Using FRCSim with C++ and Java

Last Updated: 01-25-2017

Table of Contents

Setting up and running Gazebo

Introduction to FRCSim

With FRCSim, you should be able to finish 90% Of your programming without ever touching a RoboRIO.

We want you to be able to test your code **BEFORE** you put in on your robot, and **before the robot is even built.**

FRCSim allows robot code written in C++ or Java that normally runs on your RoboRIO to be run on your laptop or desktop. It connects to custom robot models in the Gazebo robot simulator.

How it works

The project as a whole is pretty simple. The C++ and Java libraries actually have two versions, one for the RoboRIO and one for simulation. They share a lot of the same code, but differ when it comes to things like setting motor speed and reading sensors.

Your Robot Program

When you compile or run in simulation, you are using the simulation versions of WPILib. We try to keep everything the same between the two versions, so that you never have to change your code in order to run in simulation. However, if you do need to change it, you can use the following constructs.

Java:

```
if (Robot.isReal()){ #Robot here is a class extending RobotBase
  // do something
}
else {
  // do something different
}
```

C++:

```
#ifdef REAL
     // do something
     #else
     // do something different
     #endif
```

Your Robot Model

Your robot is represented in Gazebo in a file format called SDF (https://sdformat.org)

SDF Is a complicated file format based on XML, but we built a solidworks exporter to help! We've got lots of tutorials on using it. You can start here: Installing SolidWorks Gazebo Exporter plugin

It is very difficult to make good models by hand, so please try the exporter.

Gazebo Transport

Gazebo Transport is the glue holding everything together. Similar to ROS (https://wiki.ros.org) Gazebo Transport is a library built of Google's Protobuf (https://developers.google.com/protocol-buffers/?hl=en) that allows messages to be sent back and forth between different programs.

More specifically, your robot program sends out and receives messages over Gazebo Transport. The driver station also sends messages through gazebo transport, and so does your robot model. In fact, all those gazebo plugins like dc_motor send and receive messages over Gazebo Transport.

There is a utility program galled **gz** that can help you see what's going on and help debug your system. Type **gz topic --help** to get some helpful information on how to use the program.

Video Tutorials

There is a series of simulation videos on this channel:

https://www.youtube.com/channel/UC_xEeMJeS3UQ388lxNItmzQ

2016 Field

A world file for the 2016 Field can be downloaded here!

first.wpi.edu/FRC/roborio/release/simulation

Installing Ubuntu

At this time, to run FRCSim, it is necessary to install Ubuntu (64bit). Any version supported by gazebo will work, but we suggest 14.04, the LTS version. A discrete NVidia or ATI graphics card is also recommended since these are the configurations that are most tested. From our experience Intel Integrated Graphics often does not work and should be avoided if possible. We recommend following this tutorial: https://help.ubuntu.com/community/Installation/FromUSBStickQuick

Make sure you download the amd64 iso. There are various other instructions online detailing how to install Ubuntu 14.04. If you find one you like better, email pdmitrano@wpi.edu.

Running Ubuntu on an SD Card, Flash Drive, or other External Drive

Installing Ubuntu on a flash drive is a great way compromise between dual-booting and a VM. While dual booting will be higher performance, a flash drive is much easier to set up than dual booting.You'll need at least 8gb of space, but 16 is highly recommended. If you run out of space you will be very sad.

Note: Installing Ubuntu on a flash drive is NOT the same as a "Live USB".

1. Create a live USB of your favorite linux distribution. We recommend Ubuntu 14.04 or later. The Live USB can be only 6 or 8gb, but you should install onto something larger, like 16 or 32 gb.

If you have Windows: http://www.ubuntu.com/download/desktop/create-a-usb-stick-on-windows

If you have OS X http://www.ubuntu.com/download/desktop/create-a-usb-stick-on-mac-osx

2. Insert the Live USB and the device you want to install on to and reboot your computer

3. If your computer reboots back into your other operating system, you may have to use F12 or F1 to interrupt the normal start-up sequence. Do some research, and don't be afraid to remove your existing hard drive if you are worried about losing your current operating system.

4. You'll know you booted into the correct device when you see something like the image below

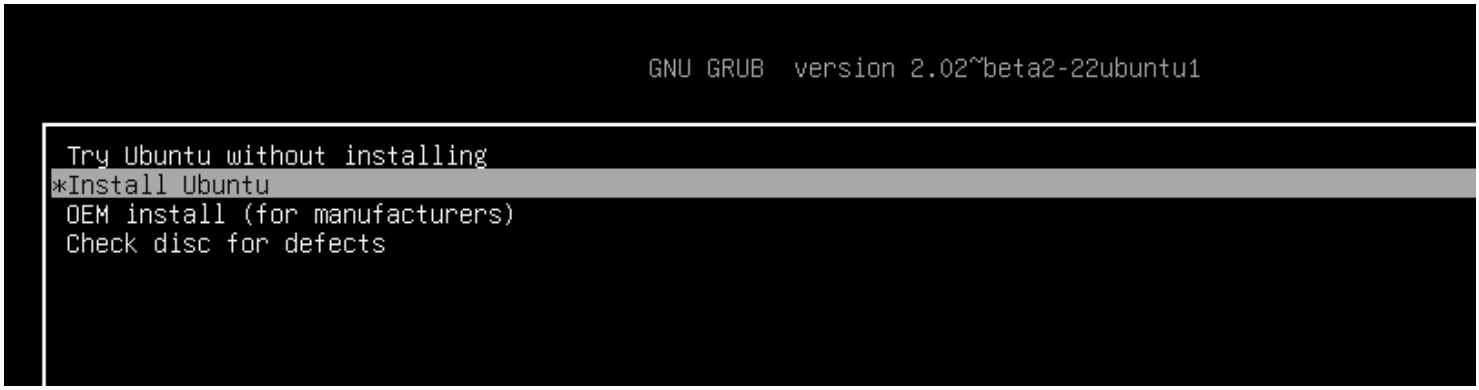
5. Follow the instructions under "Install Ubuntu". The default options are ok, just make sure you install onto the right device!

If you're having trouble, there are many fantastic guides on the internet for installing onto an SD Card or Flash Drive. We've created a video for you here:

https://www.youtube.com/watch?v=ShkwSN0RULc&list=PLhwpRbR67J4YWbKwwq5lje9dFiIHdBOm8&index=9

Another good tutorial is here:

https://www.youtube.com/watch?v=QdQ520dmg5g



Running Ubuntu on the Same Computer as Windows

It's possible to install Ubuntu alongside Windows, this is referred to as dual- booting. The instructions are similar to above, but you have to make sure that you don't delete your Windows install.

For more details see https://help.ubuntu.com/community/DualBoot.

Be careful, it is easy to mess up your bios or lose data when dual booting. Make a backup first!

Running in a Virtual Machine

A virtual machine is another viable option for using FRCSim. However, be warned that graphics performance will likely be a problem. We've found that VMWare player on windows, linux, or mac will run gazebo. However, gazebo will run very slowly.

Using FRCSim with C++ and Java

We have been unable to get Virtualbox to run gazebo, although on Mac it seems to work.

1. Download VMWare player

2. Download your favorite version of linux (Ubuntu 15.10 recommended) and install it in the virtual machine

3. Follow the directions here for manual install (Installing FRCSim The Fun Way (Manually)) and here for a script installer (TODO)

Note: We have tested with 8Gb of ram, 2 core, and 32Mb video memory, but feel free to try other settings

Installing FRCSim With A Script (Ubuntu)

The fast and easy way to setup FRCSim. I strongly recommend the manual install, because you will learn more and possibly avoid bugs in the script.

Please note, this only works on Ubuntu 14.04, and for other versions/systems please look at the manual install.

Open a Terminal

In Ubuntu, you can use Ctrl+Alt+T. Or, simply search for it in the sidebar.

Download And Run The Script

1. Copy the following command

```
wget -O frcsim-installer.sh first.wpi.edu/FRC/roborio/release/frcsim-installer.sh && chmod a+x frcsim-installer.sh && ./frcsim-installer.sh INSTALL
```

2. Paste in to the terminal

3. Hit enter

Install Eclipse Plugins

You'll need to install the usual WPILib Eclipse Plugins. Instructions for doing so can be found here:

installing eclipse plugins

Installing FRCSim Manually (Ubuntu)

FRCSim is best supported on Ubuntu 14.04, but you can run it on any platform supported by Gazebo. See http://gazebosim.org/tutorials For how to install. Gazebo does not yet run on windows.

Understanding Dependencies

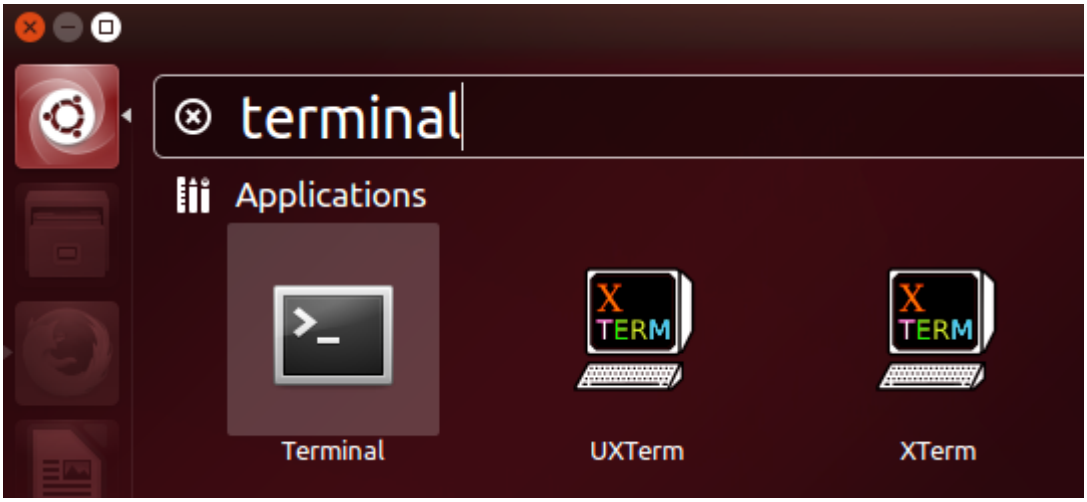
FRCSim is nothing more than a collection of existing software plus a few small libraries. In order to use FRCSim, you will need the following:

- Gazebo (5.0 or later)
- Eclipse (Mars or later)
- Gcc 4.9
- Java 8
- WPILib Eclipse Plugins
- WPILib Gazebo Plugins
- Google's Protobuf library

Even if you're team is only using C++, you still need java8.

Furthermore, even if you're team is only using Java, you still need g++-4.9

Open Terminal



1. Click on the Ubuntu icon (just below the top-left corner)

2. Type “terminal”

3. Click on the terminal application

Adding repositories to Apt

On Ubuntu 14.04, You'll need to tell Apt where to find the package you want. If you are using 15.10 you can skip this step

Run the following in your terminal, **but only the ones marked for you version of ubuntu**

```
sudo add-apt-repository ppa:openjdk-r/ppa -y #ONLY FOR 14.04 and 15.05
sudo add-apt-repository ppa:ubuntu-toolchain-r/test -y #ONLY FOR 14.04
sudo apt-get update
```

The first PPA is for java, and the second is for g++-4.9.

If you intend to also compile C++ programs for ARM as well as for Simulation, you'll need to install the C++ toolchain. See Installing Eclipse (C++/Java)

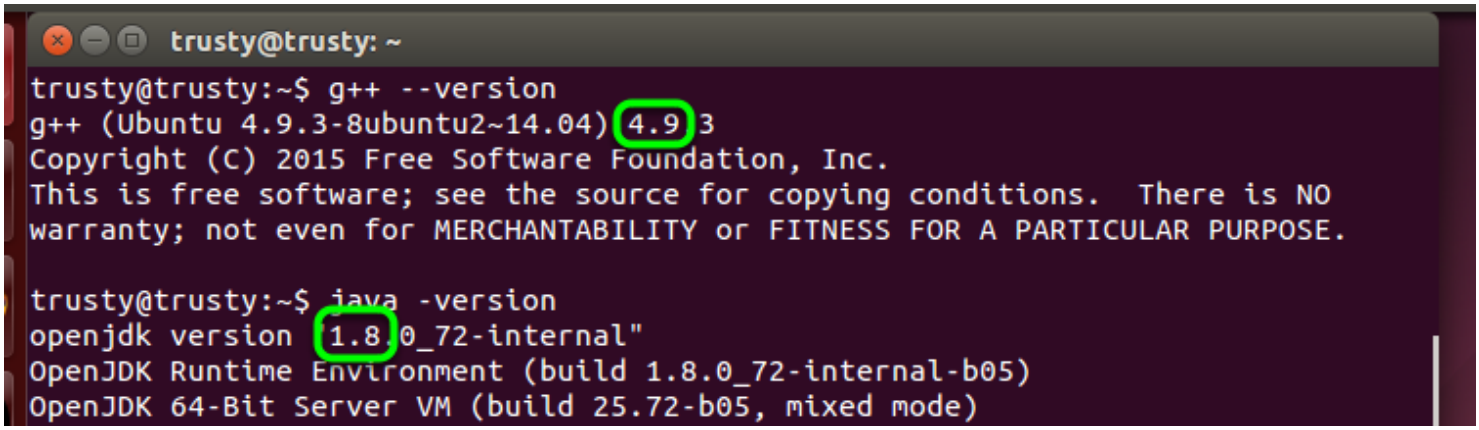
Install Dependencies From Apt

Install a bunch of our dependencies from apt. Run this in the terminal.

```
sudo apt-get install g++-4.9 openjdk-8-jdk
```

You can confirm that it worked by typing the following. The output should be similar to the image below.

```
g++ --version
java -version
```



Fixing Executable Locations and Names

For some versions of Ubuntu, you may run into issues where g++ --version gives you 4.8 instead of 4.9.

You can verify the issue by looking at the output of:

```
ls -l /usr/bin/g++
```

If it looks like this, then you've got a problem. Here it says the g++ actually points to g++-4.8, which is not what we want.

```
lrwxrwxrwx 1 root root 7 Jan  4 19:05 /usr/bin/g++ -> g++-4.8
```

To fix this, we simply run the following, which tells g++ we want 4.9, rather than 4.8.

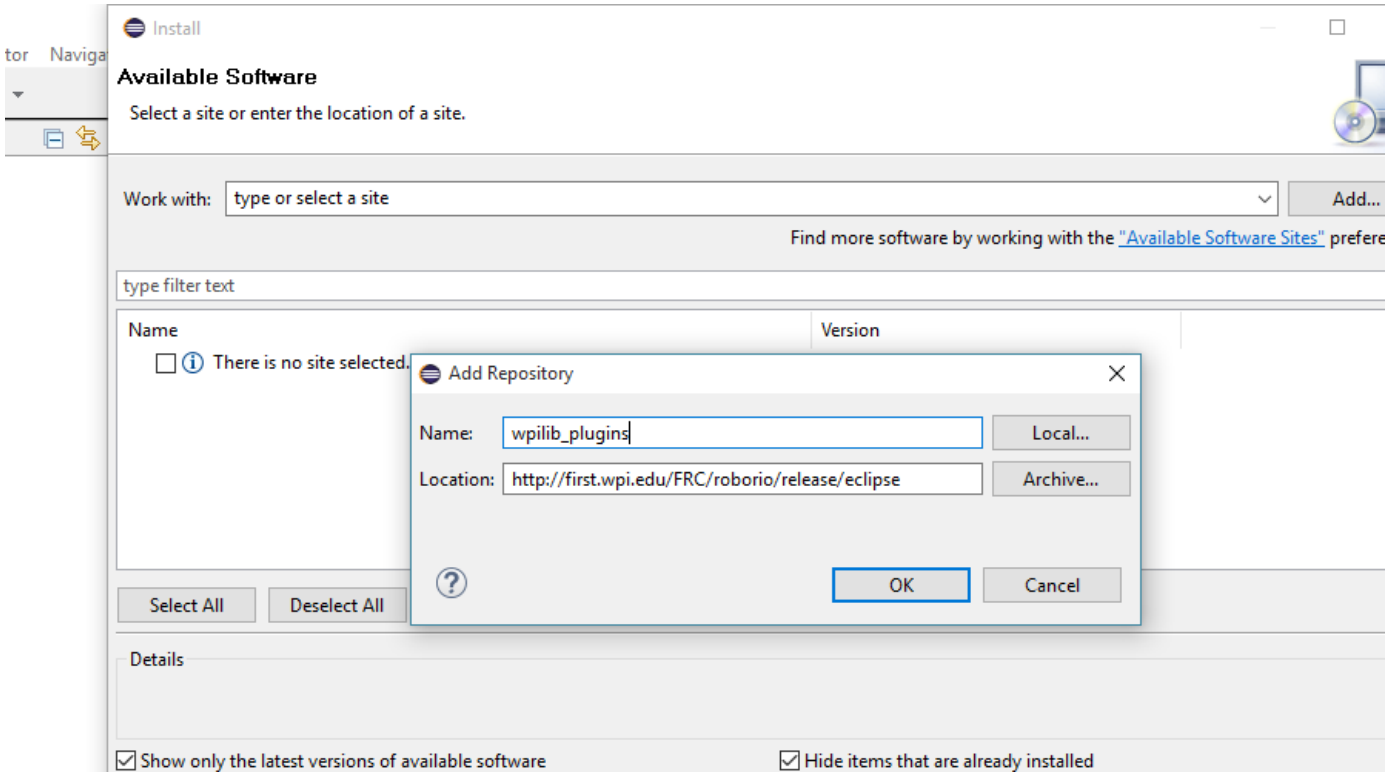
```
sudo rm -f /usr/bin/g++
sudo ln -s /usr/bin/g++-4.9 /usr/bin/g++
```

Install Eclipse

Installing eclipse is a little more work, since we want the latest version. Use the following commands to download and install.

```
wget -O eclipse.tar.gz \
       http://eclipse.mirror.rafal.ca/technology/epp/downloads/release/mars/1/eclipse-cpp-mars-1-linux-gtk-x86_64.tar.gz
tar -xf eclipse.tar.gz
sudo mv eclipse /opt
sudo ln -s /opt/eclipse/eclipse /usr/bin/eclipse
```

Install WPILib Eclipse Plugins



1. add a new software site http://first.wpi.edu/FRC/roborio/release/eclipse

2. Continue with install, and restart eclipse.

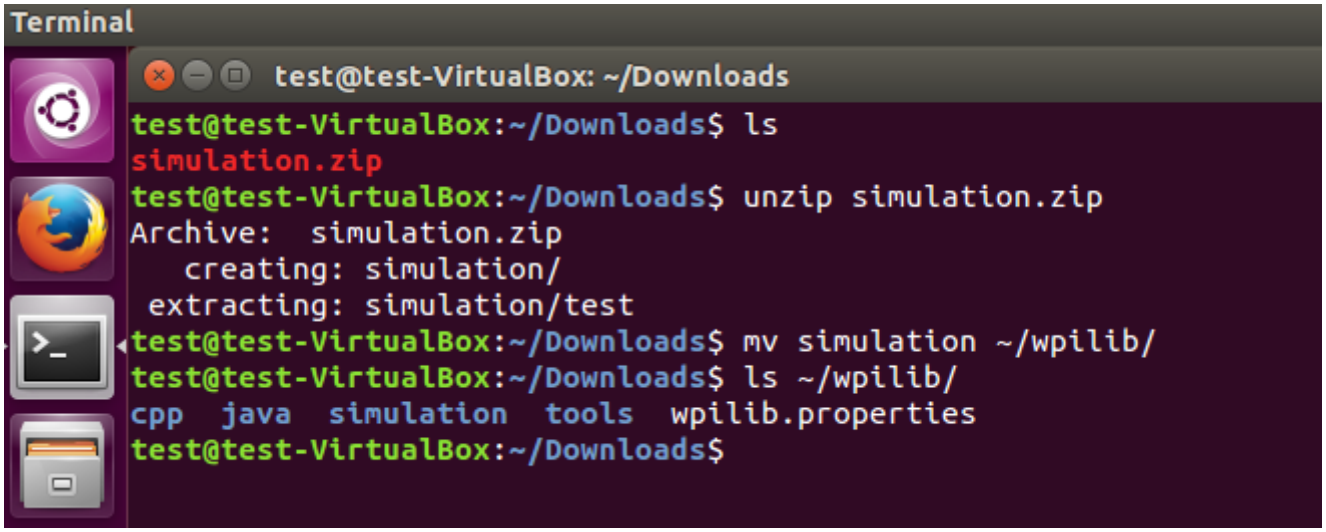
See Installing Eclipse (C++/Java) for more detailed instructions

Install Gazebo

See the gazebo website (https://gazebosim.org) for detailed instructions, or just run this in the terminal.

```
wget -O /tmp/gazebo6_install.sh http://osrf-distributions.s3.amazonaws.com/gazebo/gazebo6_install.sh; sudo sh /tmp/gazebo6_install.sh
```

Install WPILib Gazebo Plugins



The last dependency is the gazebo plugins and example models. These will be distributed in the following zip file: http://first.wpi.edu/FRC/roborio/maven/release/edu/wpi/first/wpilib/simulation/simulation/1.0.0/simulation-1.0.0.zip

1. Download the zip

2. make a directory called simulation in the wpilib directory

```
mkdir ~/wpilib/simulation
```

3. Extract it to that new simulation folder

```
unzip ~/Downloads/simulation-1.0.0.zip -d ~/wpilib/simulation
```

4. Make sure you have installed the development package for gazebo. This should already be installed if you used the gazebo script above.

```
sudo apt-get install libgazebo6-dev
```

Build Gz_Msgs

You will need to compile and install gz_msgs, which is provided in the simulation.zip you just downloaded. This is so that same zip can work with many versions of protobuf and ubuntu.

Enter the **~/wpilib/simulation/gz_msgs** directory and run the following commands:

```
sudo apt-get install cmake libprotobuf-dev libprotoc-dev protobuf-compiler -y
mkdir build
cd build
cmake ..
make install
```

You should now have libgz_msgs.so in ~/wpilib/simulation/lib and a few new header files in ~/wpilib/simulation/include/simulation/gz_msgs

Install Sample Robot Models

The field and game piece models for 2016 will be available on January 12th

There are a few sample models and worlds that will be useful when you develop your own custom robots. Models can be downloaded here (https://usfirst.collab.net/sf/frs/do/downloadFile/projects.wpilib/frs.simulation.frcsim_gazebo_models/frs1160?dl=1) and unzip into the ~/wpilib/simulation directory.

```
wget -O models.zip \
 https://usfirst.collab.net/sf/frs/do/downloadFile/projects.wpilib/frs.simulation.
frcsim_gazebo_models/frs1160?dl=1
unzip models.zip
mv  frcsim-gazebo-models-4/models ~/wpilib/simulation/
mv  frcsim-gazebo-models-4/worlds ~/wpilib/simulation/
```

Configuring your Development Environment

Now that you have all the dependencies, there are a few things you can do to make life easier. These steps are **optional** but recommended for your sanity.

1. Allow the executables to be run anywhere by making symbolic links to your /usr/bin directory

```
sudo ln -s ~/wpilib/simulation/frcsim /usr/bin/frcsim
sudo ln -s ~/wpilib/simulation/sim_ds /usr/bin/sim_ds
```

2. Pin frcsim, sim_ds, and eclipse to your taskbar. Even better, you can move the .desktop files for frcsim, sim_ds, and eclipse into the proper folder.

```
sudo mv ~/wpilib/simulation/frcsim.desktop /usr/share/applications
sudo mv ~/wpilib/simulation/sim_ds.desktop /usr/share/applications
sudo mv ~/wpilib/simulation/eclipse.desktop /usr/share/applications
```

3. Setup the icon for sim_ds. Run the following

```
sudo mkdir /usr/share/icons/sim_ds
sudo cp ~/wpilib/simulation/sim_ds_logo.png /usr/share/icons/sim_ds/
```

Ok, now that you have everything setup, continue to see how to build robots models and run them in simulation!

