

Jimmy Robot

by MO (Mike Odessky), Rev_3.00 (Hokuyo 3D lidar & diff drive base)



Welcome

Any interest in this robot is very welcome. If you want to contribute, I will enable you to the best of my ability. Let's chat.... TorontoRoboticsClub@protonmail.com

Scope

This package is the source code for a physical robot that I am building as a submission for SCIE496 for Athabasca University. This package allows you to both simulate the robot in Gazebo, as well as use the physical robot.

About the Robot

The robot is a differential drive robot, with a 3D lidar. It is using the ROS navigation stack, and will shortly receive two 7 axis robotic arms.

Software Used

Required Software

This ROS software package builds on the following system:

- Ubuntu 18.04
- ROS Melodic
- Gazebo 9.13 or Gazebo 9.0

Dependencies for Building the Project

These packages are required to build the project

- Libserial is version 1.0 and is installed.
- Nvidia driver (and card) is needed to render the lidar in Gazebo. The open source Ubuntu driver does not allow Gazebo lidar data to work.

Preparing PC to Build/Run Project

I have done some work on my end to make it reasonably easy to run this project. However, if for any reason the project does not build on your PC, please email me with your issue. This document assumes that you have already done the following:

- Installed Ubuntu 18.04 and ran all the updates.
- Installed ROS Melodic (see: <https://wiki.ros.org/melodic/Installation/Ubuntu>)
- Installed ros_control lib (see section 7 here: https://wiki.ros.org/ros_control)

Below are the actions you need to take to build and run the Gazebo project (with the option to run the physical unit):

(1) Install LibSerial POSIX Library – This library is available here:

<https://github.com/crayzeewulf/libserial/> and it makes accessing serial ports on a POSIX system easy.

Thanks Crayzeewulf and friends! You will need to install this package from source – do not install from other sources as the older version of libserial (ver 0.6) will not work with the robot! My notes are included for your convenience:

By installing from git you are guaranteed to get the latest library. This is straight out of the README.md (<https://github.com/crayzeewulf/libserial/>).

Do an update:

```
$ sudo apt update
```

Install a bunch of packages:

```
$ sudo apt install g++ git autogen autoconf build-essential cmake graphviz libboost-dev  
libboost-test-dev libgtest-dev libtool python3-sip-dev doxygen python3-sphinx pkg-config python3-  
sphinx-rtd-theme
```

I guess now is a good time to download the repository. Put it into /Desktop/some_dir. Unzip it. Navigate to the unzipped directory via the terminal.

