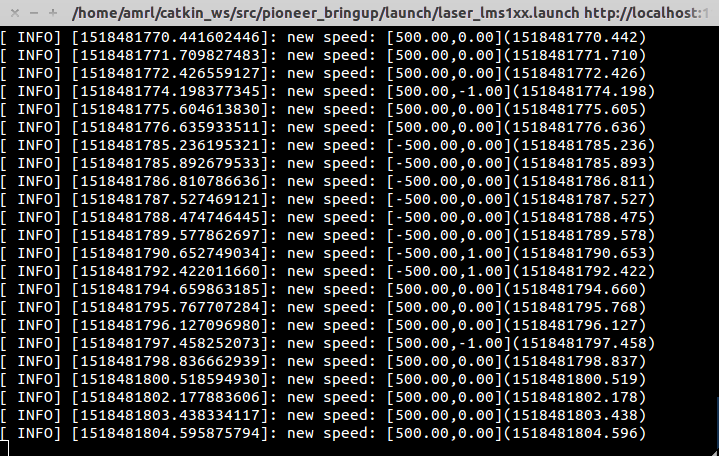
**$ sudo apt-get install ros-indigo-teleop-twist-keyboard**

We use the follow command in a new terminal to let the pioneer run. we can use the keyboard to control the pioneer.

**$ rosrun teleop\_twist\_keyboard teleop\_twist\_keyboard.py.**



This picture shows that we send the speed to pioneer.



When we scan the full map, we use the follow command to save the map. **$ rosrun map\_server map\_saver -f <map\_name>**

If we find some error that the computer cannot save the map, we will download the package for ros.

The gmapping is finish.