Video Walkthrough

A video walk-through of this process is available here:

https://www.youtube.com/watch?v=2pmtXLYYnKc&list=PLhwpRbR67J4YWbKwwq5lje9dFiIHdBOm8&index=1

Installing FRCSim Manually (Mac OSX)

This guide shows how to install and test FRCSim on Mac OSX.

This only works for java!

Install Eclipse

Download and install the Mars version of Eclipse for Java developers. After installing, to start Eclipse open a Terminal and type:

```
eclipse
```

Install WPILib Eclipse Plugins

Follow the instructions here on how to install WPLib plugins

Install Homebrew

Unfortunately the one thing Mac OSX is missing is a package manager (e.g. Ubuntu apt-get). There are several open source package managers available, but I usually use Homebrew. The instructions can be found on the web but basically involves one simple command (provided you already have Ruby installed):

```
ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

Install wget, unzip, and cmake

The UNIX command wget is handy for fetching content from the web. While there are alternatives on Mac, I prefer to install wget using Homebrew. We will also install unzip and cmake while we are at it.

```
brew install wget
brew install homebrew/dupes/unzip
brew install cmake
```

Install XQuartz

Gazebo uses UNIX X-Windows, and so you will need an open source version of X. XQuartz provides a version of X-Windows complete with a packaged Mac install.

Download and install XQuartz.

Install Gazebo

Gazebo is your robot simulator. Using Homebrew to install Gazebo is as simple as:

```
brew tap osrf/simulation
brew install gazebo6
```

To verify it runs, execute the following command from Terminal after the install completes:

```
gazebo
```

Install Gazebo Plugins

WPI provides a prepackaged download of the worlds, models and plugins required to simulate robot samples from previous years. The download has a few Ubuntu specific components which we will address below. But let's start by getting it installed. First, browse to http://first.wpi.edu/FRC/roborio/maven/release/edu/wpi/first/wpilib/simulation/simulation/ and locate the latest release (2017.2.1 at the time of writing), click the folder and locate the .zip file, then right click and "Copy link location". You will use this link in place of "ZIP_URL" below.

```
cd ~/Downloads
wget ZIP_URL
mkdir ~/wpilib/simulation
unzip ~/Downloads/simulation-1.0.0.zip -d ~/wpilib/simulation
```

Building GZ Msgs

To compile and install gz_msgs for our platform, follow these steps:

```
cd ~/wpilib/simulation/gz_msgs
mkdir build
cd build
cmake ..
make install
```

I received a few compiler warnings on building that were safely ignored.

Install Sample Robot Models

These instructions directly parallel their Ubuntu equivalent.

```
cd ~/wpilib/simulation
wget -O models.zip https://usfirst.collab.net/sf/frs/do/downloadFile/projects.wpilib/frs.simulation.frcsim_gazebo_models/frs1160?dl=1
unzip models.zip
mv  frcsim-gazebo-models-4/models ~/wpilib/simulation/
mv  frcsim-gazebo-models-4/worlds ~/wpilib/simulation/
```

Build FRC Gazebo Plugins for Mac OSX

The plugins you unzipped to ~/wpilib/simulation/plugins are compiled for Ubuntu Linux. You will need to replace these with Mac versions.

Compiling your own versions

```
cd ~/wpilib/simulation
mkdir allwpilib
git clone https://usfirst.collab.net/gerrit/p/allwpilib.git
cd allwpilib
./gradlew -PmakeSim frc_gazebo_plugins
cp ~/wpilib/simulation/allwpilib/build/install/simulation/plugins/* ~/wpilib/simulation/plugins
```

Enable Joystick Controllers

If you have a game controller you'd like to use with your simulator, you will need the Mac OSX versions of the jinput.jar. You can do this by downloading the latest jinput build, and moving the files (jinput.jar, jinput-osx.jnilib, jinput-test.jar) into the ~/wpilib/simulation/lib directory. For example:

```
cp ~/Downloads/*jinput* ~/wpilib/simulation/lib
```

Add Simulation to the Default Path

You will find it convenient to have ~/wpilib/simulation in the path. To do this:

```
open ~/.bash_profile
```

Add the following line at the end of your bash_profile:

```
export PATH=$HOME/wpilib/simulation:$PATH
```

Save and reload your environment to ensure the current terminal has the right path:

```
source ~/.bash_profile
```

Test Simulator with GearsBot

Follow these steps to test the robot simulator using the GearsBot sample:

Edit the Model

The robot models you installed are unfortunately hard-coded to the Ubuntu plugins. To run a sample, you will need to perform a one-time edit to change the file names for the libraries. To do this for the GearsBot sample, use a text editor to edit the file ~/wpilib/simulation/models/GearsBot/model.sdf, and replace all ".so" references with ".dylib". This will ensure the model looks to the appropriate libraries.

Using FRCSim with C++ and Java

Create an Eclipse Project

Start Eclipse:

```
eclipse
```

Click File -> New -> Other from the menu and type "robot" the search bar. Choose "Example Robot Project" and click Next, scroll to the bottom (CommandBased Robot), select "GearsBot" and click Next.

You will be asked for your team number, which it will use in your Java package path. Give the project a name (e.g. GearsBotTest) and click Finish.

Prevent the Fixed Plane Defect

There is a defect that will quite likely be fixed by the time you follow these instructions. In my case I was seeing the robot go below the fixed plane when run in autonomous mode (it didn't snap to the plane). But just in case, search your project for any reference to "usePIDOutput", and change all references of this:

```
protected void usePIDOutput(double d) {
  motor.set(d);
}
```

to

```
protected void usePIDOutput(double d) {
  if (!Double.isNaN(d)) {
    motor.set(d);
  }
}
```

Start FRCSim

Let's start the GearsBot world from a terminal:

```
cd ~/wpilib/simulation
./frcsim worlds/GearsBotDemo.world
```

Start the Driver Station

Let's start the Driver Station from another terminal:

```
cd ~/wpilib/simulation
./sim_ds
```

Run the Eclipse Project

Now let's get the GearsBot project working. From Eclipse click on Run -> Run and choose "WPILib Java Simulation" as your target. This will start the code and initiate communication between the Gazebo simulator and Driver Station.

You can test using autonomous mode by clicking on the Autonomous option in the Driver Station and clicking the Enable button.

If you have plugged in a game controller, you should also be able to enable Teleop mode to test manually driving around your sim world using your robot.

Simulating GearsBot with FRCSim

This tutorial walks you through running the GearsBot example, assuming you already have FRCSim installed.

A demo video of the end result can be found on the official WPILib channel here:

https://www.youtube.com/watch?v=LEJaTjNZxQI

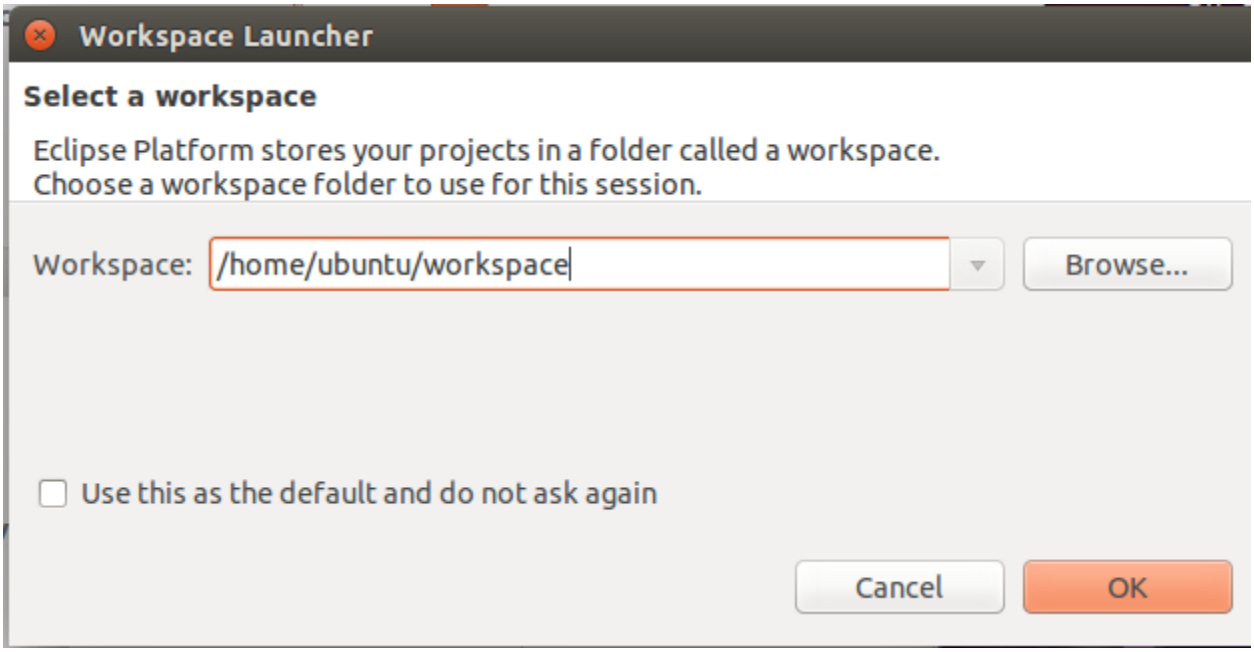
Open Eclipse

1. If you correctly followed the last optional steps in the article Installing FRCSim The Fun Way (Manually), then you can simply type "eclipse" in the launcher. Otherwise, you will have to open a terminal and run eclipse.

2. Type “eclipse”

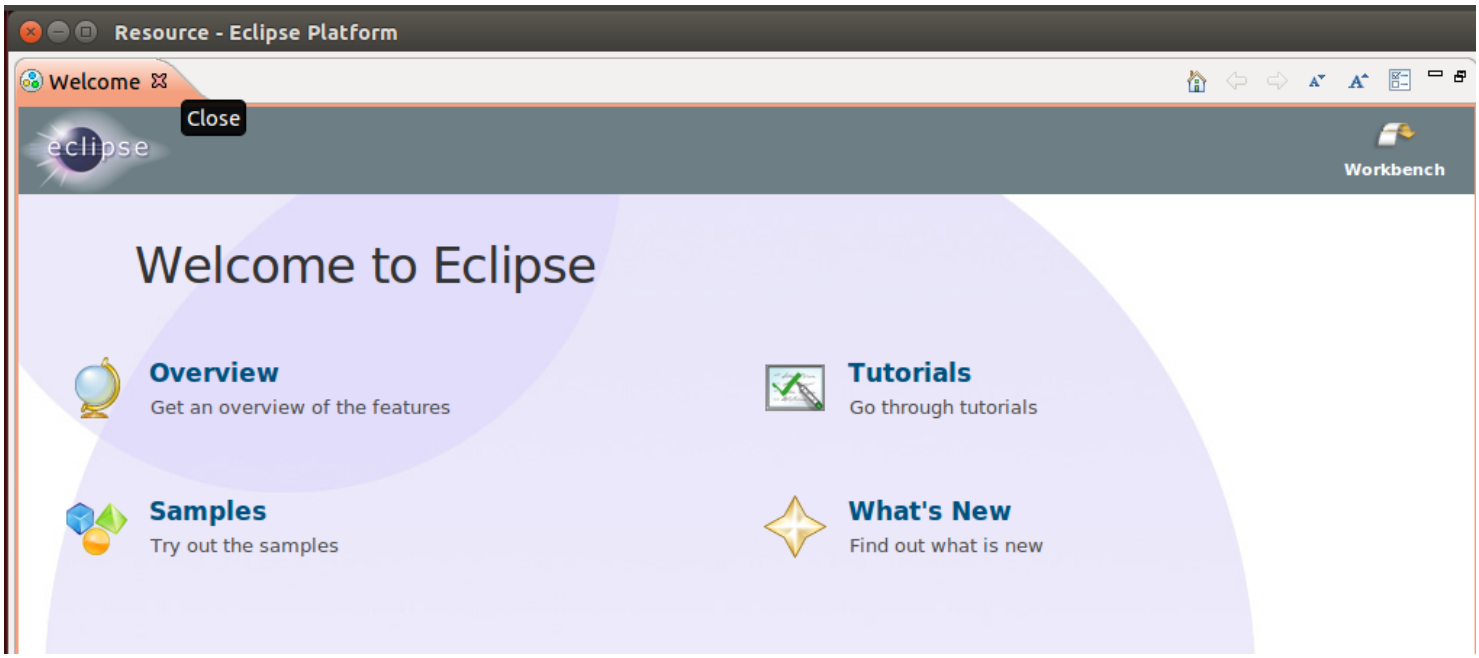
3. Click on the eclipse icon

Open the workspace



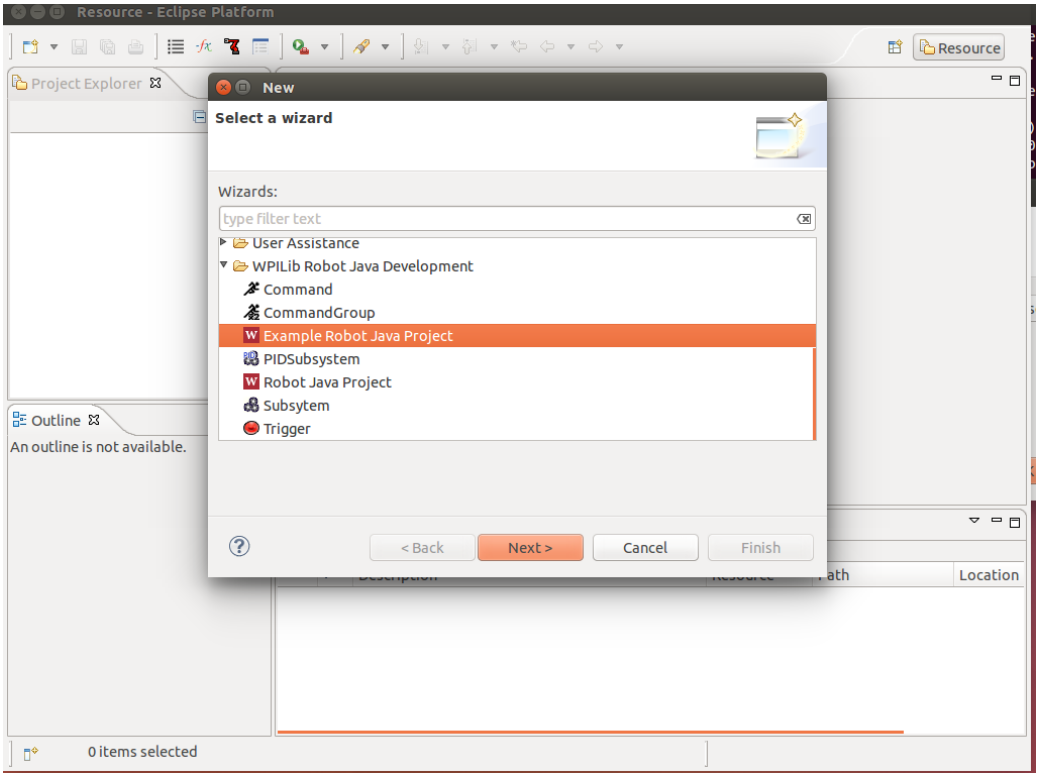
Click OK

Welcome to Eclipse



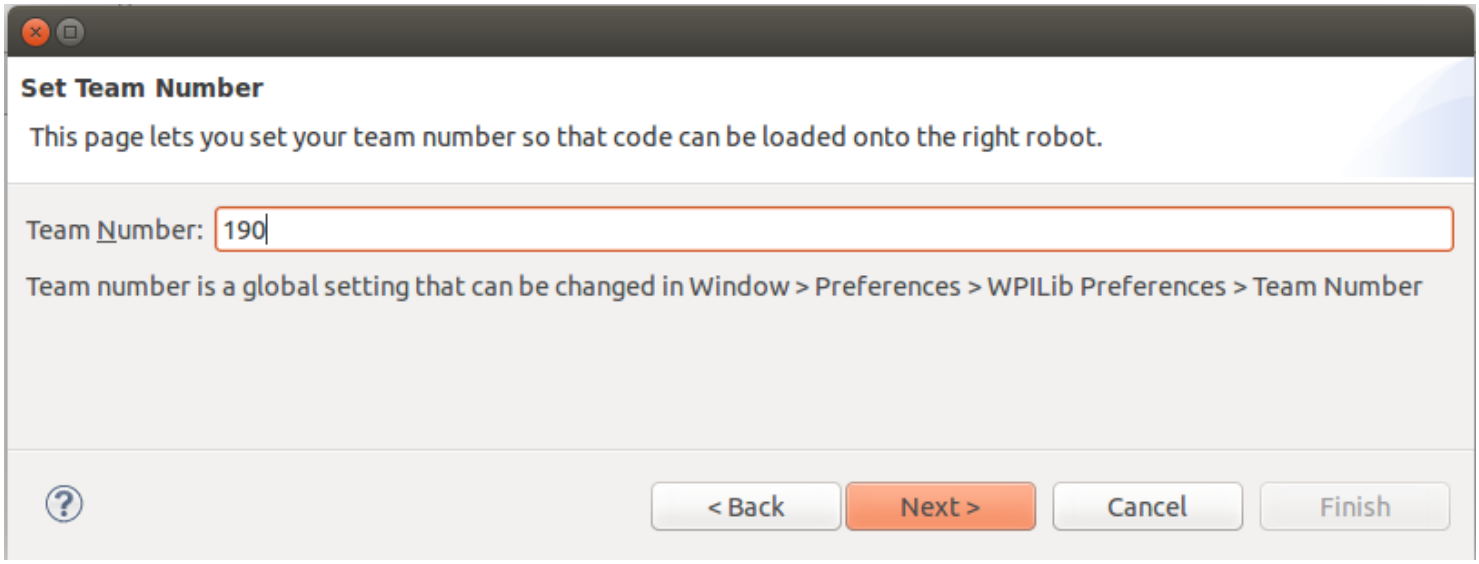
Exit the welcome menu

Start a new project



1. Click New Project
2. If using Java
   (a) Expand WPILib Robot Java Development
   (b) Select Example Robot Java Project
3. If using C++
   (a) Expand WPILib Robot C++ Development
   (b) Select Example Robot C++ Project
4. Click Next

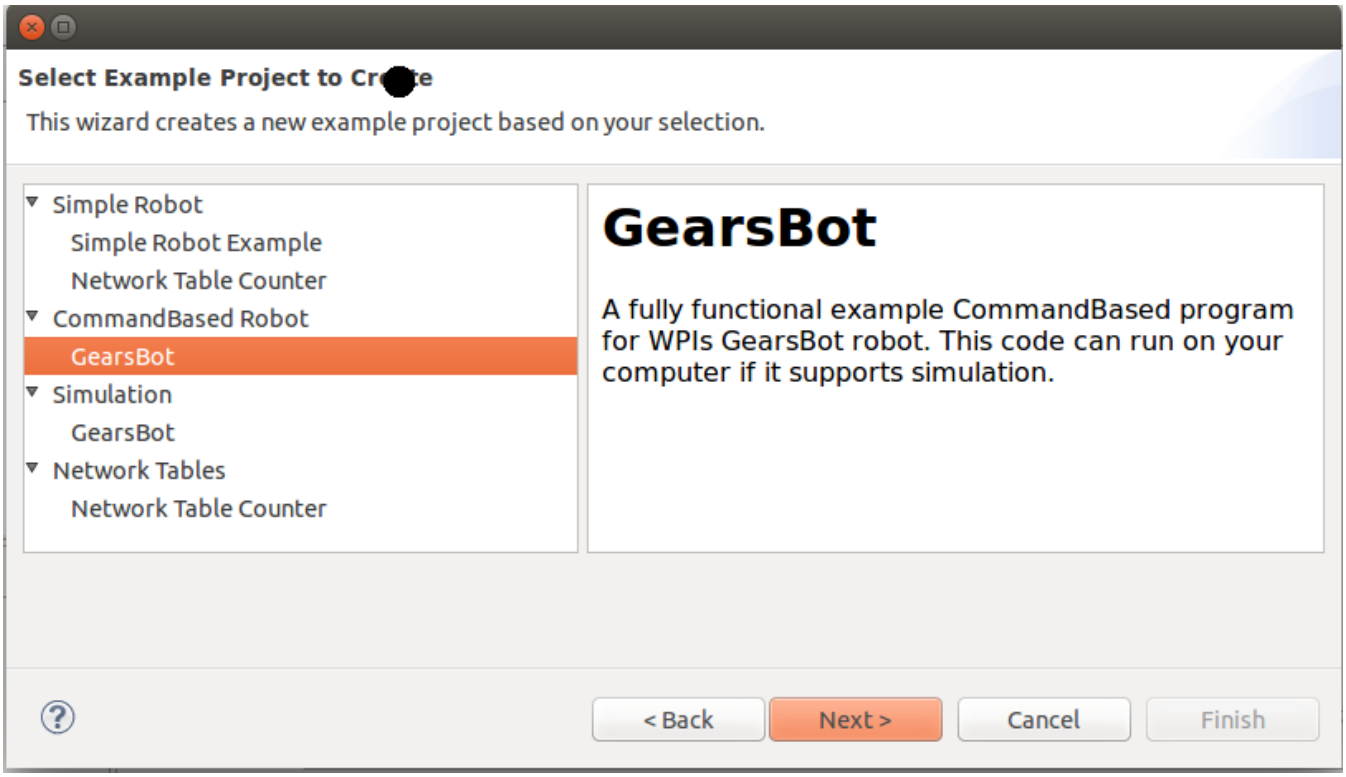
Set your team number



1. Enter your team number

2. Click Next

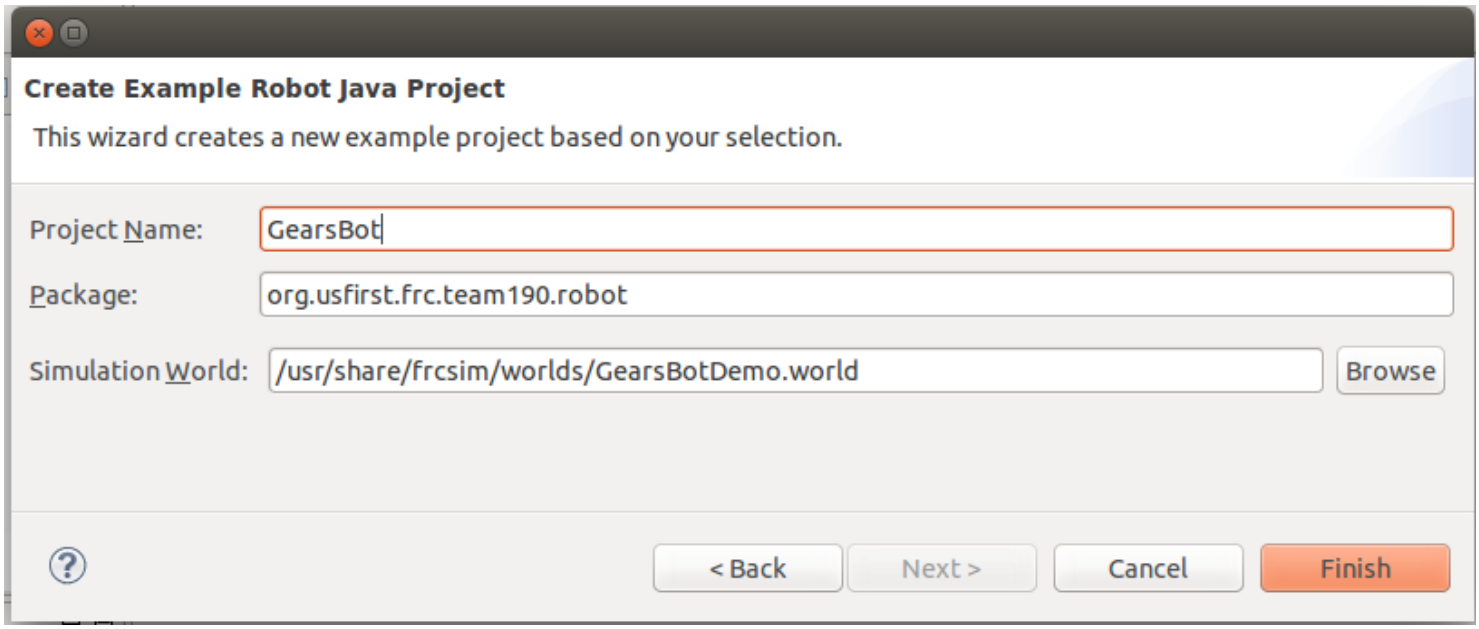
Choose the GearsBot example



1. Select the GearsBot example

2. Click Next

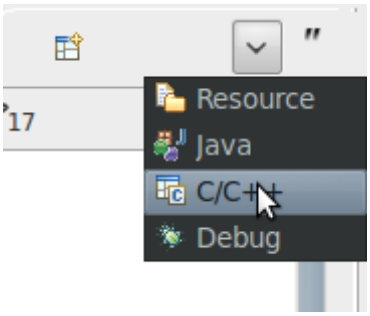
Naming the project



1. Type in a project name such as “GearsBot”

2. Click Finish

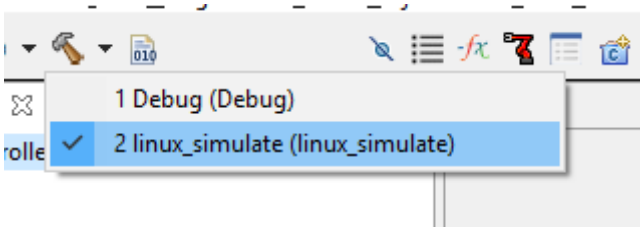
Switching to the C++ Perspective (for C++ programs only)



The C++ perspective sets up the eclipse environment with toolbars and menus specifically for developing C++ applications. To switch to the C++ perspective:

1. In the upper right corner click on the arrow.
2. Select C/C++

Switching to simulation compilation (for C++ programs only)



Selecting the drop-down menu (down-arrow) next to the "hammer" icon in the toolbar, you can select the default compilation target. Choose **Linux Simulate** to set the default for your robot simulation project.

Launching The Simulator

Running frcsim from the terminal is the better option because you can see error messages, and other information printed.

Open up a terminal and run the following.