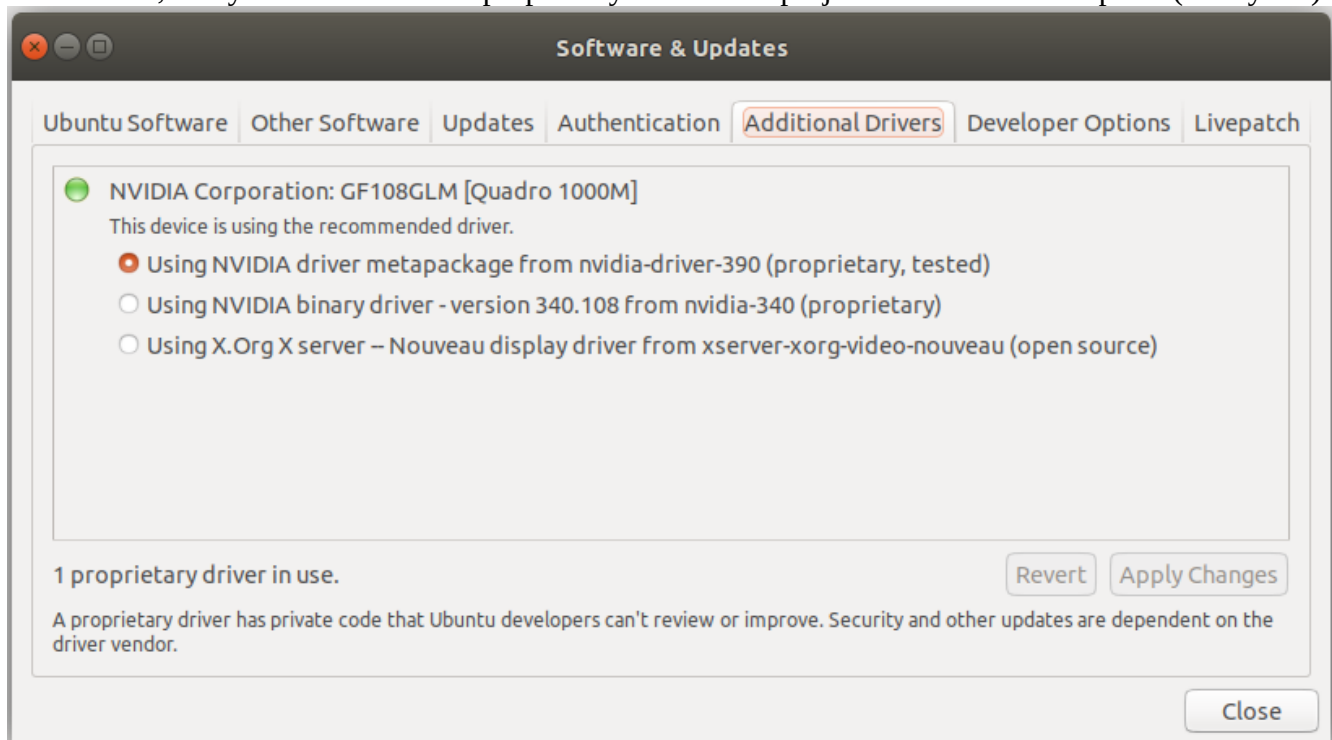
```
$ ./compile.sh
```

build and install

```
$ cd build  
$ sudo make install
```

All of this worked for me. Sweet. CrayzeeWulf and friends rock!

(2) Nvidia Driver Activation – The file workspace/src/jimmy_description/urdf/jimmy.gazebo takes care of loading a lidar plugin (gpu_ray plugin). The lidar plugin provides virtual lidar data based on the gazebo environment. If the Nvidia driver is not activated, this plugin breaks – and the lidar is not rendered in Gazebo, and the scan topic /jimmy/lidar/scan will have no messages. Basically, you need an Nvidia card, and you need to use the proprietary driver. The project works with this option (on my PC):



(3) Install All ROS Packages – It turns out that many navigation packages (that can be seen here <https://wiki.ros.org/navigation>) are not preinstalled in a default ros installation. This section is a list of all the packages I had to install into my PC as part of a ROS install. It is assumed you have these packages installed (ros_control, and the navigation stack):

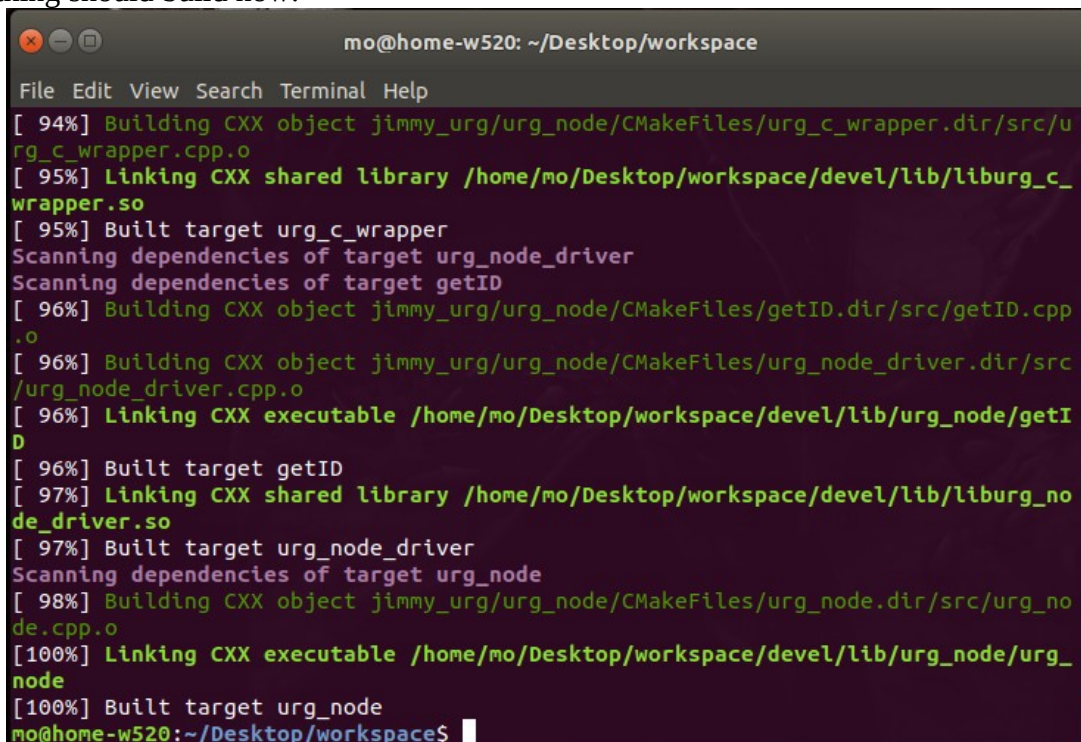
- \$ sudo apt-get install ros-melodic-ros-control ros-melodic-ros-controllers
- \$ sudo apt-get install ros-melodic-costmap-2d
- \$ sudo apt-get install ros-melodic-map-server
- \$ sudo apt-get install ros-melodic-mrpt-map
- \$ sudo apt-get install ros-melodic-amcl
- \$ sudo apt-get install ros-melodic-amcl-dbgSYM
- \$ sudo apt-get install ros-melodic-mrpt-localization //not amcl, but look cool
- \$ sudo apt-get install ros-melodic-mir-navigation
- \$ sudo apt-get install ros-melodic-move-base-msgs
- \$ sudo apt-get install ros-melodic-dwa-local-planner
- \$ sudo apt-get install ros-melodic-global-planner
- \$ sudo apt-get install ros-melodic-global-planner-dbgSYM
- \$ sudo apt-get install ros-melodic-global-planner-tests
- \$ sudo apt-get install ros-melodic-move-slow-and-clear
- \$ sudo apt-get install ros-melodic-move-slow-and-clear-dbgSYM
- \$ sudo apt-get install ros-melodic-navfn-dbgSYM
- \$ sudo apt-get install ros-melodic-rotate-recovery-dbgSYM
- \$ sudo apt-get install ros-melodic-voxel-grid-dbgSYM

