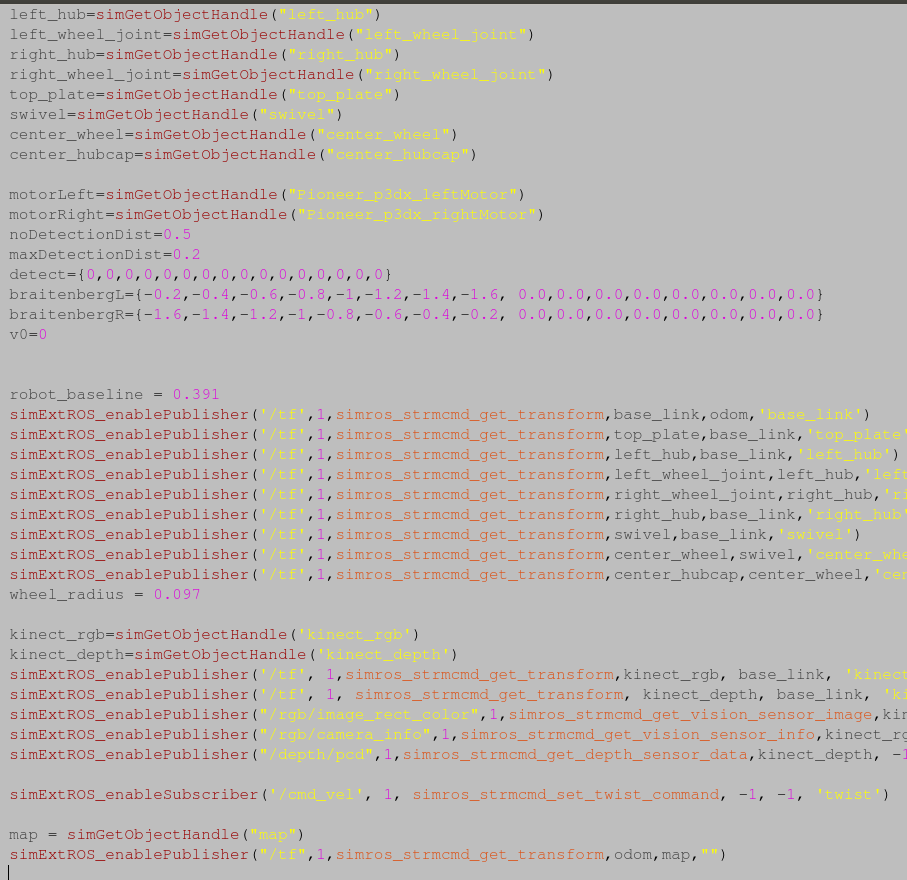
Robot simulation

# Interconnecting V-REP and ROS

To interconnect V-REP and ROS, a ROS workspace was created by installing the required packages and compiling the ROS workpspace with catkin\_make. Next to connect the V-REP with ROS, a terminal was loaded to start the roscore program to allow the two programs to communicate, then opening a V-REP simulation and loading the robot in the simulation. To actually pass information to ROS from V-REP the command simExtRos\_enablePublisher was used. This command passed the information, in this case the transformations, of the robot components and how they are organized in the heirachy. The command can be broken down into its parameters to explain it, first is the name of the topic ‘/tf’ for transform, 1 is the size of the queue, the simros\_str\_mcmd\_get\_transform denotes what type of message is being sent, ie the transformation. The next two are the transformation objects, the first one is the child frame, the second is the parent frame and the last parameter is the name for ROS to call this transformation in its own display. Below is the code for the information passing of the transformation of all the components.

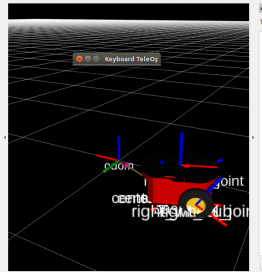
Which resulted in this diagram being created in ROS which clearly shows a diagram of the robot.



# Implementing Teleoperation

To allow the teleoperation of the robot within the simulation the movement information must be passed between V-REP and ROS, to do this we pass the linear and angular velocities of the robot. Using the command simExtRos\_enableSubscriber() instead of simExtRos\_enablePublisher as we are receiving velocities rather than sending them , with the parameter simros\_strmcmd\_set\_twist\_command instead of transformation. 