```
frcsim ~/wpilib/simulation/worlds/GearsBotDemo.world
```

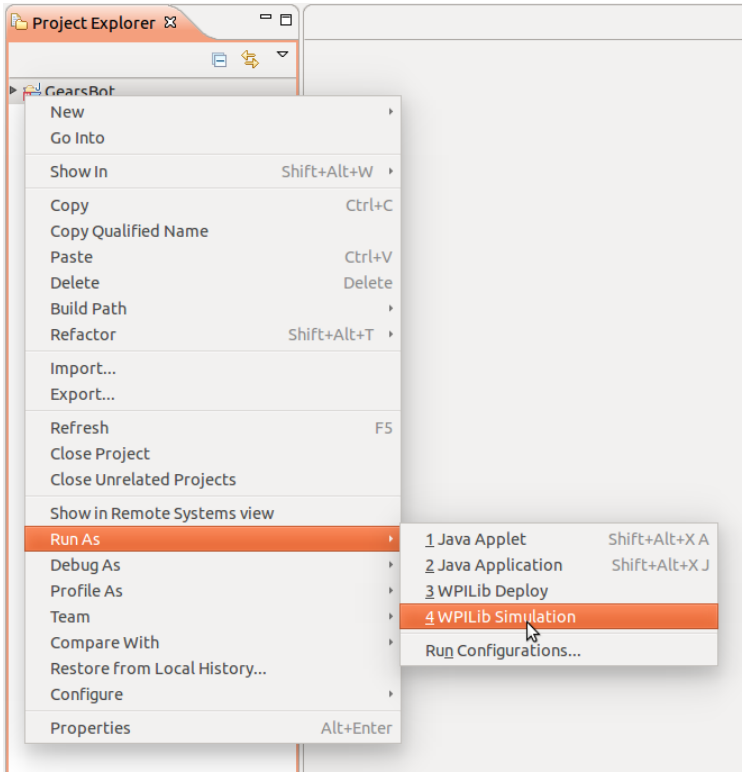
Launching the DriverStation

If you created a sim_ds.desktop file as shown in Installing FRCSim The Fun Way (Manually) simply open the launcher and type sim_ds.

Or, open a terminal and run:

```
sim_ds
```

Running Your Code

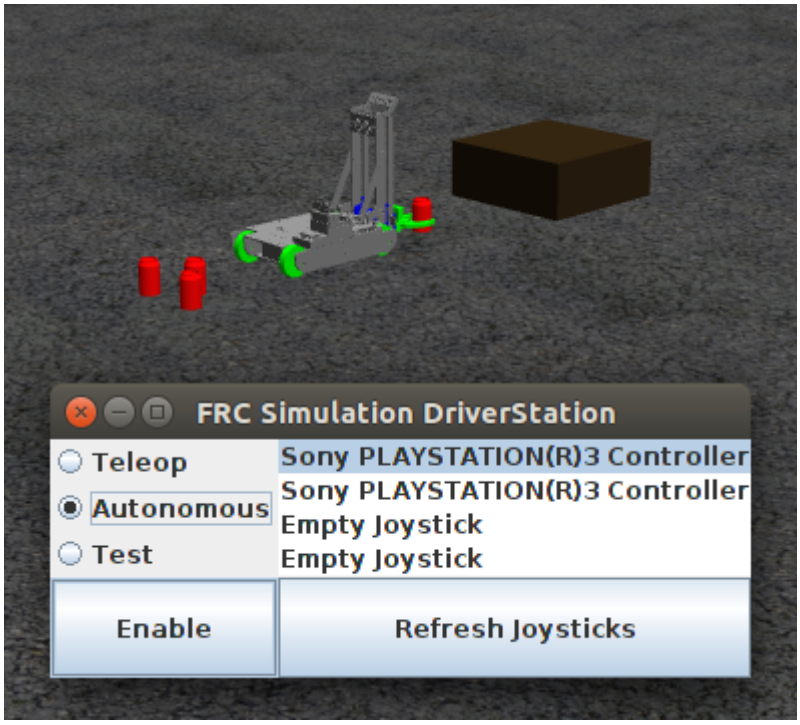


1. Right click on the project

2. Select Run As

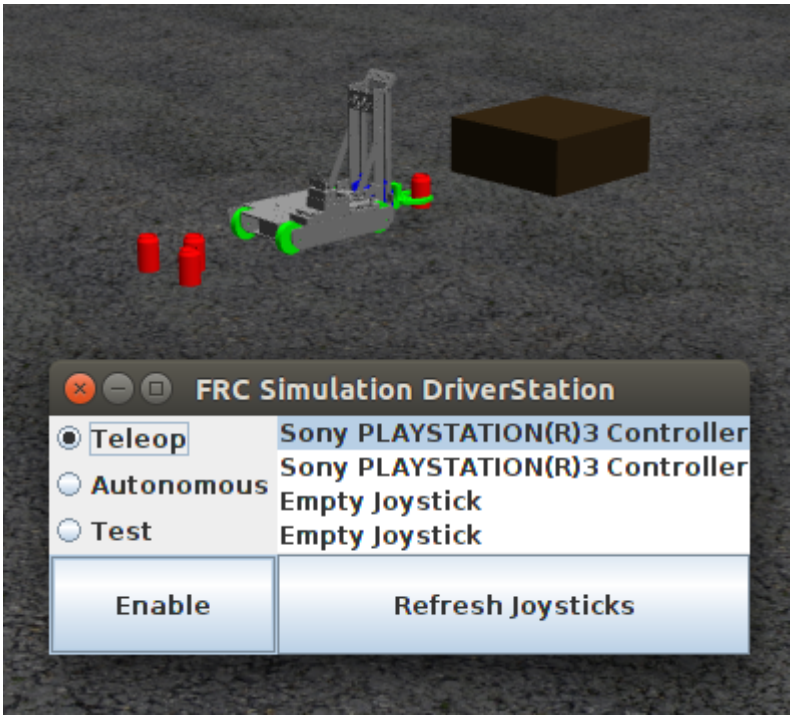
3. Click WPILib Simulation

Running Autonomous



1. Make sure the simulator is running
2. Select Autonomous
3. Click Enable
4. You should observe something like this video.

Using a PS3 joystick to drive



1. Plugin a joystick (preferably a PS3 SIX AXIS/DUALSHOCK) Start the simulator
2. Select Teleop
3. Click Enable
4. Drive around and pickup soda cans (button mapping available in appendix)

Simulating PacGoat with FRCSim

This tutorial walks you through running the PacGoat example, assuming you already have FRCSim installed.

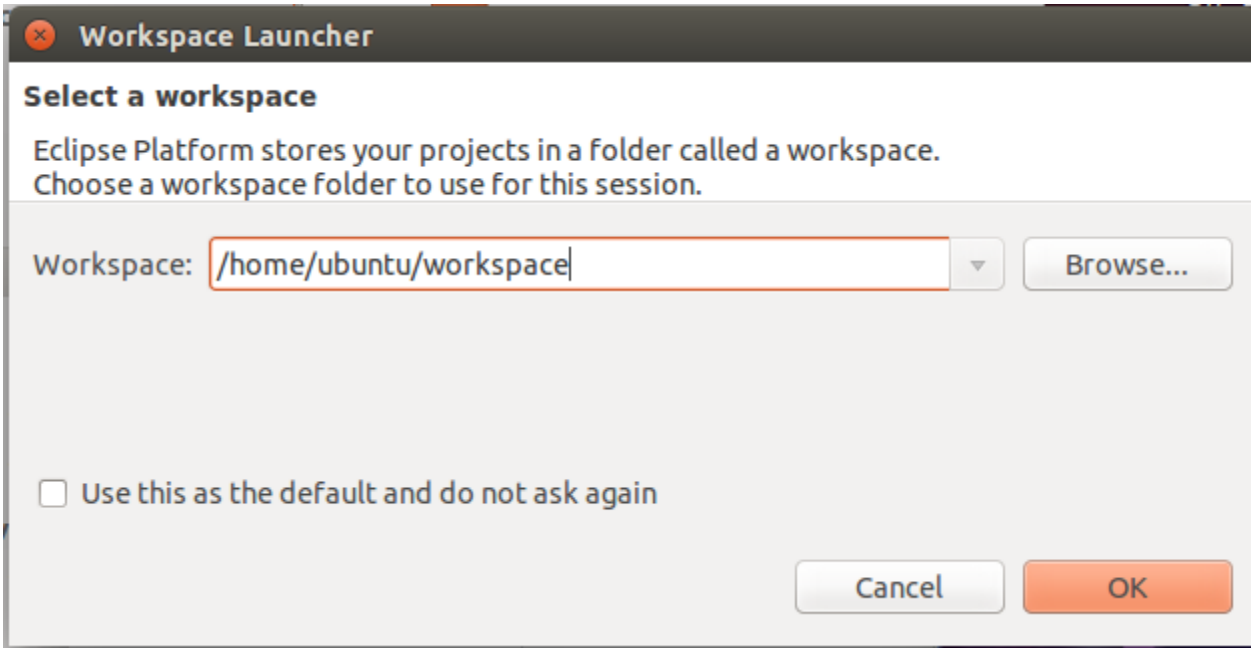
Open Eclipse

1. Click on the menu

2. Type “eclipse”

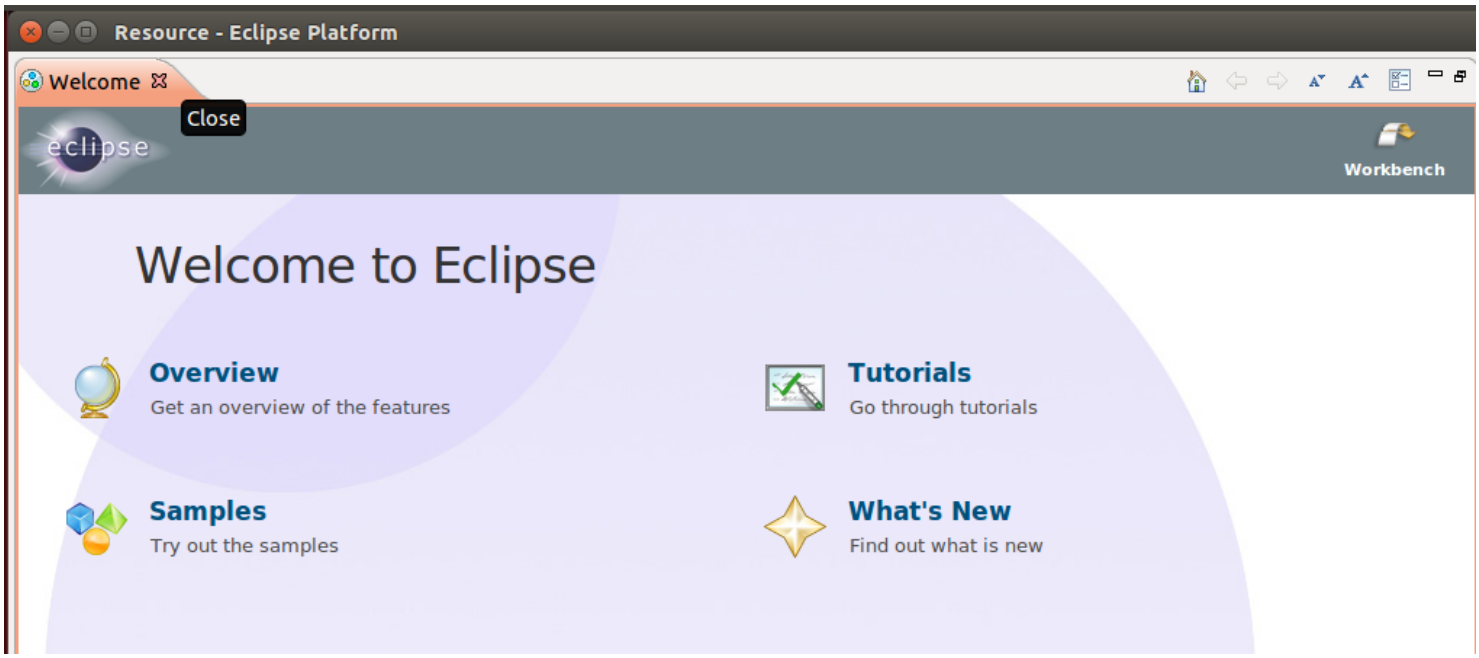
3. Click on the eclipse icon

Open the workspace



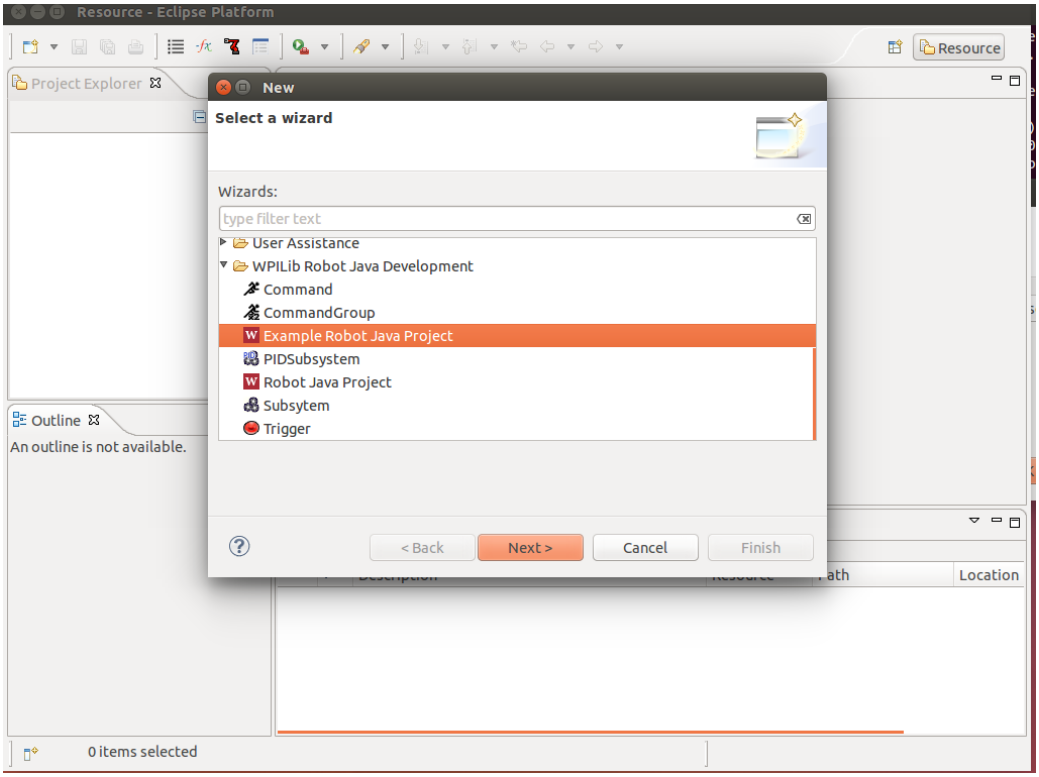
Click OK

Welcome to eclipse



Exit the welcome menu

Start a new project



1. Click New Project

2. If using Java

   (a) Expand WPILib Robot Java Development 13(b) Select Example Robot Java Project

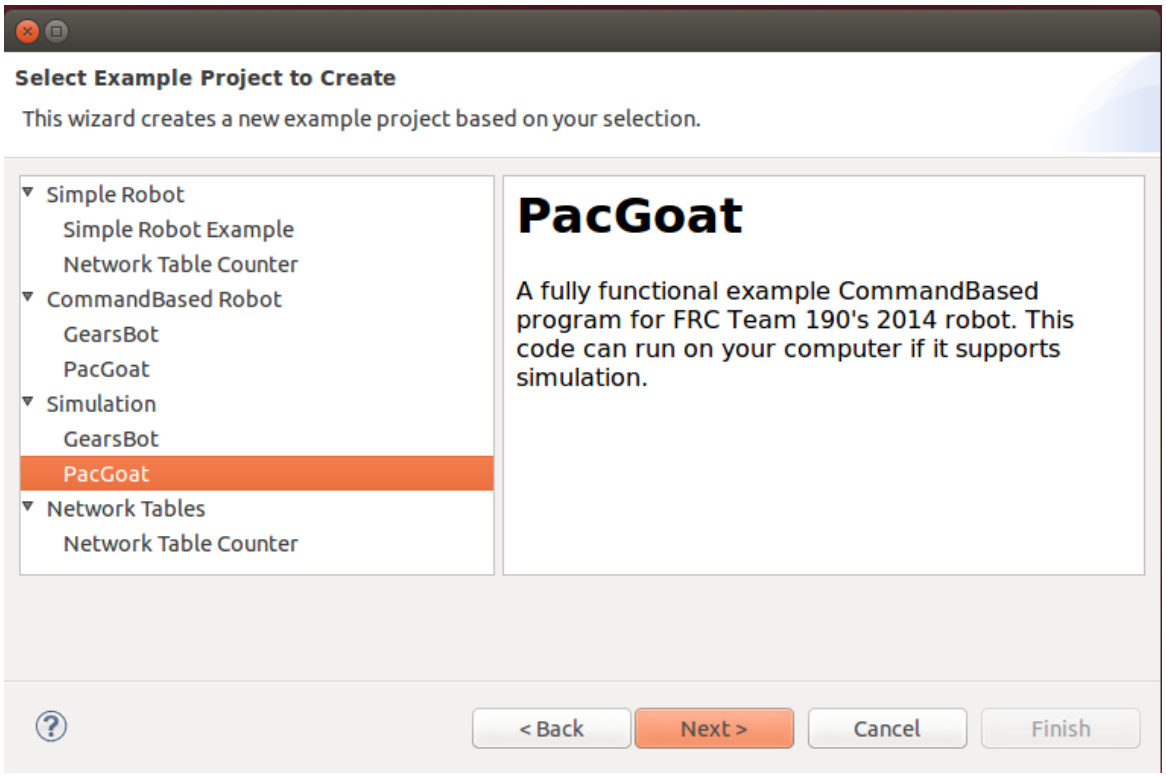
3. If using C++

   (a) Expand WPILib Robot C++ Development

   (b) Select Example Robot C++ Project

4. Click Next

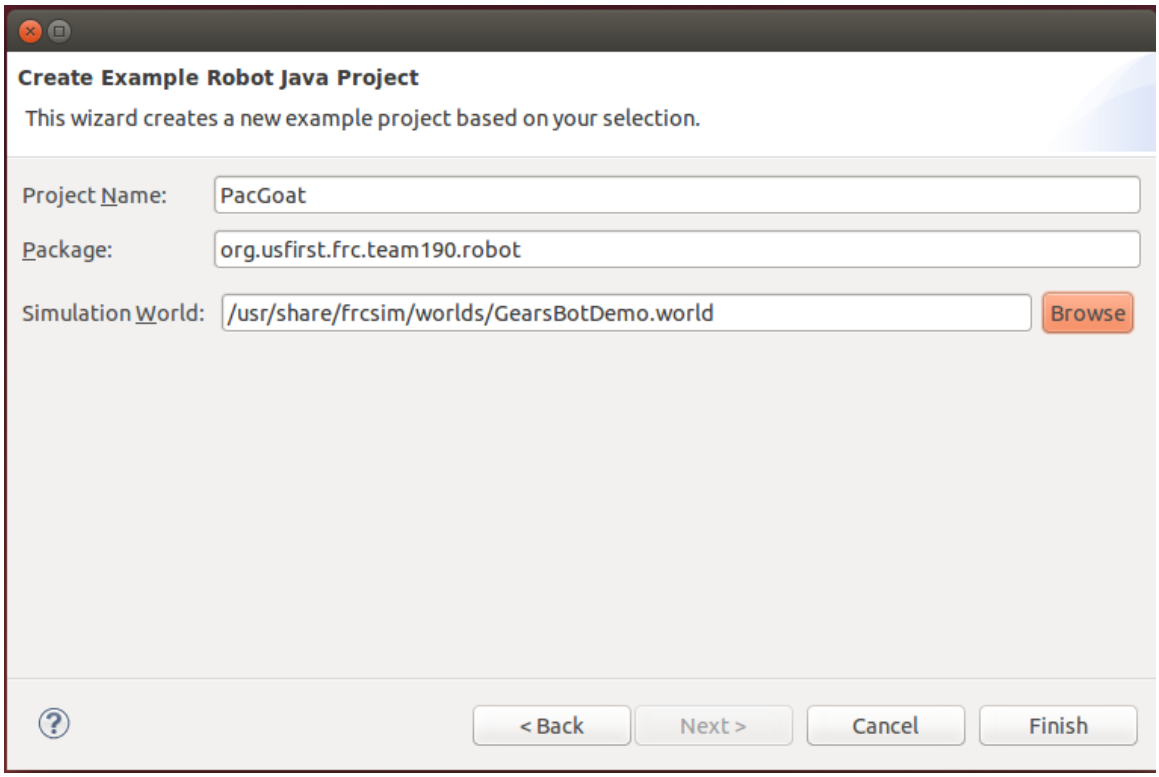
Choose the PacGoat example



1. Select the PacGoat example

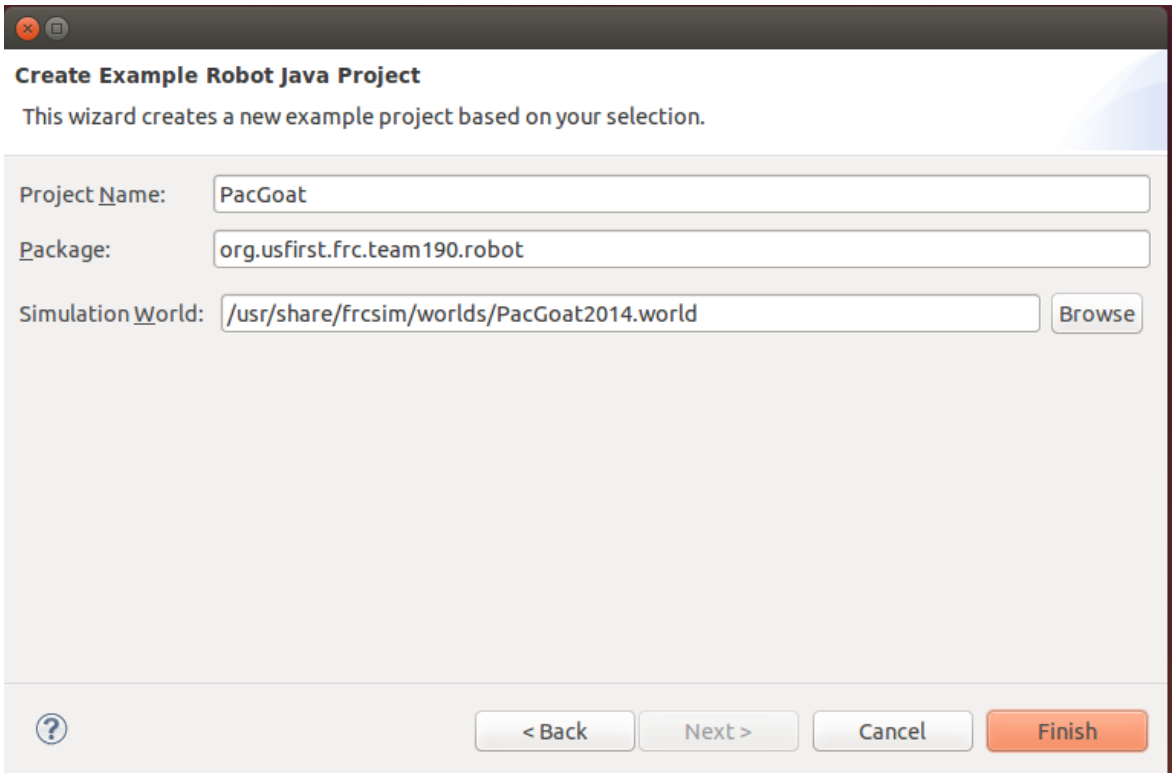
2. Click Next

Naming the project



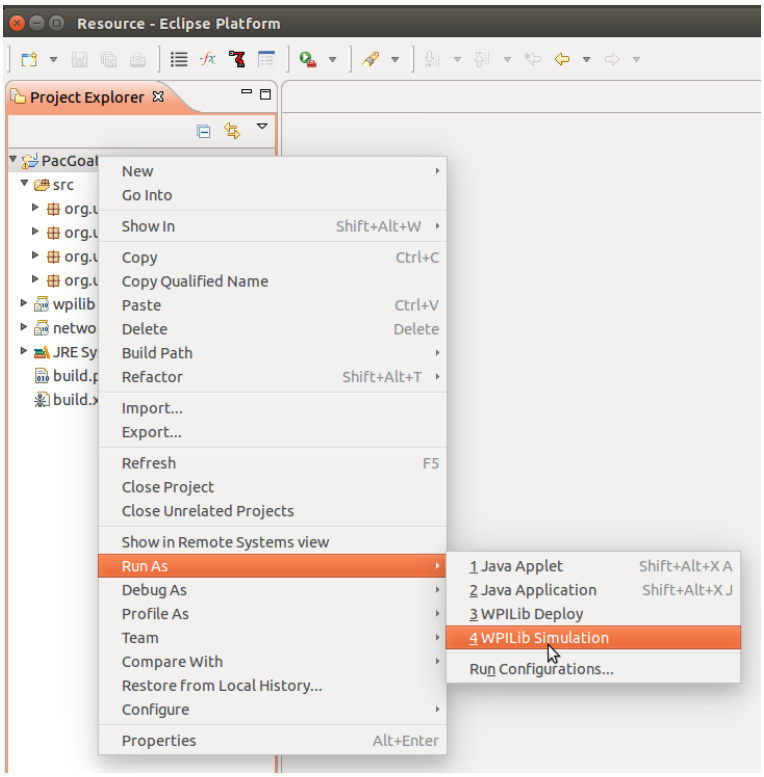
Type in a project name such as “PacGoat”

Finish



Press Finish

Running the Your Robot Program



1. Right click on the project

2. Select Run As

3. Click WPILib Simulation

Running the Simulator and Driver Station

1. Open a terminal and run