Building the Code

I know the steps here are pretty micro, but it's useful documentation to create. Create a 'workspace' directory on your Desktop, and put the 'src' directory from github inside workspace. Build the code with these instructions:

- \$ cd Desktop/workspace
- \$ catkin_make

Everything should build now:



```
mo@home-w520: ~/Desktop/workspace
File Edit View Search Terminal Help
[ 94%] Building CXX object jimmy_urg/urg_node/CMakeFiles/urg_c_wrapper.dir/src/urg_c_wrapper.cpp.o
[ 95%] Linking CXX shared library /home/mo/Desktop/workspace/devel/lib/liburg_c_wrapper.so
[ 95%] Built target urg_c_wrapper
Scanning dependencies of target urg_node_driver
Scanning dependencies of target getID
[ 96%] Building CXX object jimmy_urg/urg_node/CMakeFiles/getID.dir/src/getID.cpp.o
[ 96%] Building CXX object jimmy_urg/urg_node/CMakeFiles/urg_node_driver.dir/src/urg_node_driver.cpp.o
[ 96%] Linking CXX executable /home/mo/Desktop/workspace/devel/lib/urg_node/getID
[ 96%] Built target getID
[ 97%] Linking CXX shared library /home/mo/Desktop/workspace/devel/lib/liburg_node_driver.so
[ 97%] Built target urg_node_driver
Scanning dependencies of target urg_node
[ 98%] Building CXX object jimmy_urg/urg_node/CMakeFiles/urg_node.dir/src/urg_node.cpp.o
[100%] Linking CXX executable /home/mo/Desktop/workspace/devel/lib/urg_node/urg_node
[100%] Built target urg_node
mo@home-w520:~/Desktop/workspace$
```

Running The Code

The project was set up in such a way as to make running it as easy as possible. The core capabilities of the robot (for now) were broken up into 6 demonstrations that are found in the `src/jimmy_welcome/launch` directory. The capabilities (along with the corresponding launch file name) are as follows:

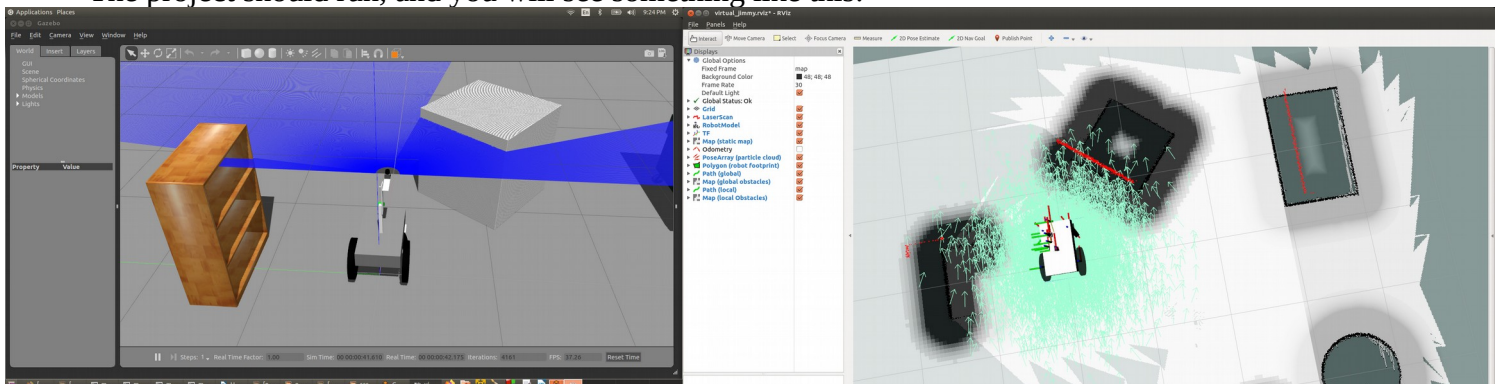
Launch File Name	Purpose / Demonstrated capability
<code>a_gazebo_teleop.launch</code>	Start Gazebo, load the turtlebot world and teleoperate the robot around the environment.
<code>b_gazebo_navigation_map_building.launch</code>	Start Gazebo, load the turtlebot world, launch gmapping and teleoperate the robot around the environment. As you teleoperate the robot you will build a map of the environment. If you want to save your map, open a new terminal and run the command: <code>\$ rosrun map_server map_saver -f ~/PATH/workspace/src/jimmy_navigation/maps/MAP_NAME</code> replace PATH with the path on your PC, and MAP_NAME with the name you want to give to the map.
<code>c_gazebo_navigation_on_map.launch</code>	Start Gazebo, load the turtlebot world and fire up move_base (which is the core navigation package). Here you can give a “2D Nav Goal” in Rviz and watch the robot move in Gazebo. Use these tools to figure out what is going on: <ul style="list-style-type: none">• <code>\$ rqt_graph</code>• <code>\$ rostopic list</code>• <code>\$ rostopic echo /topic_name</code>
<code>d_physical_teleop.launch</code>	Teleoperate the physical robot.
<code>e_physical_navigation_map_building.launch</code>	Teleoperate the physical robot, and build a map of the environment.
<code>f_physical_navigation_on_map.launch</code>	Send navigation goals via Rviz to the physical robot, and watch it navigate in the environment.

To run the project do the following in order:

NOTE: T1 means terminal 1, T2 is terminal 2, etc...

- T1: `$ cd Desktop/workspace`
- T1: `$ source devel/setup.bash`
- T1: `$ roslaunch jimmy_welcome a_gazebo_teleop.launch`

The project should run, and you will see something like this:



now try to run the other Gazebo launch files.

Getting a Feel for the Code

Each package in this project is described below. Have a look over this before trying to read the code.

Package Name	Description
jimmy_base	This package is using the ros_control library. The main reason for using the ros_control library is because it offers a diff_drive plug-in (convert /cmd_vel to rad/sec commands for each driver wheel) which allows the robot to handle the output of the navigation stack. In essence this package loads a diff driver, reads the status of the encoders on the driver wheels (via jimmy_cpp), updates all the controllers, and sends commands to each driver wheel (via jimmy_cpp). This package is very minimalistic to try to keep things simple.
jimmy_control	This package contains yaml files for the physical robot, and a virtual Gazebo robot. I kept this structure as other projects had it. Monkey see, money do.
jimmy_cpp	This is a library that other packages use to access the hardware on the robot: <ul style="list-style-type: none">• ampsteppers – this is a library to communicate with the Ethernet stepper drives that drive the wheels. The library accepts rad/sec and converts those values into commands the stepper driver understand.• arduinomega – library to allow communication with an Arduino over ethernet. The arduino provides the driver wheel encoder data.• lidar_magnetic_encoder – communication with an Arduino over the native usb to get the lidar position (using a magnetic encoder).• lidar_stepper_motion – this is communication to an Arduino via the native usb that is controlling pulses to a stepper driver to achieve lidar motion. This library does the accel, decel and speed control.• main – I think this is needed to build the project.• trc_logging - a logging library that I need to figure out how to utilize efficiently along ros messages. It writes to files and does all kinds of fancy stuff. Maybe I'll get rid of it...
jimmy_description	A package describing the robot via URDF files that are written using xacro.
jimmy_gazebo	A package that takes care of all the gazebo simulation items. It launches the worlds, the robot model, controllers, and so on to get Gazebo working.
jimmy_hokuyo_lidar_control	This package pulls 3 pieces of hardware (stepper motor axis, encoder, Hokuyo lidar) into a unified 3D lidar unit. It takes care of homing the lidar, putting it at a horizontal level, and nodding the unit in front or behind the robot.
jimmy_navigation	This directory contains all the launch files and configuration files for the navigation stack. It defines the global planner, local planner, topic names, etc...
jimmy_urg	Contains all the needed code to establish a serial connection with the Hokuyo urg device, to send commands, and to receive the lidar data. In essence this is a driver that pumps out lidar data. There are multiple items that exist in this directory to support the urg_node driver.
turtlebot3_teleop_modified	Allows teleoperation of the physical robot (or the virtual robot). I changed the name of the package as I modified various items and needed to make sure people know that it is not the same thing you can download online.