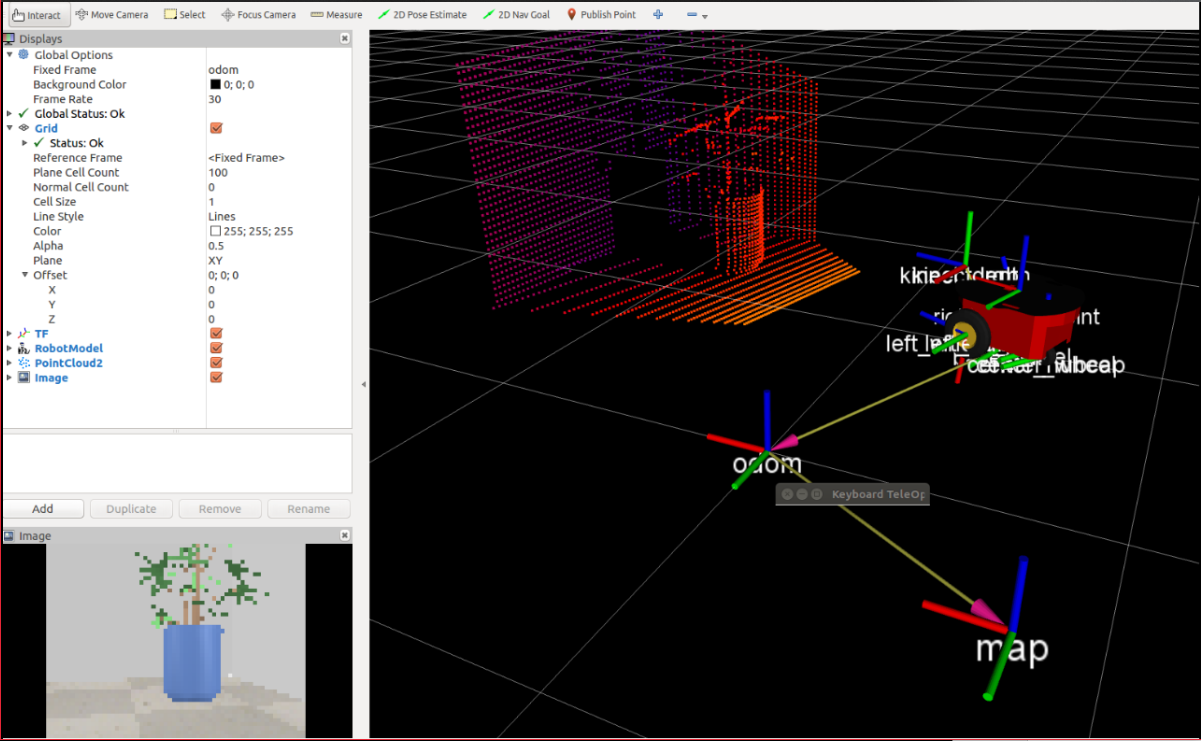
Then the teleoperation program was installed in the workspace and the ROS workspace recompiled. Running the teleoperation program in another terminal opened a small program that when focused let the user use the arrow keys on the keyboard to control the robot the program took the inputs converted them into velocities and through ROS sent those to the V-REP simulation.



# Extending Robot Perception

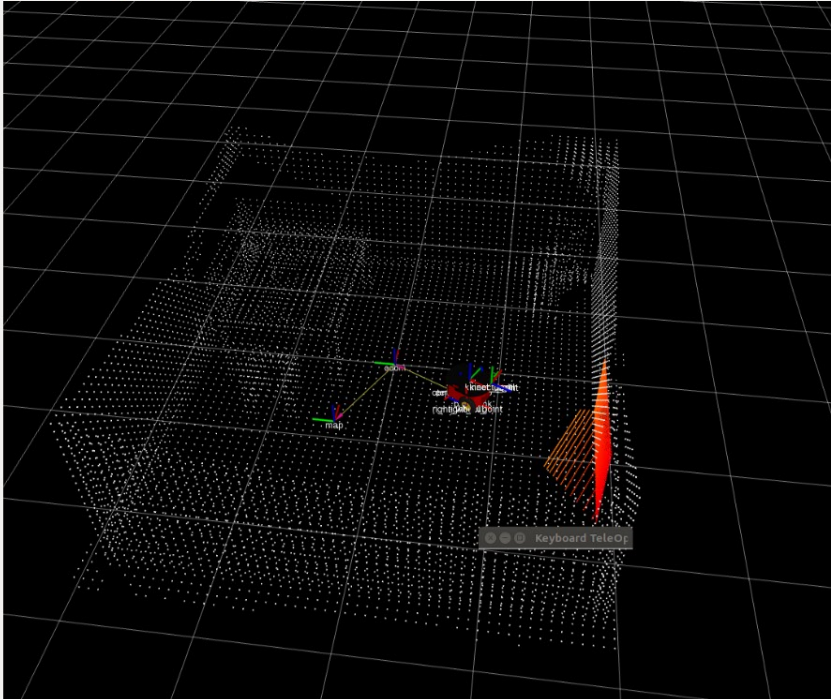
In order for a robot to have perception it needs to have a visual sensor, so this was placed onto the robot and attached to it within V-REP. Next the data from the sensor has to be transferred to ROS, this is done using the following code. 

This gives the kinetic rgb and Kinect depth imaging data which can be visualised in ROS by adding the image type to the display which when run looks like the figure below.



The last extension is to add the point cloud to ROS so that a world is build based on the data received, to do this a placed is needed to store the data that is collected. This is done by creating a dummy in V-REP, in this case named map and putting another simExtROS\_enablePublisher command transformation to the dummy. When the simulation is run again and the point cloud added to ROS, ROS now shows the point cloud display of what the sensors now see.

This figure is the room design in the V-REP simulation.

This Data here is the point cloud representation of what the robot’s sensor collects.