```
frcsim ~/wpilib/simulation/worlds/PacGoat2014.world
```

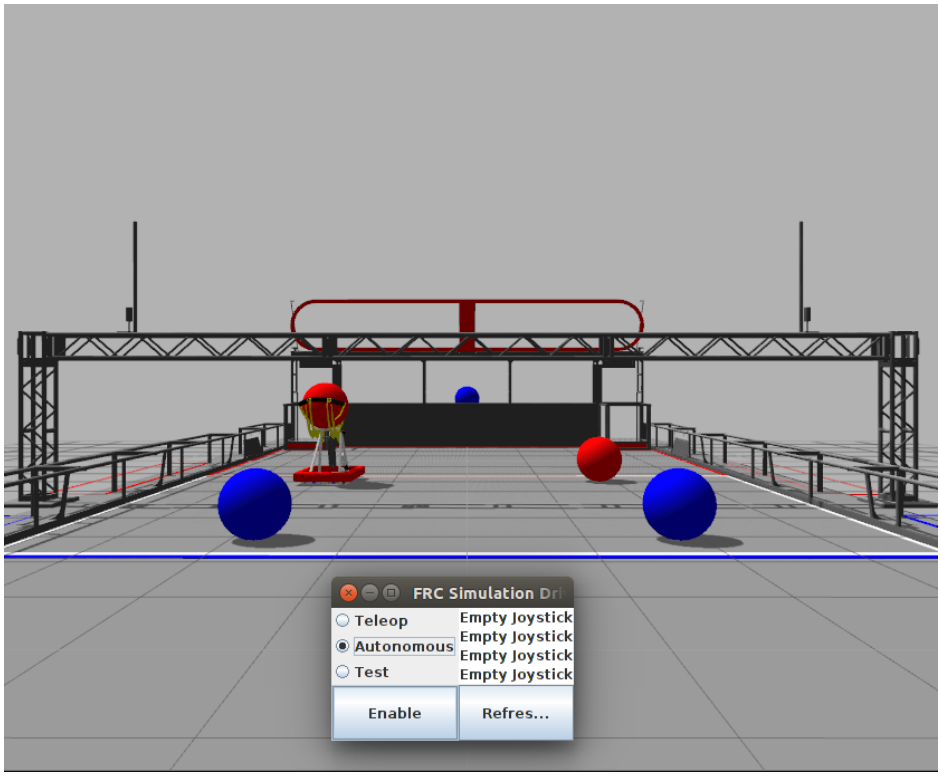
2. Click on the sidebar and type "**sim_ds**" to launch the driver station

2. Insert the PacGoat model from the "insert" tab in gazebo.

If you do not see the programs in the search bar and you installed manually, that means you didn't correctly follow all the instructions in the installation tutorial. Read carefully!

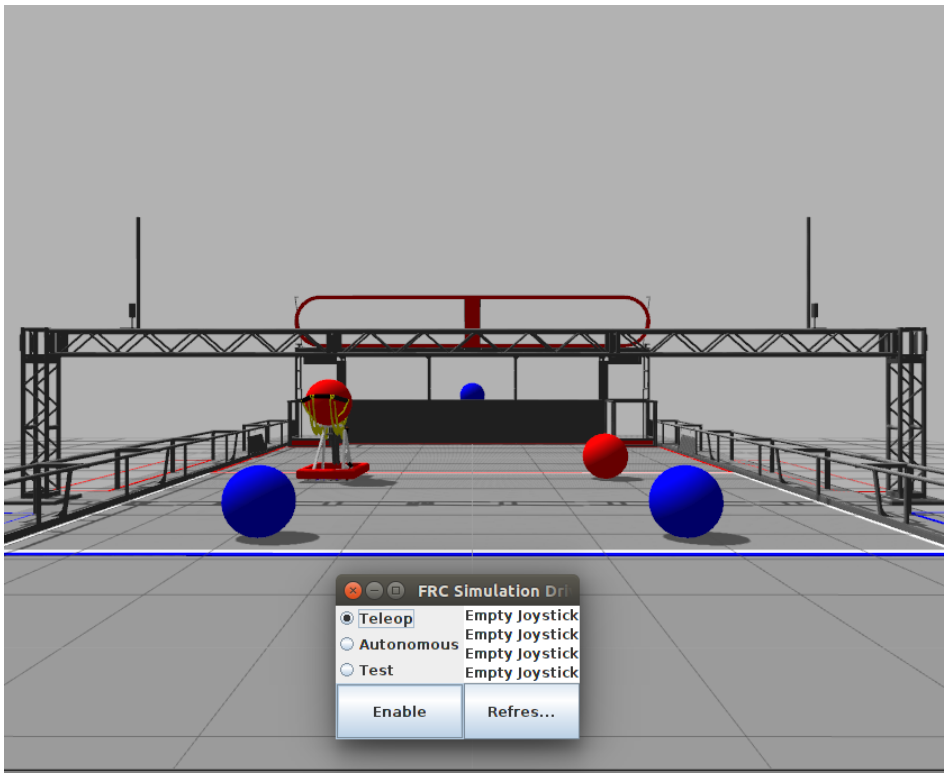
[Installing FRCSim Manually]

Running autonomous



1. Make sure the simulator is running
2. Select Autonomous
3. Click Enable

Using a PS3 joystick to drive



1. Plugin a joystick (preferably a PS3 SIX AXIS/DUALSHOCK)
2. Select Teleop
3. Click Enable
4. Drive around and shoot balls (button mapping available in appendix)

Next Steps

Once you have the GearsBot and PacGoat examples running, there are a number of possible next steps to get familiar with programming FRC robots in simulation and to improve your robotics programming ability.

Extend autonomous

- GearsBot: Extend the example autonomous to pickup and deliver the three sodas behind the robot in addition to the one in front of it.
- PacGoat: Extend the example autonomous to pickup and score a second ball.

Alternate autonomous

- GearsBot: Write an alternative autonomous mode where the robot drives around the box to place the soda.
- PacGoat: Write an alternative autonomous mode that scores in the low goal.

Autonomous choices

GearsBot: Allow the driver to select the autonomous using a SendableChooser. (Hint)

Write a program from scratch

Try creating a new CommandBased project from scratch to make either of the robots do interesting things, such as:

- Drive in a triangle
- Arrange the sodas in a square
- Whatever else you can think of

Debugging Simulation

This article will cover some common problems, debugging techniques, and solutions. Check out the video series for more information!

FRCSim video series

Debugging your robot code

Please, use a real time debugger like GDB or the Eclipse Java debugger!

I cannot emphasize how awesome this is. Since your robot code is running on your computer and not a RoboRIO, you have full access to it at run time and can use the Eclipse Debugger, or GDB, or whatever else you'd like. Furthermore, we left all the debug symbols in WPILib so you can go as deep as you want into our code and find out what's going on!

Print statements are ok for some things, but they are slow to write and test, and require re-building your program just to test things that a debugger can give you for free.

There's a video on how to use the eclipse debugger to solve a real problem in the video series. Check it out!

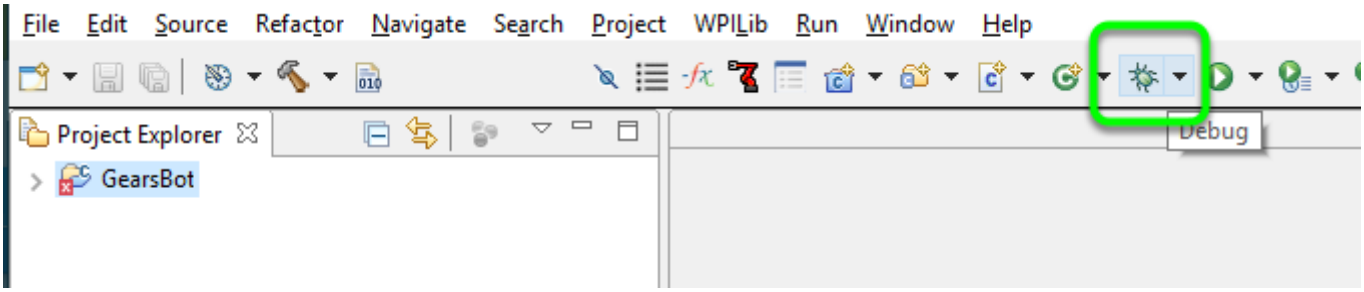
The Eclipse Java Debugger

The eclipse java debugger is one of the most powerful java debuggers available. Learning to use it effectively will mean you spend less time debugging, and more time writing awesome robot code!

There are four main features of the debugger. These include **stepping, breakpoints, watchpoints, and tracing.**

There is a very detailed video tutorial on using this debugger available on youtube here

To run in debug mode, press the green bug icon, or right click and choose "Debug As" > "WPILib Java Simulation"

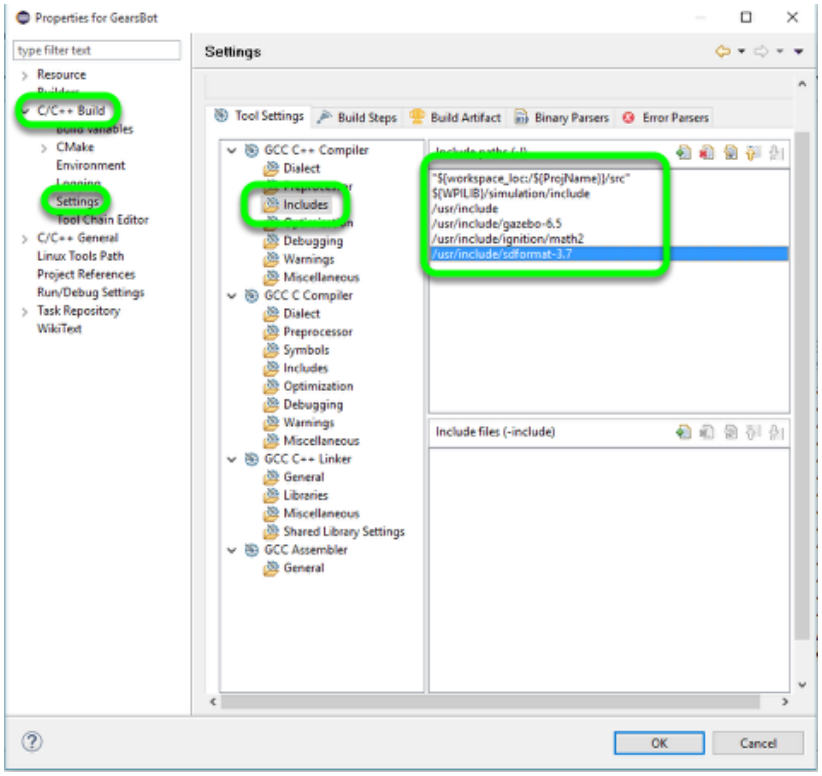


C++ Code Fails to Compile

If you see an error message about including sdf or gazebo like below, there is an easy fix!

Simply right click on your project, and go the C++ settings.

Right click on your project, and go to C++ Include Settings



Change the include path

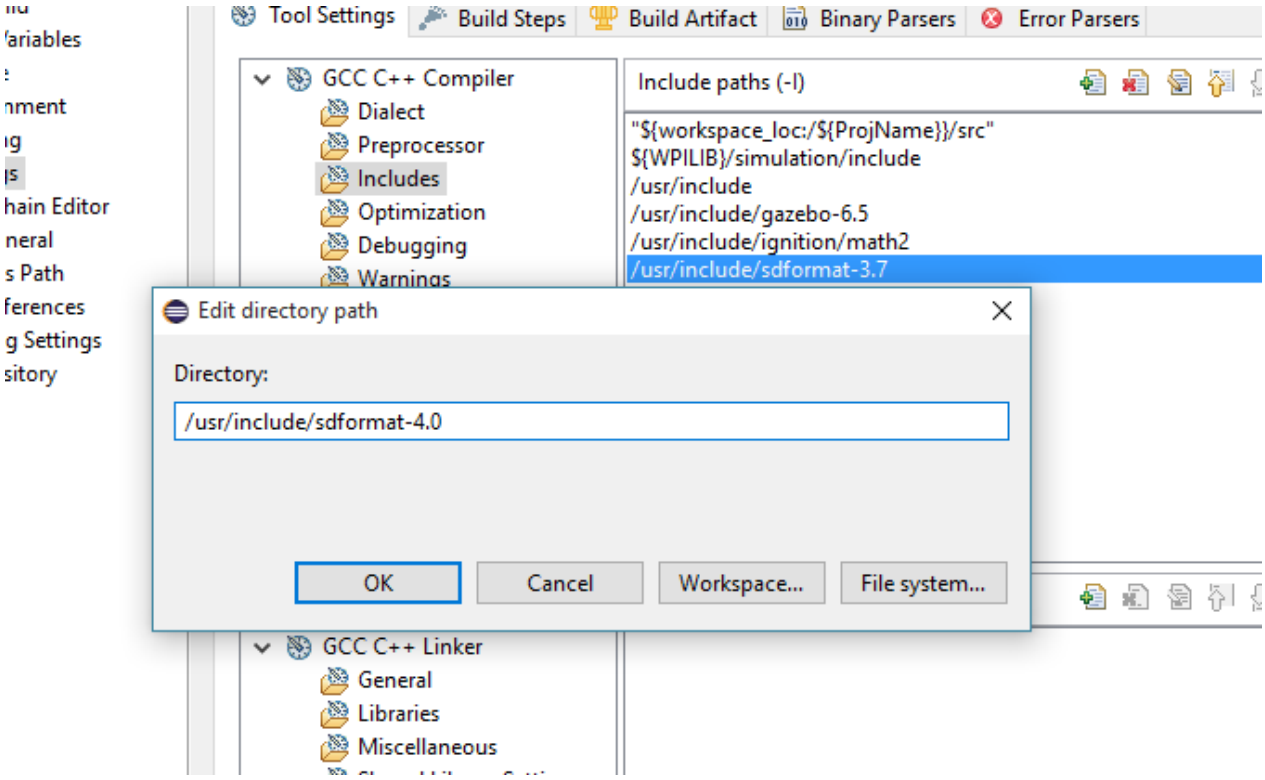
Using the following to command to find what versions of gazebo and sdf you have

```
find /usr/include/ -name gazebo*
find /usr/include/ -name sdf*
```

The output should be something like:

```
/usr/include/gazebo-6.5/
/usr/include/sdformat-4.0/
```

From here you can copy those paths to replace the existing ones in eclipse.



Help! My robot isn't moving!

The most common symptom you'll see is, unfortunately, nothing happening. There are a number of reasons this might be happening, and we'll go through a few of them here.

1. Your robot code isn't built/running properly.

If you're using C++, make sure you select "linux_simulate" from the hammer drop-down when you're compiling. When you run, make sure you hit "simulation".

2. Gazebo is paused

If you hit the little "pause" button in gazebo, everything will freeze. Which means your robot code may also freeze in time as well.

3. Motor messages are not being published

This one is a little trickier to solve. For instance, if you expect a motor on port 1 to be moving, then messages should be published to a gazebo transport topic like

```
/gazebo/frc/simulator/pwm/1
```

You can check if something is being published to this by running the following command in the terminal

```
gz topic --hz /gazebo/frc/simulator/pwm/1
```

This will print out the frequency with which messages are being publish. For instance, any time gazebo is running we can print out the frequency of a default gazebo message

```
gz topic --hz /gazebo/default/world_stats
```

If this is happening, it means your **set** function on your motors is not being called, or is being called with a value of 0. Use a debugger and figure out why!

4. The clock plugin is missing or not working

Every robot model in simulation needs clock plugin. Check the GearsBot model for an example:

```
<plugin name="clock" filename="libgz_clock.so">
    <topic>/gazebo/frc/time</topic>
</plugin>
```

This plugin is responsible for publishing time messages, and nothing will work without it functioning. You can verify it is working by running the following

```
gz topic --hz /gazebo/frc/time
```

If this doesn't print anything, you probably forgot to put the clock plug-in in your robot model. To diagnose, instead of running frcsim from the search bar, run it from the command line. This will print out lots of useful information as gazebo loads and as you try adding models.

How can I make robot models?

Field and game piece models will be available January 12th

Making robot models by hand is very tricky, so I suggest using the Solidworks plugin. However, if you absolutely can't use the solidworks plugin, it is still possible to do by hand. You will need to install **meshlab** before continuing, and you should have already made any CAD models for the parts of your robot, and exported them as STLs.

1. Install meshlab

```
sudo apt-get install meshlab
```

Using the model editor

Gazebo's built-in Model is a nice way to make simple robots to test your code, or to edit custom robots that have small errors.

Follow the guide below to use the model editor:

http://gazebosim.org/tutorials?tut=model_editor&ver=6%2B&cat=model_editor_top

Using the Gazebo Documentation

The Gazebo community has a lot of experience making models, so you can always look at tutorials (http://gazebosim.org/tutorials?cat=build_robot) If you're looking for some information, or ask questions on the gazebo forum (answers.gazebosim.org) if you have a question.

Gazebo is an open source project, and thrives on people asking questions and contributing back to gazebo! You can even add your own custom robots to the online gazebo model repository!

Working by Example

1. Copy an example

The easiest way to make a new robot model is by basing it off a similar one. For FRC Robots, that usually means GearsBot is a good example. Copy the files in ~/wpilib/simulation/models/GearsBot to another folder, like ~/wpilib/simulation/models/MyRobot. This will give us a good starting point.

2. Understand the links and joints

Look through the copied GearsBot file. You'll see there are several links and joints. At this time we want to start replacing those links and joints with our own. It's common to have four wheels and a chassis in your robot, so you can start by simply replacing the models in those *links* with the models for your robot.

You might see this:

```
<geometry>
        <mesh filename="package://GearsBot/meshes/chassis.STL" />
</geometry>
```

And you can replace it with this:

```
<geometry>
        <mesh filename="package://MyRobots/meshes/chassis.STL" />
</geometry>
```

3. Update poses

Another difference between your robot and GearsBot are the pose and inertia of the links. For pose, start by setting all the numbers to zero, and adjust them until the links line up correctly. You can "insert" the model into gazebo, see in which direction they are off, make the adjustment, and re-insert.

4. Update intertia

The inertia for different links needs to be change for a new model. This is where meshlab comes in handy. Follow the instructions here (http://gazebosim.org/tutorials?tut=inertia&cat=) to use meshlab to find out the values for the inertia tag.

The easiest way to calculate inertia for most simple links is to use the following table of inertial tensors: https://en.wikipedia.org/wiki/List_of_moments_of_inertia

Video Tutorials

Don't be afraid to comment on the videos if you have questions or comments!

FRCSim video series

GearsBot information

GearsBot is a simple demo robot that is used to demonstrate writing WPILib programming. There are two sets of talks using it available on youtube: one with RobotBuilder and the other with just NetBeans. The example program provided is very similar to the code written in those talks, but it does have a few extra features to make it more simulation friendly.

There are two major differences for supporting simulation. The first is

taking advantage Encoder.setDistancePerPulse() and the AnalogPotentiometer to convert the readings into meaningful units such as feet and degrees. The simulator returns values in these more meaningful units by default and by

21converting the readings from the real robot to these units the code can be written at a higher level that runs in both simulation and on the real robot. The second major difference is that the PID values are different on the real robot and on the simulated robot. While it would be ideal if they could be the same, the reality is that the model isn't accurate enough and for

better performance they use different PID values.

All of the other code is the same, so it's one codebase that can be run in

two different ways. In order for the code to know whether or not it is run- ning on the real robot or in simulation, there are two convenience methods: Robot.isReal() and Robot.isSimulation() which return booleans. Any code that is specific to either the real or the simulated robot is wrapped in an if statement with a call to one of these methods so that it only runs in the right configuration.

Operator interface

PS3 Controller		
Button #	*PS3 Button*	*Action*
1	Select	None
2	Left-Stick	None
3	Right-Stick	None
4	Start	None
5	D-Up	Raise Elevator
6	D-Right	Close Claw
7	D-Down	Lower Elevator
8	D-Left	Open Claw
9	L-2	Autonomous
10	R-2	Pickup
11	L-1	Place
12	R-1	Prepare to Pickup

The example program has an operator interface for a PS3 controller that allows basic teleoperated control. The robot is driven using the left and right joysticks y-axis to control the corresponding left and right motor speeds. The left and right bumpers provide access to semi-autonomous control and the D-pad provides more granular control over the claw and elevator.

Robot Map

PWMs		
Pin	*Subsystem*	*Actuator*
1	Drive Train	Front Left Motor
2		Front Right Motor
3		Back Left Motor
4		Back Right Motor
5	Elevator	Motor
6	Wrist	Motor
7	Claw	Motor

Analog		
Pin	*Subsystem*	*Sensor*
1	Drive Train	Gyro
2	Elevator	Pot
3	Wrist	Pot

Digital		
Pin	*Subsystem*	*Sensor*
1	Drive Train	Left Encoder $^{A}_{B}$
2		
3		Right Encoder $^{A}_{B}$
4		

PacGoat information

PacGoat is FRC Team 190's robot that competed in the 2014 season. The code distributed for the example is a modified version of what ran on the actual robot, so as to be realistic as possible in simulation. There are some differences:

- The operator interface and commands have been simplified to work on a PS3 controller as opposed to needing multiple joysticks.
- It has been updated to use 2015 WPILib API, including Simulation support, non-module based port numbers, etc
- Refactoring, comments and general cleanup.
- Some features, such as compressors and some sensors aren't supported properly in simulation yet.
- The firing mechanism in simulation is currently simplified to a single pneumatic cylinder.

Operator interface

PS3 Controller		
Button #	*PS3 Button*	*Action*
2 3	Left-Stick Right-Stick	Shoot
9	L-2	Aim Near
10	R-2	Collect
11	L-1	Aim Far
12	R-1	Low Goal

The example program has an operator interface for a PS3 controller that allows basic teleoperated control. The robot is driven tank style, using the left and right joysticks y-axis to control the corresponding left and right motor speeds. The left and right bumpers control the pivot and rollers. Pressing both sticks simultaneously fires the ball.

Robot Map

PWMs		
Pin	*Subsystem*	*Actuator*
1	Drive Train	Front Left Motor
2		Front Right Motor
3		Back Left Motor
4		Back Right Motor
5	Pivot	Motor
6	Collector	Roller Motor

Pneumatics		
Pin	*Subsystem*	*Actuator*
1	Collector	Claw Piston
2	Shooter	Latch
3		Piston 1
4		
5		Piston 2
6		

Analog		
Pin	*Subsystem*	*Sensor*
1	Pivot	Potentiometer
2	Drive Train	Gyro
3	Pneumatics	Pressure Sensor

Digital		
Pin	*Subsystem*	*Sensor*
1	Drive Train	Right Encoder $^{A}_{B}$
2		
3		Left Encoder $^{A}_{B}$
4		
6	Collector	Claw Open Limit Switch
10		Ball Grabbed Limit Switch
9	Shooter	Piston 1 Front Reed Switch
11		Piston 1 Back Reed Switch
12	Pivot	Lower Limit Switch
13		Upper Limit Switch

Importing a model into Gazebo

Shows how an exported SolidWorks model can be opened in Gazebo

Verify model install location

Ensure you have exported your SolidWorks model and have saved it onto your computer.

If you place the folder in *~/wpilib/simulation/models* then you can skip the next step.

Custom Install Location

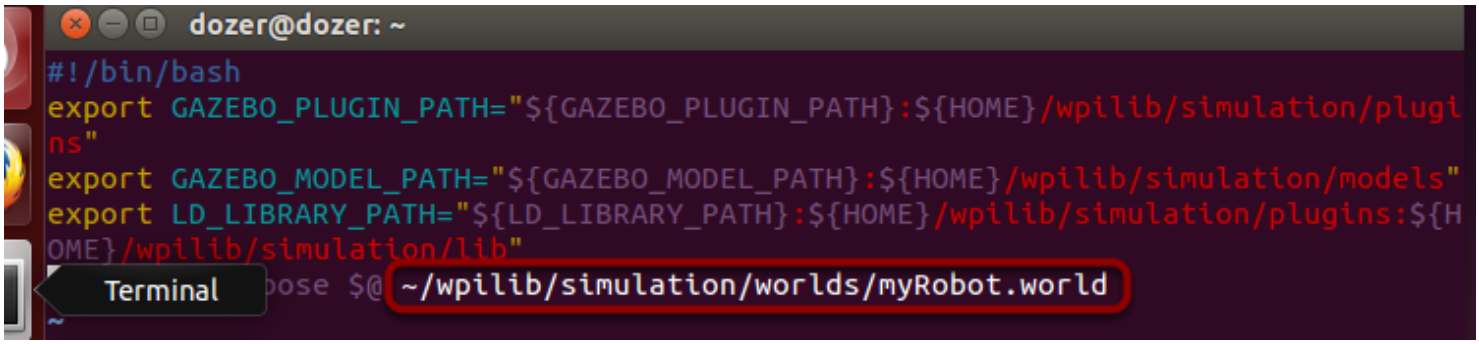
Locate wherever you exported your model. Within that folder you will find a folder with your model's information, and a *.world* file. You can tell Gazebo to look in that folder for models by adding it to the *GAZEBO_MODELS_PATH* environment variable.

Open up the file called **frcsim** located at **wpilib/simulation/** and find the line that says:

```
export GAZEBO_MODELS_PATH=~/wpilib/simulation/models/:$GAZEBO_MODELS_PATH
```

Concatenate your path to that variable. Here's what it might look like after:

```
export GAZEBO_MODELS_PATH=/usr/share/models:~/wpilib/simulation/models/:$GAZEBO_MODELS_PATH
```



Opening the model in Gazebo

Models in gazebo can be opened in one of two ways

1. Through a world file
2. From the insert tab

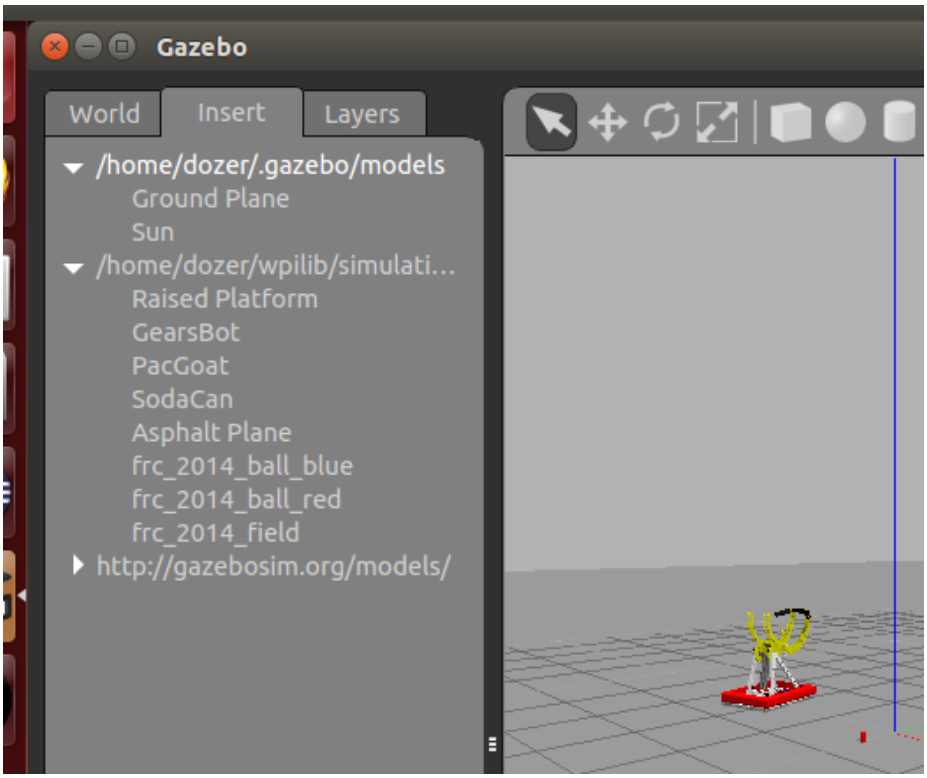
World Files

World files can describe multiple models' locations, as well as lighting and camera information. Open up a terminal and run:

```
frcsim path/to/file.world
```

This will launch gazebo with that world file. If you want to use the same world file all the time, you can simply add that to the **frcsim** file in ~/wpilib/simulation

From the Insert Tab



FRC 2016 Field!

The 2016 field models are now available in the insert tab of gazebo! We've written a world file for you as well, so you can import all the pieces in the right spot.

Checkout the download section here!

first.wpi.edu/FRC/roborio/release/simulation

Running an Robot Program in Simulation

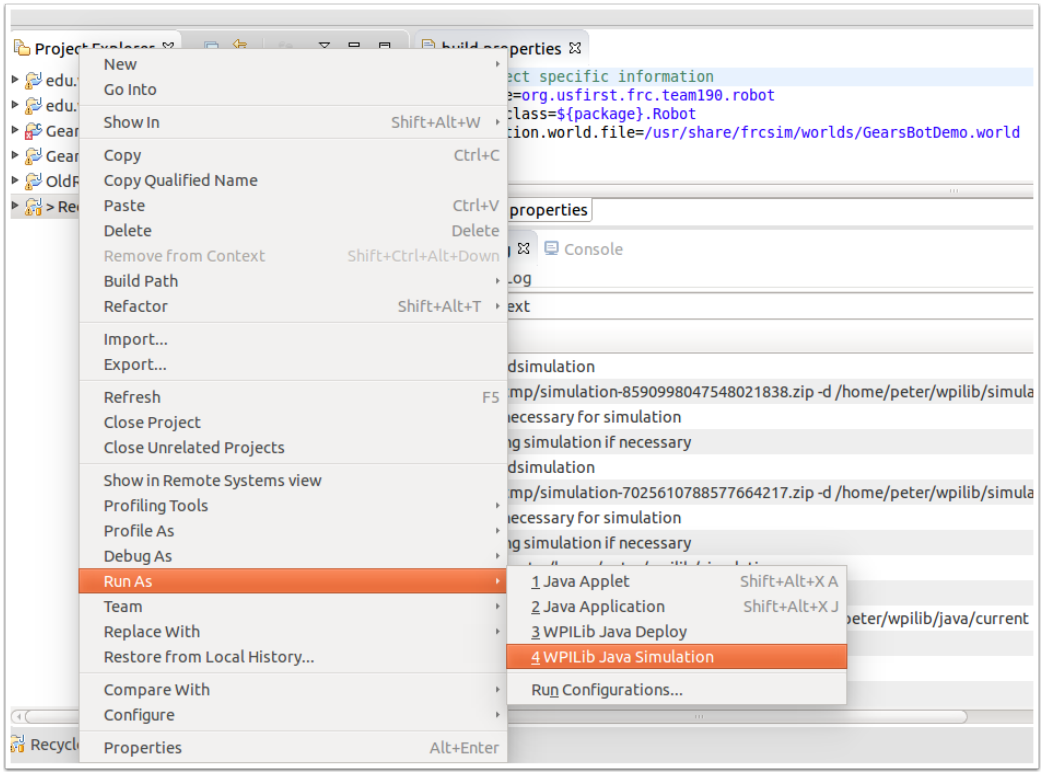
Learn how to run your robot code in simulation.

Locate your project

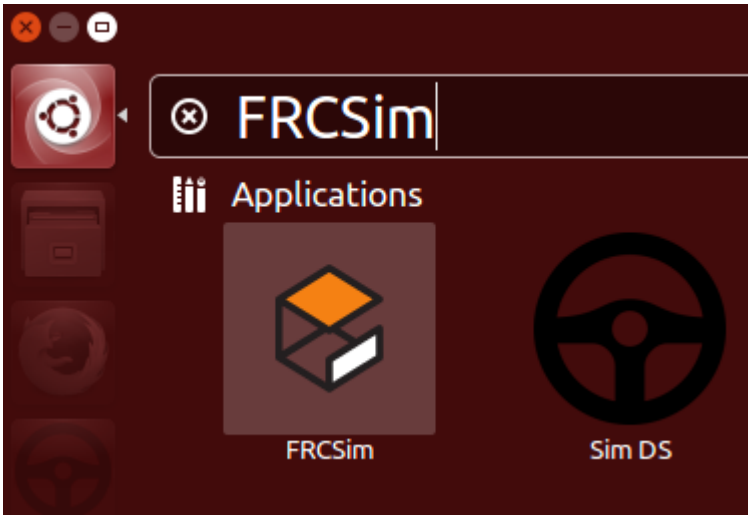
Open eclipse and find the project you want to run in simulation. If you're trying to run this on an old project, follow this tutorial on how to update your old project to work with FRCSim.

Assuming you've done that...

Just hit run!



Launch Gazebo



Or, run "frcsim" from the command line.

If nothing happens or nothing appears, go back and finish the manual install process.

Connecting with the Driverstation

Using FRCSim with C++ and Java

Start gazebo first. If you haven't started gazebo, nothing will happen. Open the launcher and type "Sim DS".

WPILib comes with a linux driverstation to use in simulation. It lacks most of the features of the other driverstation, but it works to change the robot state and read from joysticks. The Logitech F130 and XBox gamepads are known to work. Others may, but you need to have the right drivers.

You can also run the following in a terminal:

```
sim_ds
```

If nothing happens or nothing appears, go back and finish the last step of the manual install

Note: if you see the warning about gazebo transport, that's ok! It just means you haven't started Gazebo yet. If you HAVE started gazebo and still see this, then you probably need to set GAZEBO_MASTER_URI to the right IP address. See here for more details.

Using and old robot project with 2016 FRCSim

Learn how to take an old C++/Java project and use it with FRCSim

Netbeans or Windriver Projects

Overview

Create a new Eclipse WPILib Java or C++ Project, delete all the source files, and copy the sources files from you old project

Step-by-Step

- Open up a new WPILIb C++ or Java project
- Name your project if you like, but the defaults are fine
- Once your project is created, open up a file browser or a shell and copy the entire src directory from you previous project into your new project

Eclipse Projects

C++ projects

1. Create a new C++ Robot Project
2. Copy all the source files located from the old project into the new project.

Note: The .cproject file is a complex XML file that describes your 'build configurations'. If you read it, you'll see new configurations added for simulation compared to a your other old robot projects!

Note: If you want to try using the terminal to do this, try using the "mv" (move) command!

Java Projects

Nothing! Simply update your plugins so a version that supports simulation and you're all set!

Exporting a SolidWorks robot model for simulation

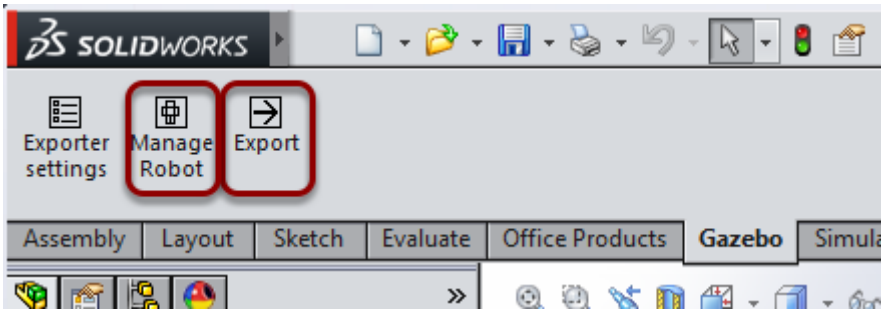
Exporting overview

This article provides a brief overview of the process of exporting your robot from Solidworks. For more detailed instructions, refer to the next article: Exporting A Solidworks Model To Gazebo.

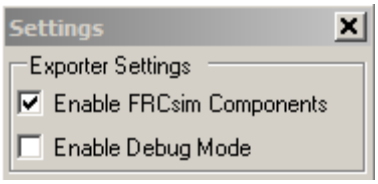
An overview of the process of exporting a Solidworks model to Gazebo

Once you have made your robot model in Solidworks, there is some more information you must provide through the following steps:

1. Set the robot name and frame-of-reference in the "Manage Robot" window. Steps 2-6 will also be done from the "Manage Robot" window.
2. Add/remove links and connect/disconnect links with joints.
3. Create collision models as a new Solidworks Configuration.
4. Set each link by selecting SolidWorks components and setting various property values. For more info about selecting components, see the article Assigning components to links.
5. Set each joint by setting its type and various property values. For a list of currently supported joints and their properties, see the article Setting joint
6. Add attachments, which may be actuators (e.g. piston) or sensors (e.g. gyro). For info about currently supported attachments, see the article Adding Motors and Sensors to a model.
7. Verify that the necessary info has been provided, and export your robot from the "Export" window.
8. Transfer your robot files in your computer to access in Gazebo. For more info, see the article Importing a Model into Gazebo



Exporter Settings



Exporter settings are settings that are used for all assemblies/robots:

- Enable FRCSim Components - Check to enable addition of FRCsim attachments and sensors like motors, gyros, etc and for the option to export your robot in a world with an FRC Field.
- Enable Debug Mode - Check to disable certain safety checks and enable all side tags.

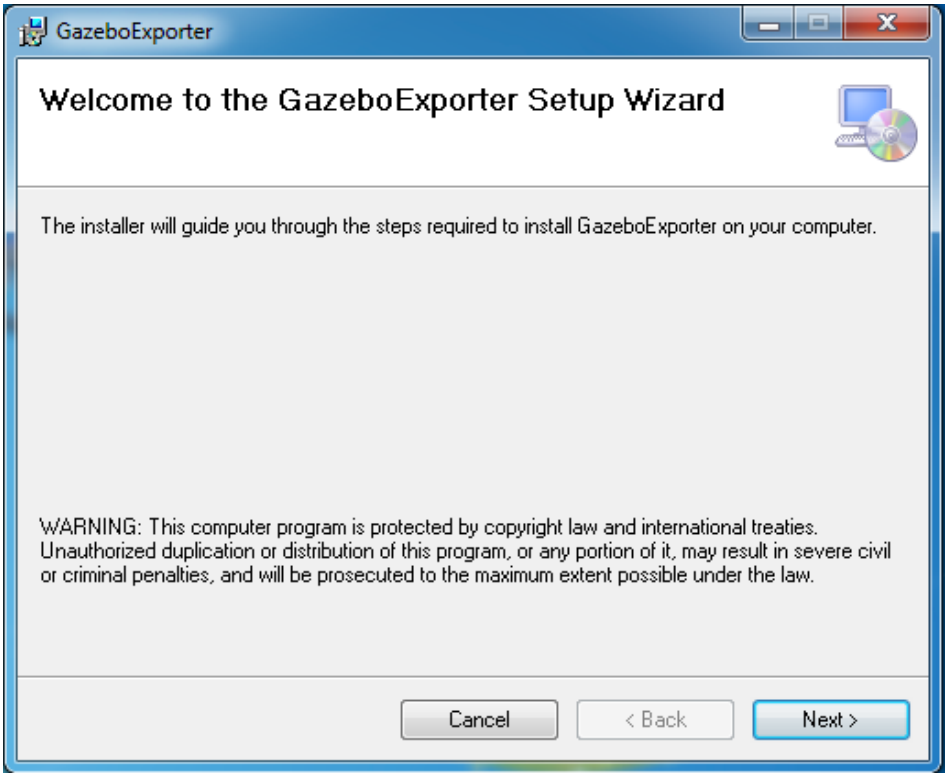
Installing SolidWorks Gazebo Exporter plugin

The SolidWorks Plugin allows you to export CAD models into Gazebo simulator. The plugin has only been tested in SolidWorks 2014-2015.

Download the installer

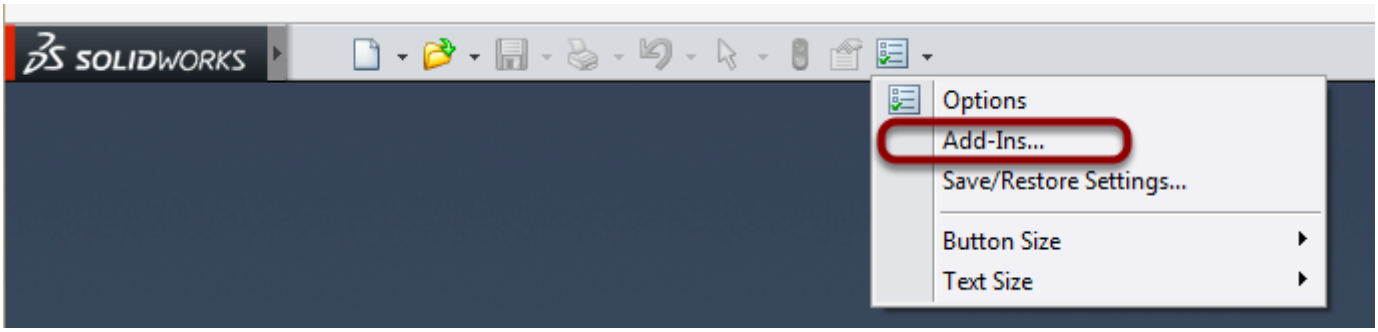
The SolidWorks plugin can be downloaded here.

Run the installer



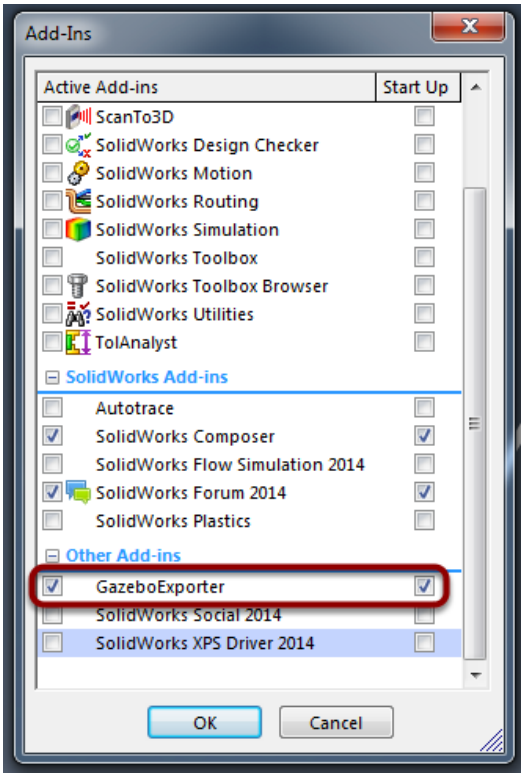
Run the installer, taking all the defaults. If .Net is not currently installed on your machine, you will be asked to install it. Follow the directions it provides.

Verify the install



Click the options button in the tool bar then select the Add-Ins option.

Enable the Add-In

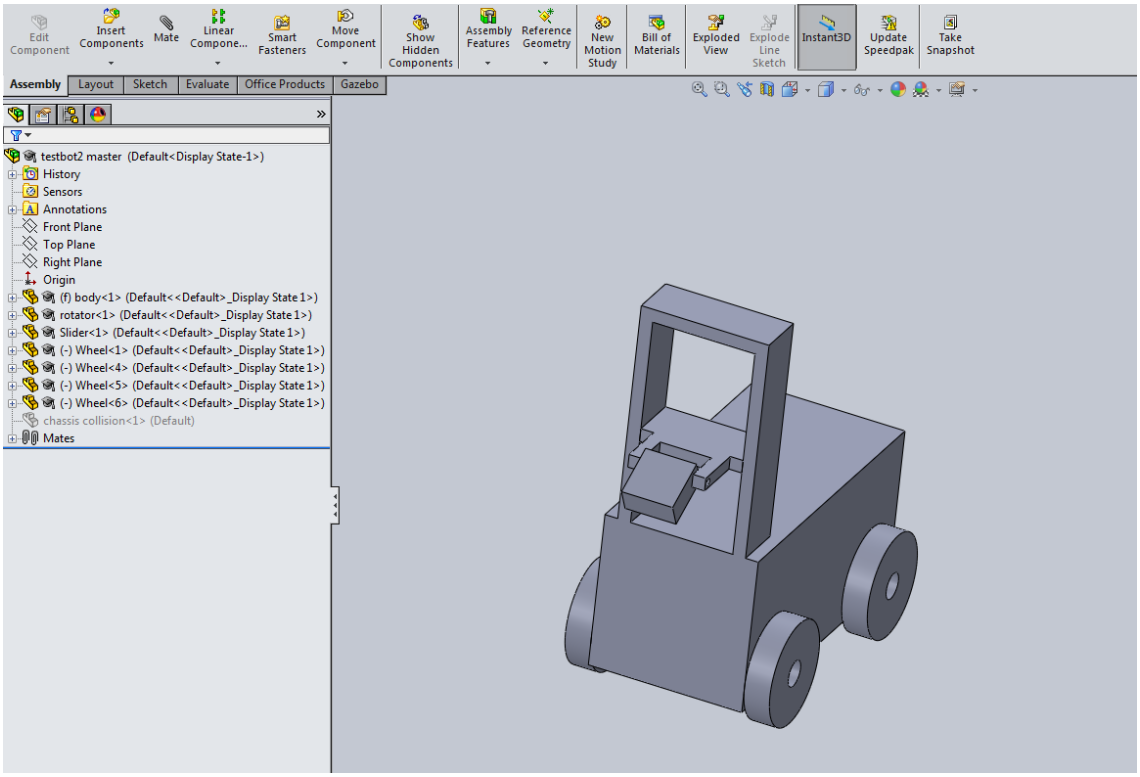


Scroll down to the GazeboExporter under other Add-Ins and ensure both checkboxes are checked.

Exporting a SolidWorks model

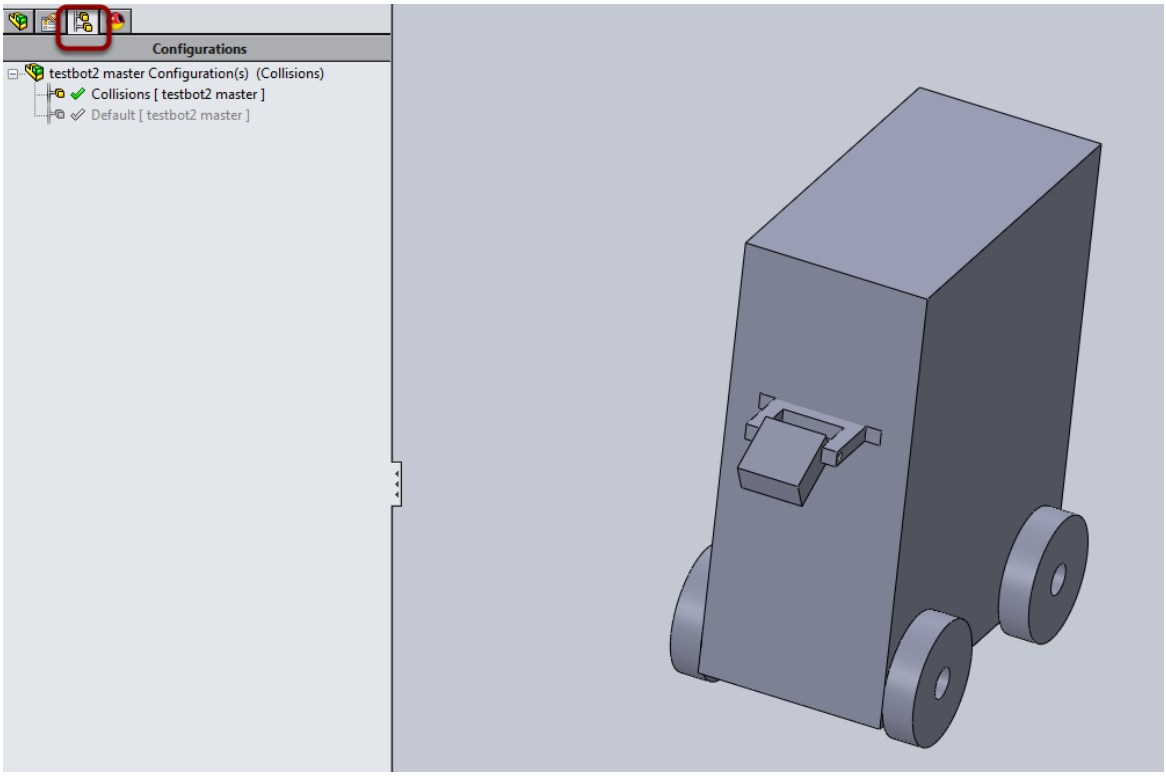
A tutorial on how to use the SolidWorks Gazebo plugin to export a robot model.

Open the Desired Model



Open the robot model that you desire to import.

Create Collision Models

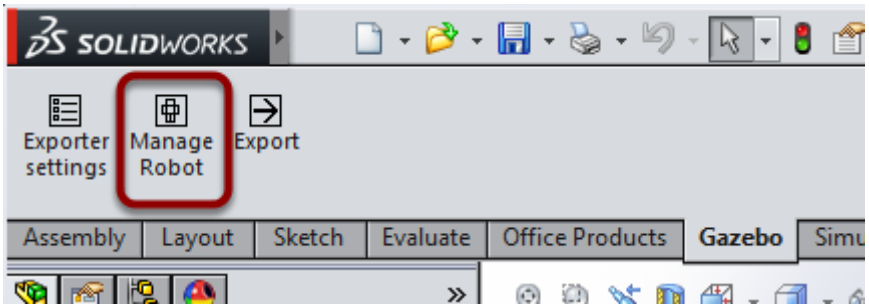


For each link, click the "Select Visual Components" buttons to switch to the Solidworks window and select components in the assembly that should be part of the link. Re-click a selected part to de-select it. Select the green checkbox to return to the "Manage Robot" window.

Repeat for with the "Select Physical Components" and "Select Collision Components" buttons to specify different models for calculating the center of mass/inertia values and for use by simulator. Switch Solidworks configurations in between selecting components, if you created a separate collision robot model. This is not necessary, but recommended for complex models.

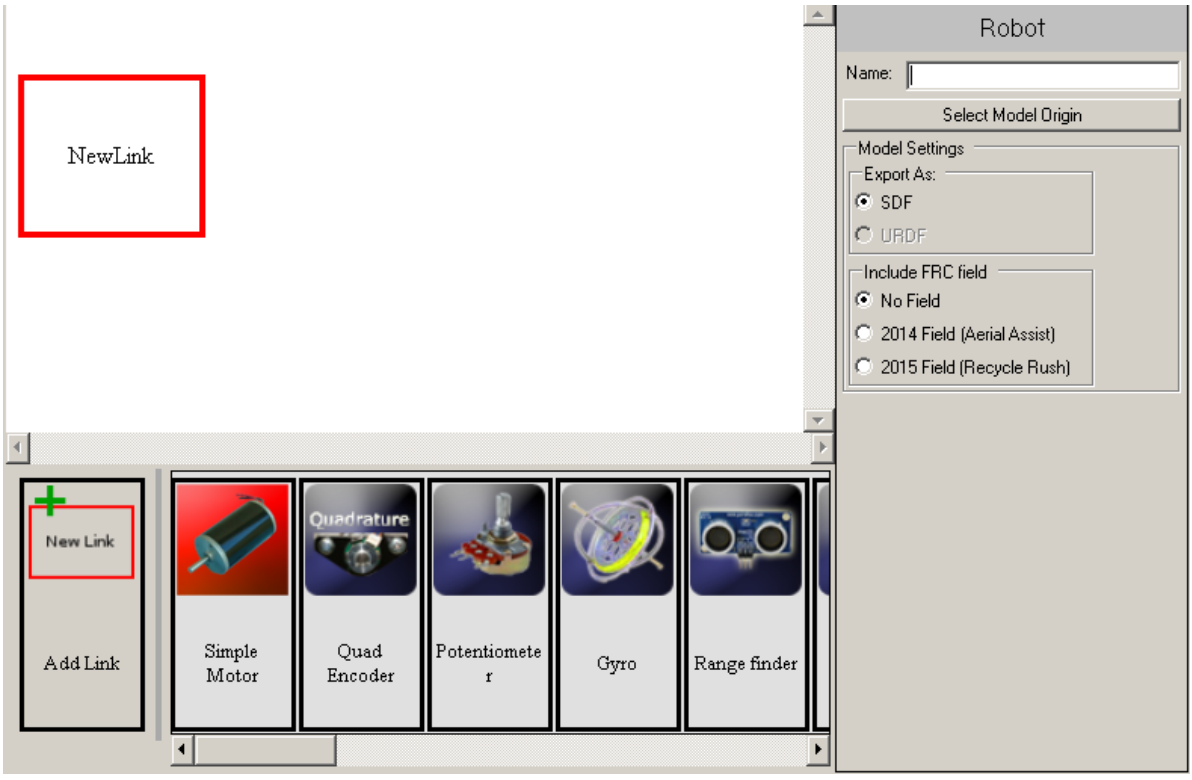
For more details, see the article: Assigning components to links.

Open the Manage Robot Tab



Click on the "Manage Robot" button in the Gazebo tab to open up the Manager Robot Window. This window will let you edit various properties of the robot, its links, its joints, and its attachments.

The Manage Robot window

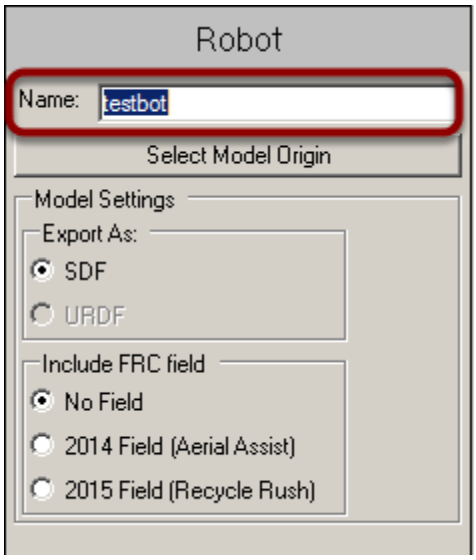


The Manage Robot window has 3 areas:

- in the top-left is a graphical representation of the robot as links connected to one or more other links by joint(s)

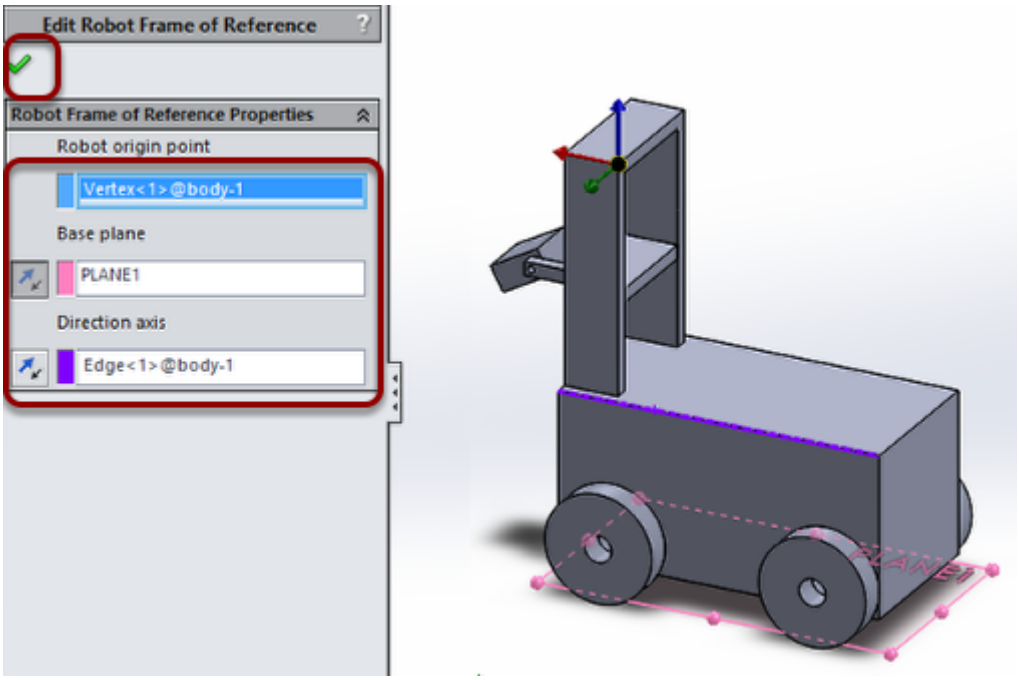
- in the bottom are the link and attachments that can be added to the robot graph
- in the right is a property manager panel where you can edit the properties of links, joints, sensors, and the robot after clicking something in the graph area.

Name the Robot Model



In the name box on the right hand side, give a name to the robot model. This is the name that will be used in Gazebo.

Set robot orientation

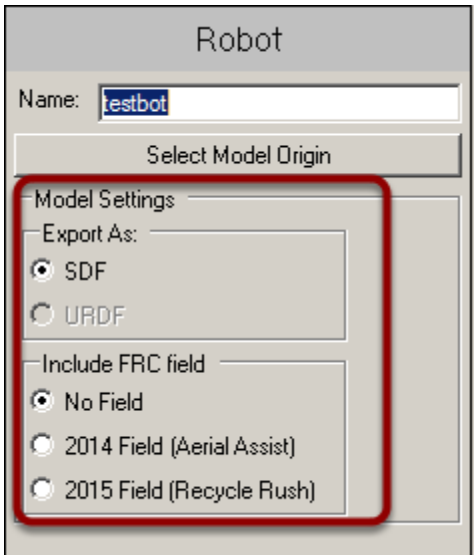


Click the "Select Model Origin" button to select an origin point for the robot, the base plane on which the robot stands, and the axis along which the robot faces. A set of coordinate axes will be drawn at the origin point based on your selections; if the arrow on the chosen axis is in the opposite direction, toggle the direction of the axis or plane as necessary by clicking the direction button to the left of the selection box.

Note that the base plane represents the offset from the origin point to the ground as well as defining the Z-axis of the robot. When the robot is brought into gazebo the robot will start with this base plane being coplanar to the ground plane.

Click the green checkmark to return to the "Manage Robot" window.

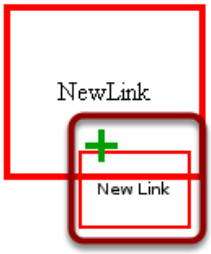
Set robot settings



If you have enabled FRCSim components in the Exporter Settings, you have the option to export your robot model with a FRC competition field in a world file.

Note: The option to export in .URDF format is not yet supported.

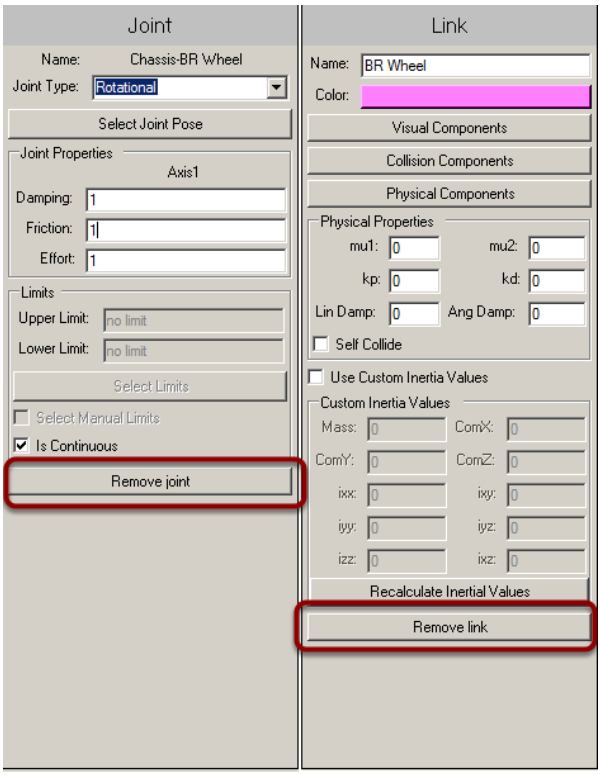
Create links and joints



Custom collision models should be created for complex models in order to not slow down the physics simulation. These models should be made to only represent the important geometry for simulation. Custom collision models are not required, but are strongly suggested for all but the simplest of models. For more information on collision models see article: ARTICLE

If you have not done so already, you should also set the material of your model parts in the Feature Manager Design Tree Tab.

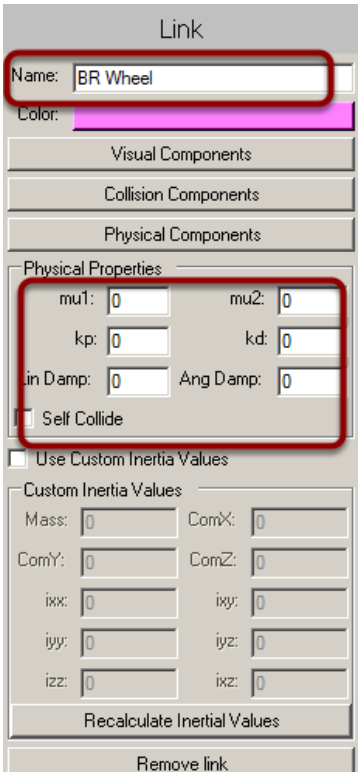
Delete links and joints



To delete a link or joint, select it in the graph area, and press the remove button in the right panel.

Note that this will also delete any attachments, dependent sublinks, and connected joints. Thus, if removing a joint or link completely cuts off a set of links from the base link, the link(s) and their associated joints and attachments will be removed.

Set link properties



For each link, select the link's box in the graph panel, give it a unique name, and set the following property values in the right panel:

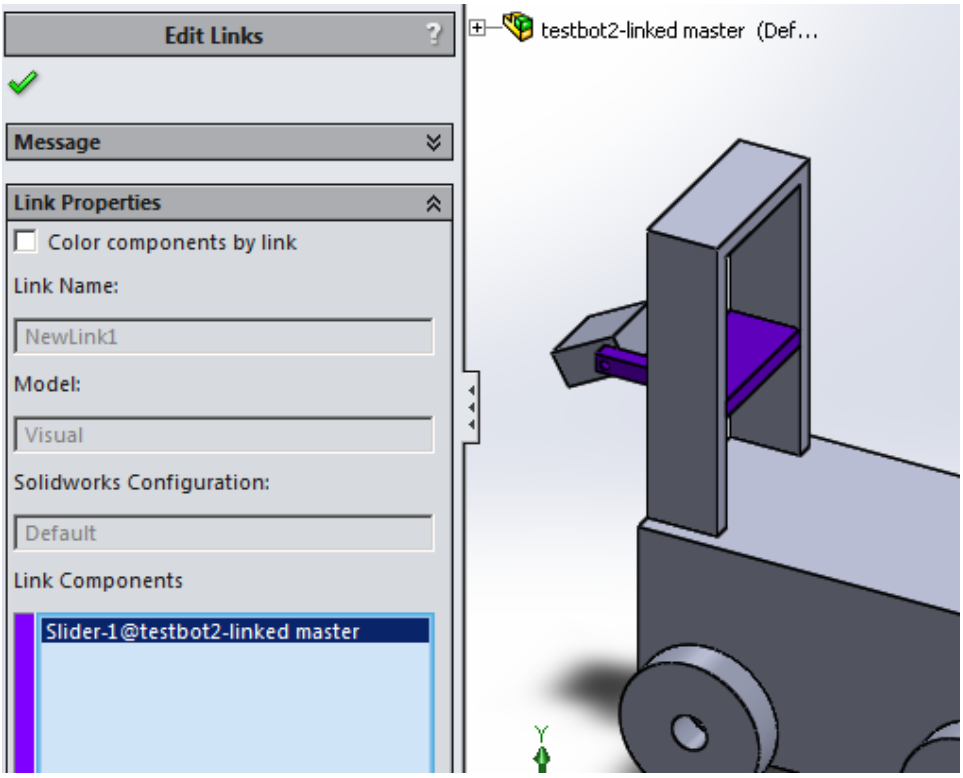
- mu1 and mu2 represent friction coefficients along both directions of the surface of a contact joint (unitless).

These properties can often be left blank, but for some models may be helpful to adjust:

- kp represents the contact stiffness coefficient of the link, and is to a material as the spring constant is to a spring. This is usually a very large value as greater values indicate harder surfaces. Soft, deformable surfaces can be simulated by adjusting this coefficient and the kd coefficient. The smaller the values the softer the material will be. An example of a good value for a hard rubber gripper would be about 10000.
- kd represents the contact damping coefficient of the link. Like the kp value, this coefficient represents the stifness of the link. This value tends to be much smaller than the kp value. A hard rubber gripper would have a value of about 1.
- Linear damping and angular damping is represents how much the link resists force (in Newton-seconds/meter), similar to the damping property of joints.

- Checking "Self Collide" enables this link to collide with other links of the robot. Note, even with self collide enabled a link will never collide with any links it is directly connected to via a joint.

Set link components

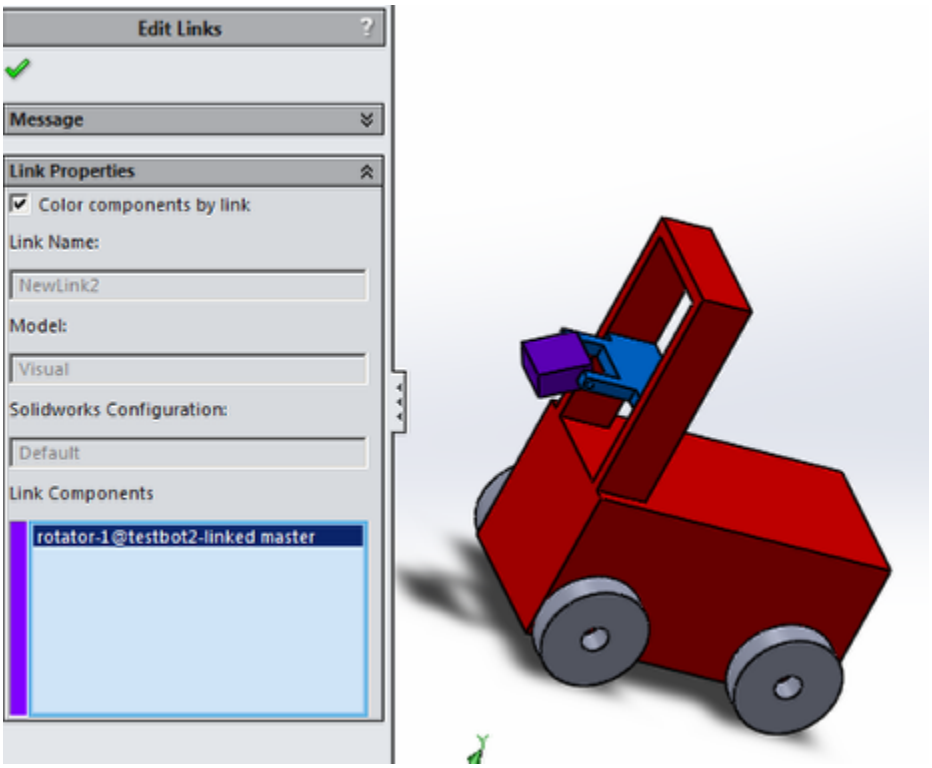


A link is a set of components that do not move relative to one another and can be treated as one part of the robot, and links are connected by joints.

To add a child link to an existing link, drag a new link from bottom selection panel onto any link in the graph area. This will automatically create a joint connecting these links.

To connect any two existing links with a new joint, simply click on the desired parent link, drag the mouse, and release on the desired child link. Note that the link that the drag is started from will be the parent link of the joint, and the other link will be the child of the joint.

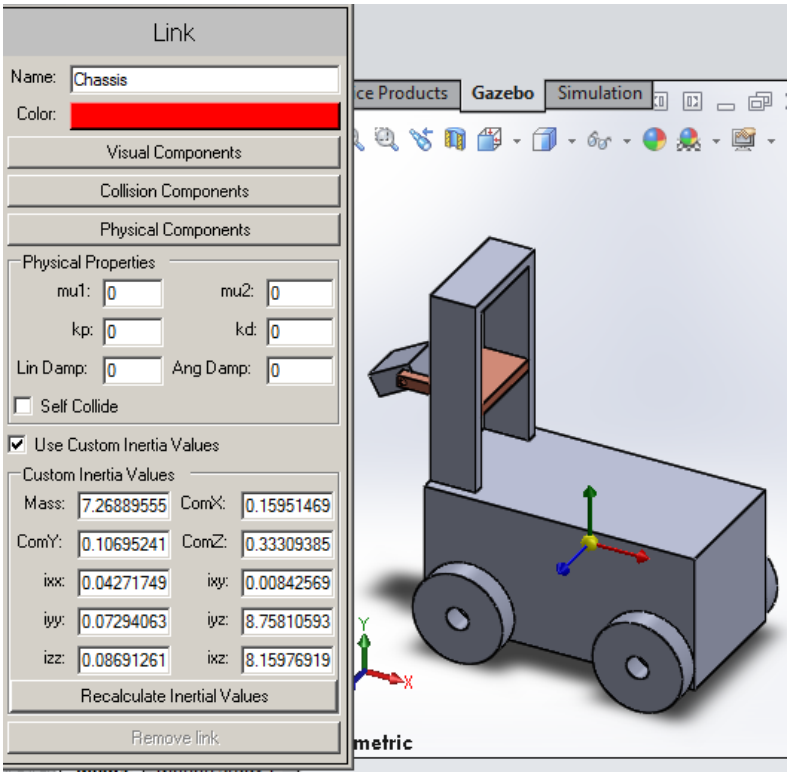
Color components by link



You can see which components have been assigned to other links by checking "Color components by link". The links will be colored based on the colors chosen in the robot manager. Note that if a component is assigned to multiple links, it will only be colored to the color of one link.

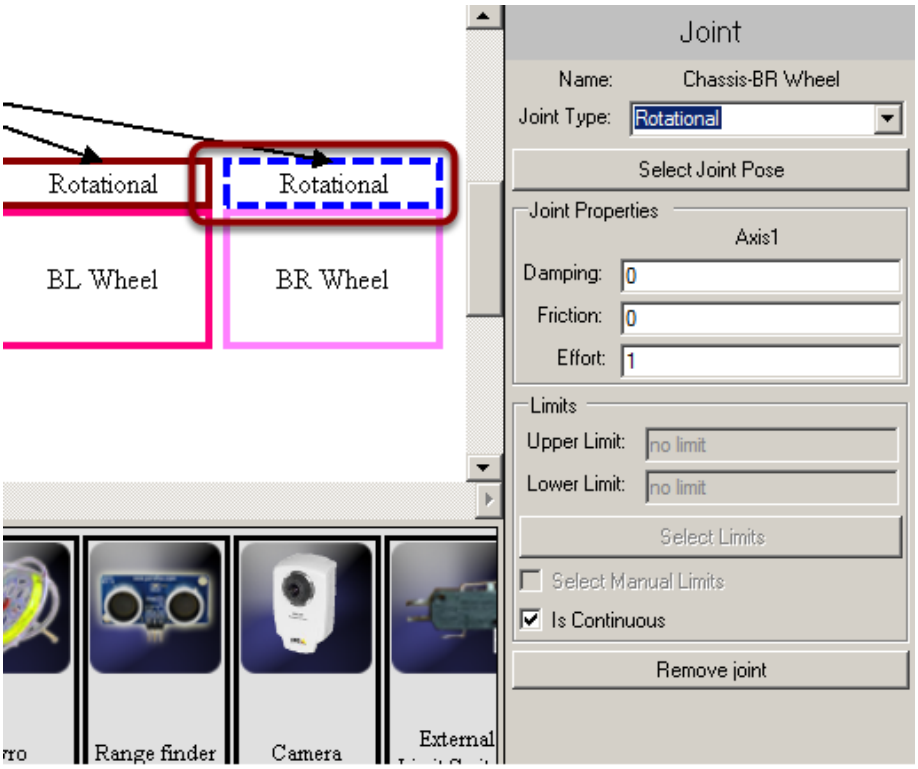
This is helpful to check that all your components have been assigned to at least one link.

Set Manual Inertia Values (optional)



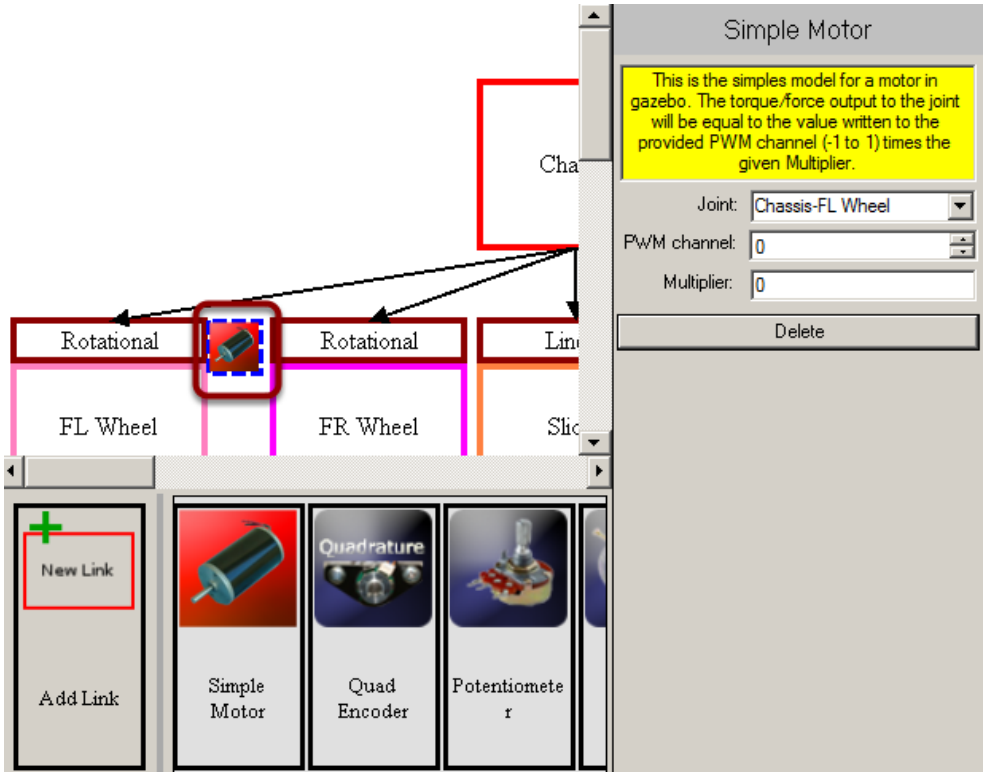
By default, the mass and inertial values are calculated using the physical properties of the model, but they can also be set manually, if needed. To do this, first check the box labeled "Use Custom Inertial Values." The values can then be edited using the text boxes. A triad will also appear at the center of mass to give a preview of its location. This handler can also be dragged around to adjust the location of the center of mass. All values are metric.

Define Joints



The next step is to define the joints. For each joint, select by clicking on it in the graph area, and edit its properties in the right panel. For more detailed instructions and a list of currently supported joints and their properties, see the article: Setting joints

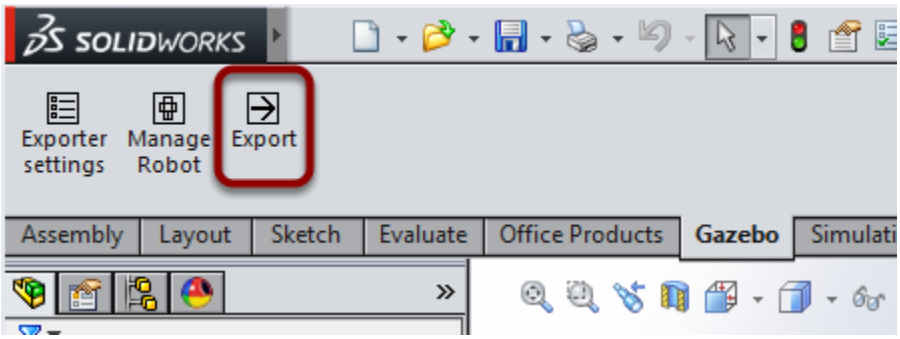
Add attachments and sensors



Once you have defined all your links and joints, you may add attachments to them by dragging the desired attachment from the bottom panel to a link in the graph. Then, click on the attachment to edit the attachment's properties in the right panel. To remove the selected attachment, click the "Delete" button.

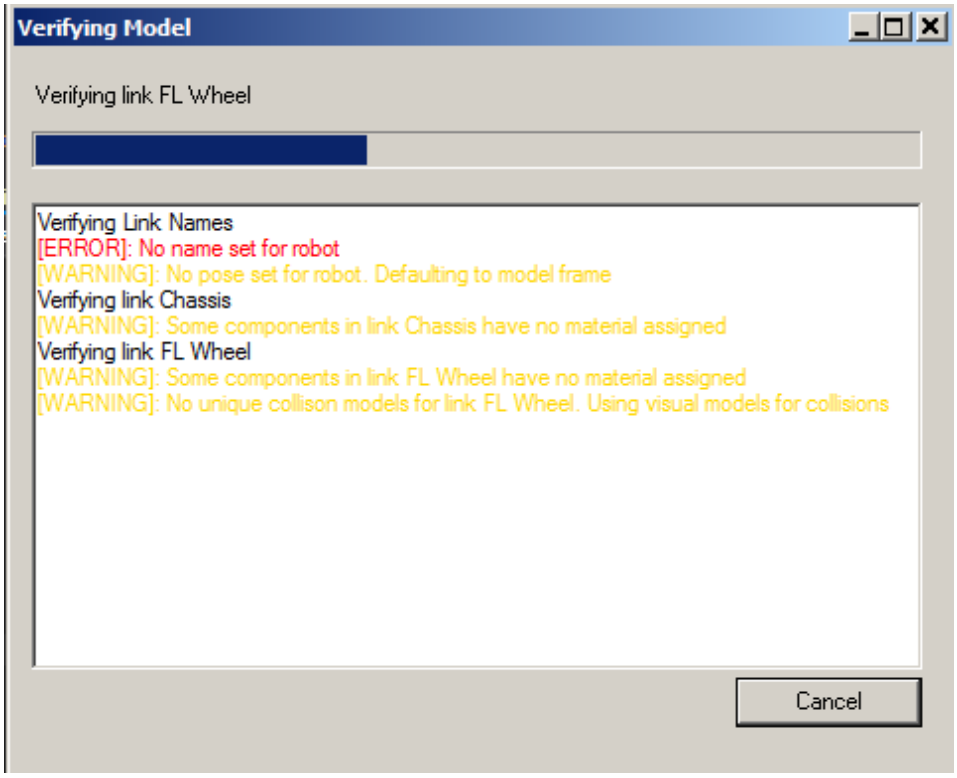
For more detailed instructions and a list of attachments, see the article: Adding Motors and Sensors to a model.

Export the Robot



Once all settings have been adjusted, export the robot by clicking the "Export" button under the Gazebo tab.

Verifying model

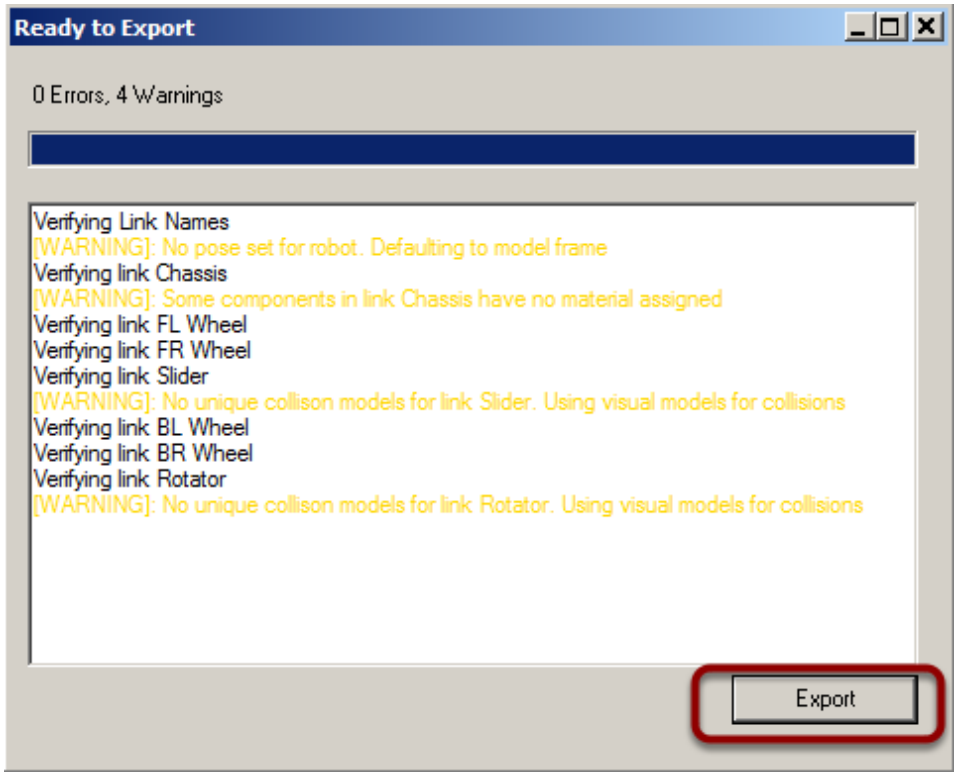


A window will appear while the exporter verifies the model. The exporter will run through a series of steps to verify that the model will successfully export.

Any minor issues will be reported in yellow. These warnings will not prevent the model from being exported but may cause issues when running the simulation.

Any major errors will be reported in red These errors will prevent the model from exporting and the model must be corrected before the model can be exported.

Exporting model



Once the verification is successfully completed without any errors, click the export button in the window.

A window will appear where you can select the location that the exported model should be saved to. The window will default to the HOME\wpilib\simulation directory if one exists as this is the default location for using the models in windows. If you are using Gazebo in linux you can save the model to another location and then transfer the files to your linux computer.

Once this is done hit OK, and then the model will be exported. The export can be canceled any time by clicking on the cancel button. Once the export has finished, you may close the window.

Assigning components to links

This article explains how to assign components to links and the purpose of the different SolidWorks configurations and the different component models.

Introduction

This article explains how to assign components to links and the purpose of the different SolidWorks configurations and the different component models.

Each link must be assigned components in at least one of its three component models (Physical, Visual, and Collision).

Component models

MODEL TYPE	PURPOSE	DETAILS
Visual	Shown in the simulator	This model must be defined to export.
Collision	Used by the simulator for calculations	Optional, but collision models of simple shapes are recommended in most cases for faster simulations. If unspecified, defaults to Visual model.
Physical	Used by the exporter to calculate COM/inertia	Optional, you can provide center of mass/inertia values instead (check "Use Custom Inertia Values"). If unspecified, defaults to Visual model.

Each component model is a set of SolidWorks components that represent the link, and the components for each model should be chosen from any one Solidworks configuration. The component models can either have their own SolidWorks configurations or can all be in the same configuration. The three types of component models are described in the table above.

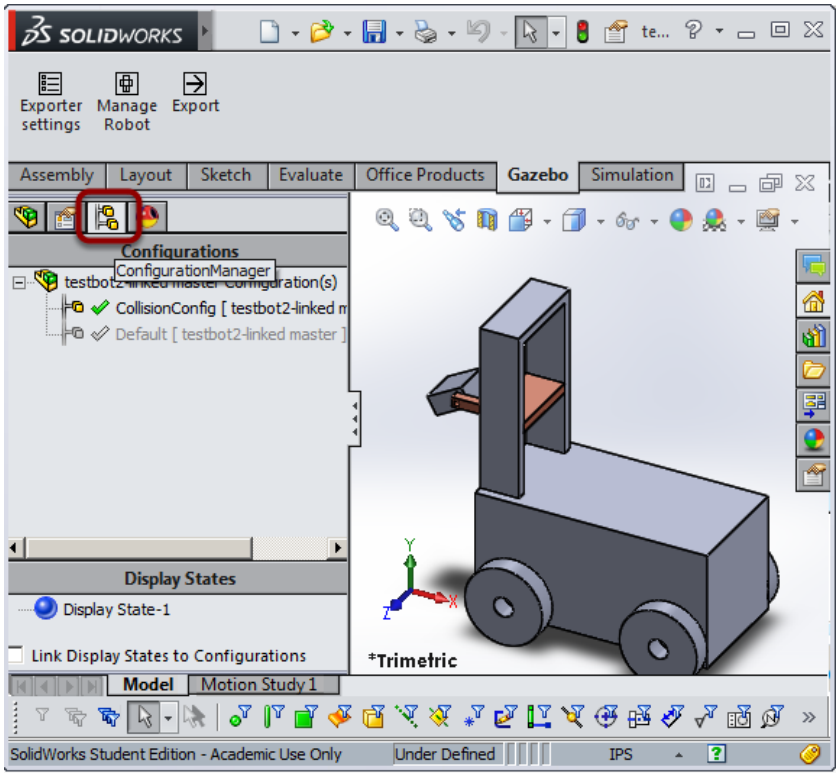
Visual Component Model - This set of link components is what is shown in the simulator. This model must be defined in order to export the robot.

Collision Component Model - This set of link components is used by the robot simulator for calculations and should be as simple as possible. For each link, if no components are assigned to the collision model, the visual model will be used instead. Although this model is optional, it is recommended for all but the most basic robots.

Physical Component Model - This set of link components is used to calculate the center of mass and inertia values of the link. For each link, if no components are assigned to the physical model,

the visual model will be used instead. If you check "Use Custom Inertia Values" for a link and provide those values, you do not need to assign components to that link's physical model.

SolidWorks Configurations

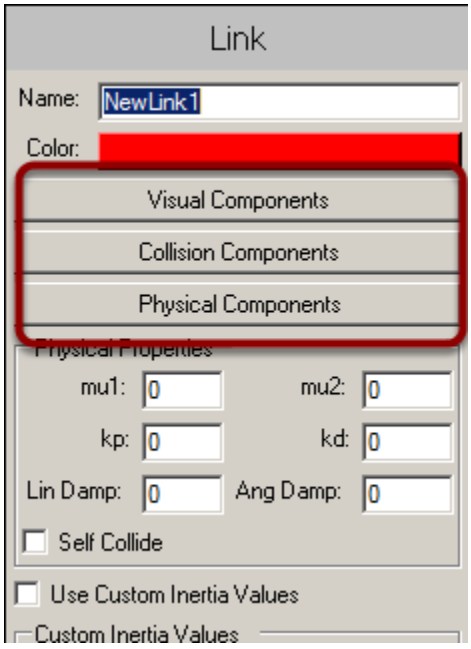


SolidWorks configurations are a useful way of organizing the components that will go in to each different component model.A SolidWorks configuration is a set of SolidWorks components that can be toggled between. A configuration can be created to store the models for the different models for export, e.g. a configuration for collision models, one for visuals, and another for physical models.

If using configurations, all components of a link's component model must be in the same configuration, although different links can have models in different configurations.

New configurations can be created in the SolidWorks "ConfigurationManager" tab.

Opening the component selector panel



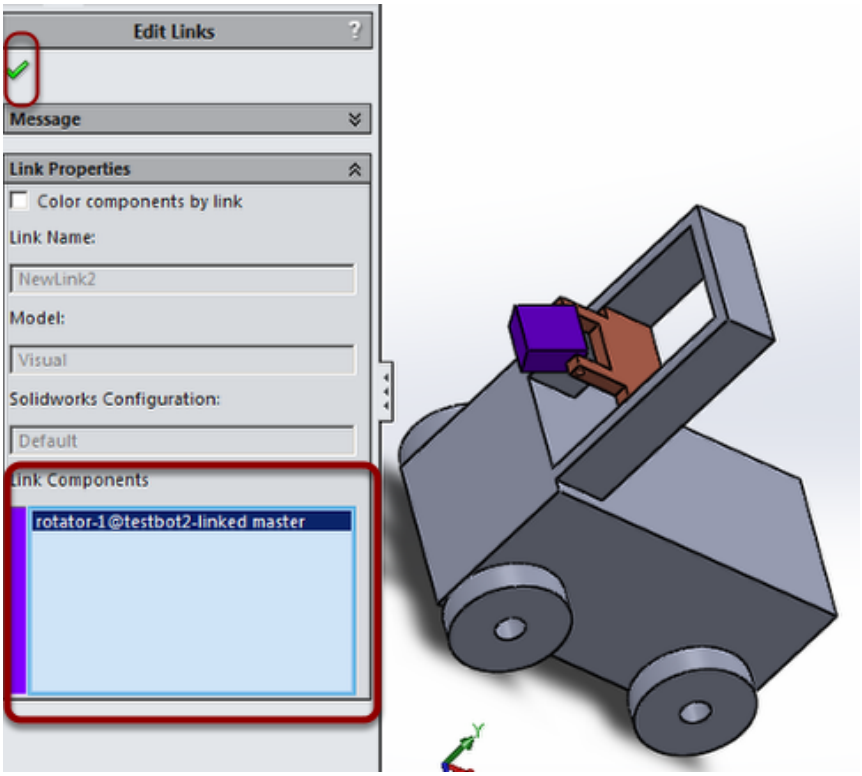
To open the panel for adding components

1) If using configurations, select the configuration in the SolidWorks Feature Manager.

2) Open the Manage Robot window and select the link

3) In the right pane, select the Physical, Visual, or Collision component model button.

Adding components

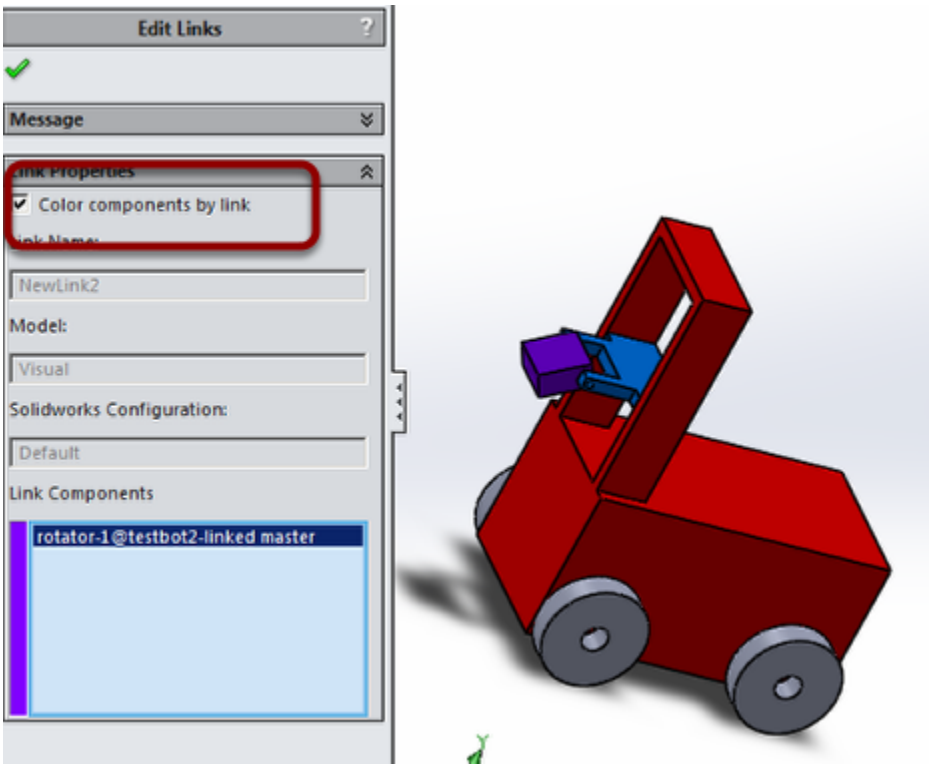


Click on a component to select or de-select it.

For collision components you can check the box labeled "No Collision model for this link ". This will make there be no collision model for this link. This can be useful to speed up simulation if you are not worried about checking for collisions on this link.

When you are done selecting components, press the green checkmark button to return to the Robot Manager and ensure that all our components are saved properly.

Coloring components



If other links have already been assigned components of the same configuration, you can see which components have been assigned to other links by checking "Color components by link".

The links will be colored based on the colors chosen in the robot manager. Note that if a component is assigned to multiple links, it will only be colored to the color of one link.

This is helpful to check which components have been assigned to at least one link.

Setting Joints

This article explains the supported types of joints and the differences between them.

Joint Types

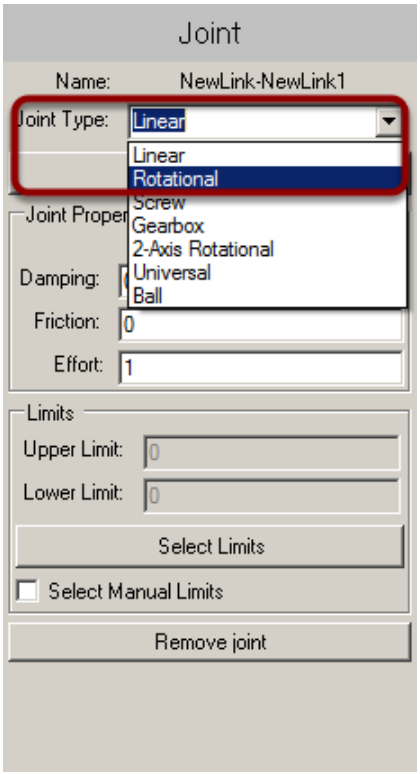
Currently, there are 7 supported joint types for exporting a robot in .SDF. Their properties (number of axis, type of motion, etc.) are summarized in the table below.

Note: Because of known problems with gearbox and screw joints in Gazebo, it is recommended to avoid using them currently. If needed gearbox joints can often be mimicked by applying a motor to both joints with the same port. This will apply a force to both joints and should make them move together.

Name	# of axes	Movement type	Limited movement?	Special properties
Linear	1	Linear	Yes	
Screw	1	Linear along and rotational around an axis	Yes	Thread pitch (radians per meter)
Rotational	1	Rotational	May be limited or continuous	
Gearbox	1*	Rotational	*	Gearbox ratio Mimicked joint
2-axis rotational	2	Rotational in two directions	1st axis may be limited or continuous. 2nd axis is continuous	
Universal	2**	Rotational	No	
Ball	0	Rotational around a point	No	

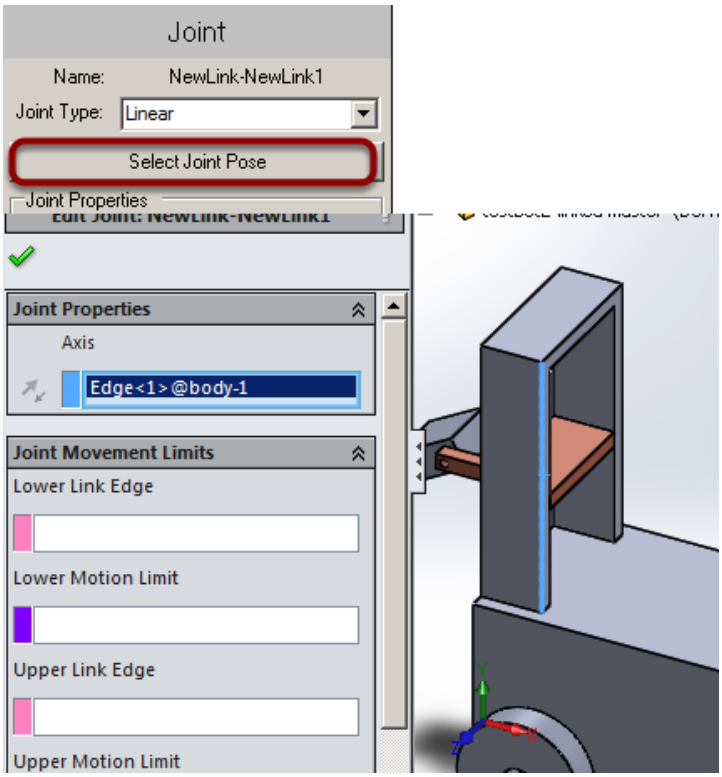
* Gearbox joints mimic other rotational joints that have parallel axis(es).
** Universal joints must have 2 perpendicular axes

Set Joint Type



Use the drop down to set what kind of joint to use.

Set Joint Axes, Limits, and Points

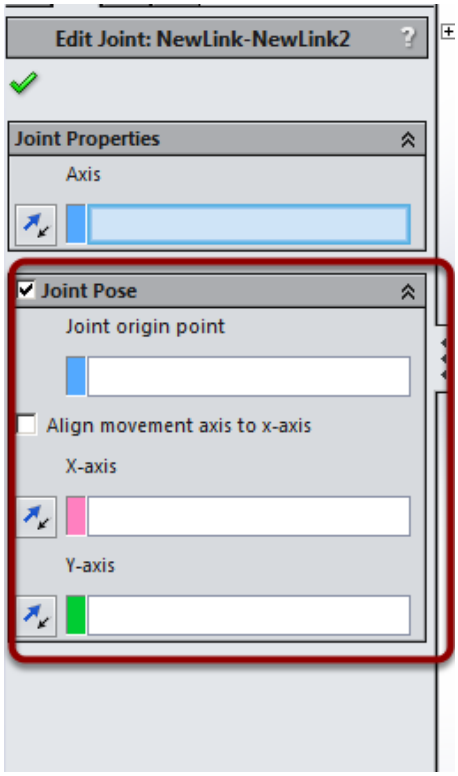


Depending on the type of joint, at least one point, axis, or plane may be necessary to define the range/limit of movement for each joint. To do so, click the "Select Joint Pose" button, which will bring you to a panel in the Solidworks window where you can use the selection boxes to click on an vertex, edge, or plane of the model, as necessary. For axes, use the toggle direction button to flip the direction of movement.

Note that these objects must be selected from the visual configuration model (not the collision models).

Click the green checkmark to return to the "Manage Robot" window.

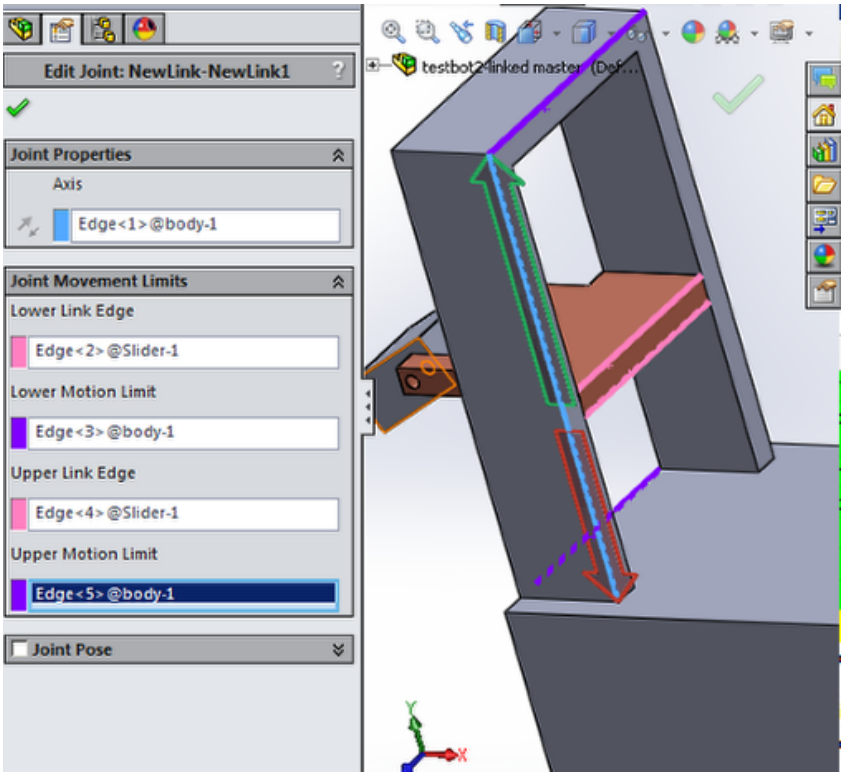
Set Joint Pose (Optional)



In addition, you may specify the position and frame of reference of the joint by checking the Joint Pose checkbox and selecting a point and 2 axes. Note that if you select a plane for the x-axis or y-axis, the axis will be in the direction of the normal vector of that plane.

This step is optional and should only be completed by advanced users. If no joint pose is specified, it will be automatically calculated and set.

Set Joint Limits



The limits of a joint can be set graphically by un-checking "Select Manual Limits" and pressing the "Select Limits" or "Select Joint Pose" buttons. To define the limits a motion limit and an edge must be defined in each direction. The edge defined as the point on the link itself that will move to the limit. The motion limit is a point not on the link that the edge will move to at its furthest extent. Through this method, the limit values are calculated automatically for you.

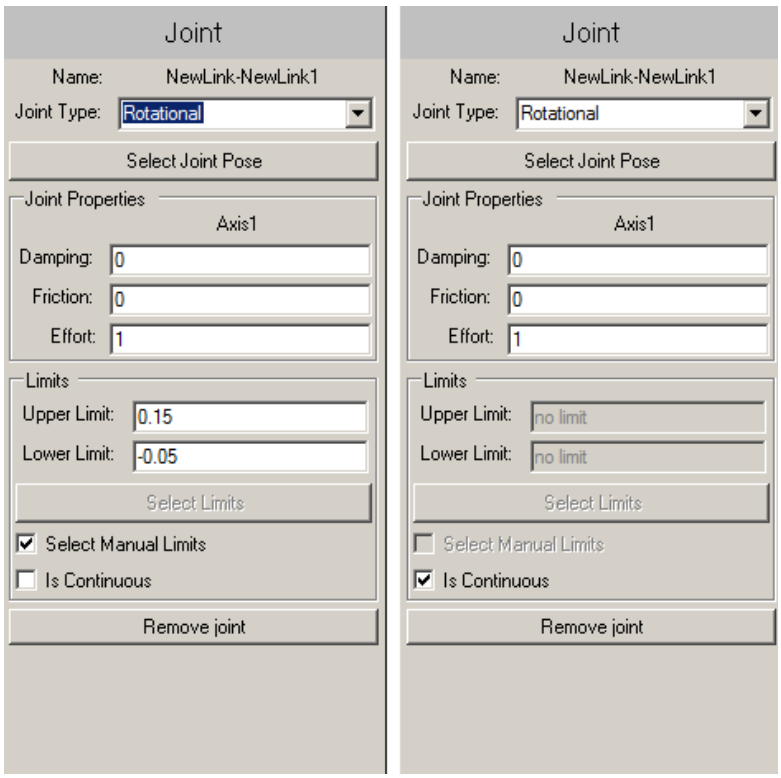
The green preview arrow represents positive movement of the joint, and the red arrow represents negative movement.

If the link will only move in one direction, only the limits in that direction need to be selected.

For rotational limits:

On rotational joint the limits will define the angular movement of the link. Because of this, the selection will define itself on to one of the 2 sides of the axis of revolution. The direction can be seen with the pink and purple lines that are projected on to the model. If the limit is on the wrong side of the axis, simply click on flip selections button next to the corresponding selection box to flip the limit to the other side of the axis.

Set Joint Limit Values



The motion limit values (in meters and radians, respectively) may be set directly by checking "Select Manual Limits" and inputting their value in the text boxes. Note that the lower motion limit must be a negative value and the upper motion limit must be a positive value.

Furthermore, for rotational joints, if a joint axis has no motion limits and rotates infinitely, check the "Is Continuous" checkbox. Note that Universal joints are defined as having infinite rotation around both axes, so you do not need to set limits for this joint; similarly, Revolute2 joints are defined as having infinite rotation around at least one axis (Axis 2), and Ball and socket joints have infinite rotation in any direction around a point.

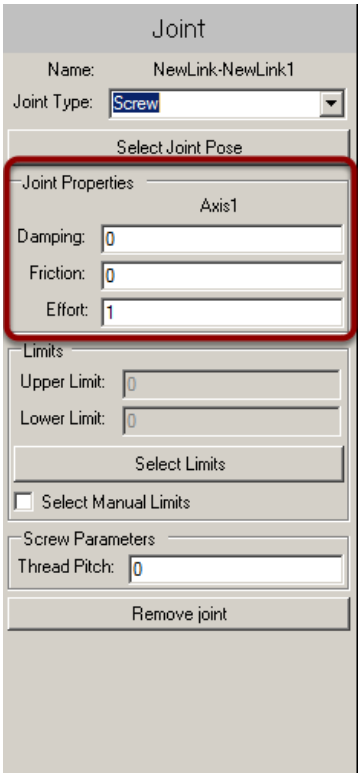
Set Joint properties

In addition to the motion limits, damping (in Newton-seconds/meter), static friction value (in Newtons), and effort limit values should be specified for each axis of each joint. The effort limit is the maximum force or torque (in Newtons or Newton-meters) that can be applied to a linear or rotational joint, respectively.

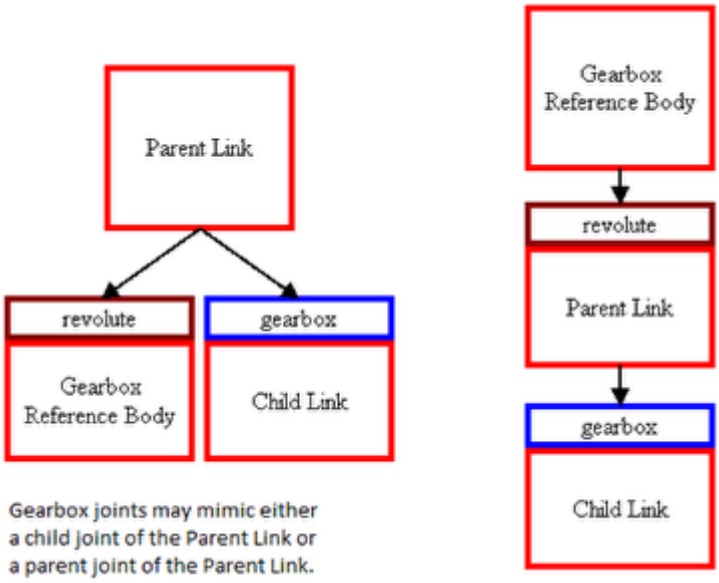
Due to a bug in gazebo, high damping values on lightweight models can sometimes cause the model to break. If the model apears to jump to the origin of the world and be imobile

immediately on being placed in the world, lower the damping values to see if this corrects the problems.

Certain joints, including screw joints and gearbox joints, may have additional properties. See the table above. Note that in Gazebo, thread pitch is defined as the amount of thread revolutions (in radians) divided by the length of the screw (in meters).



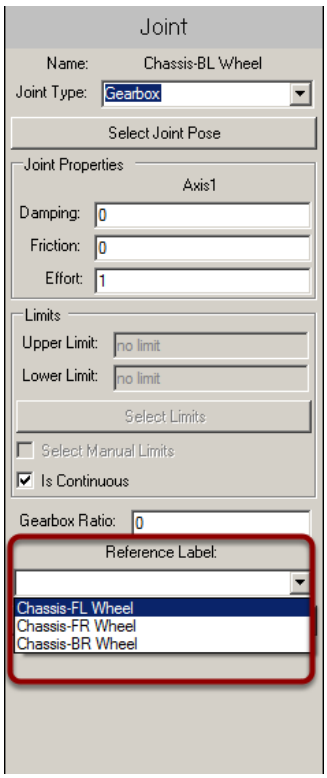
Gearbox joints



Gearbox joints are unique because they can be used to mimic other rotational joints. You can only mimic a joint that is connected either as a parent or a child to the parent of the gearbox joint.

Note: Unless you are an experienced Gazebo user, it is highly recommended that you just make two rotational joints with the same property values instead of mimicking a joint with a gearbox joint. These two joints can then be linked by putting an actuator on both joints and putting them both on to the same port.

Mimicking a joint with a gearbox joint

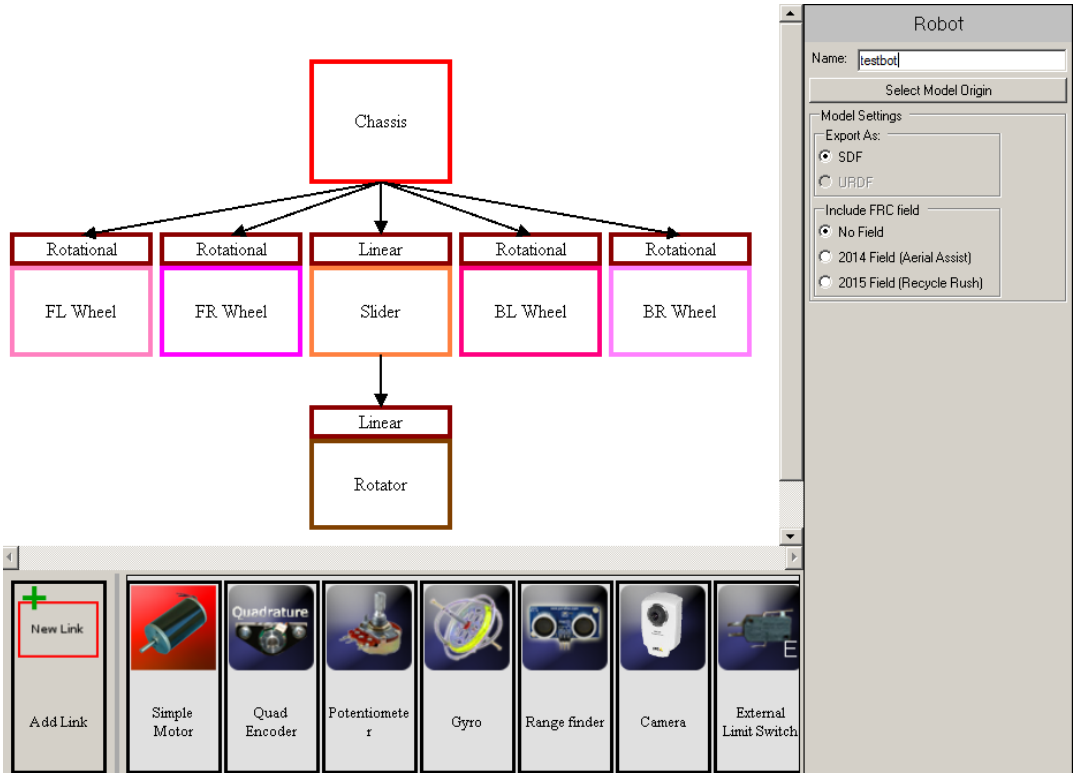


To use a gearbox joint, first check that the axis of the joint that you want to mimic has been defined. Then, define the axis on the gearbox joint, which should be parallel to the axis of the mimicked joint, and select the joint from the dropdown box. This will tell the gearbox joint to link to the mimicked joint. Finally, set the gearbox ratio in the textbox. A positive gear ratio will make the two joints rotate in opposite direction, like a gear, while a negative ratio will make the joints rotate in the same direction, like a sprocket and chain.

Adding Motors and Sensors to a Model

This tutorial will show how to add sensors and motors to a model to be used with FRCsim

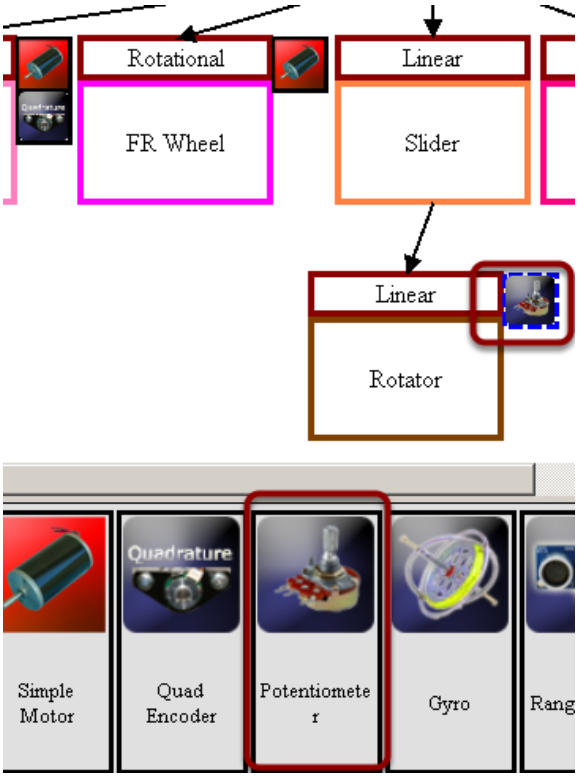
Define Links and Joints



Before adding in any attachments, you must check "enable FRCsim Components" in the Exporter Settings.

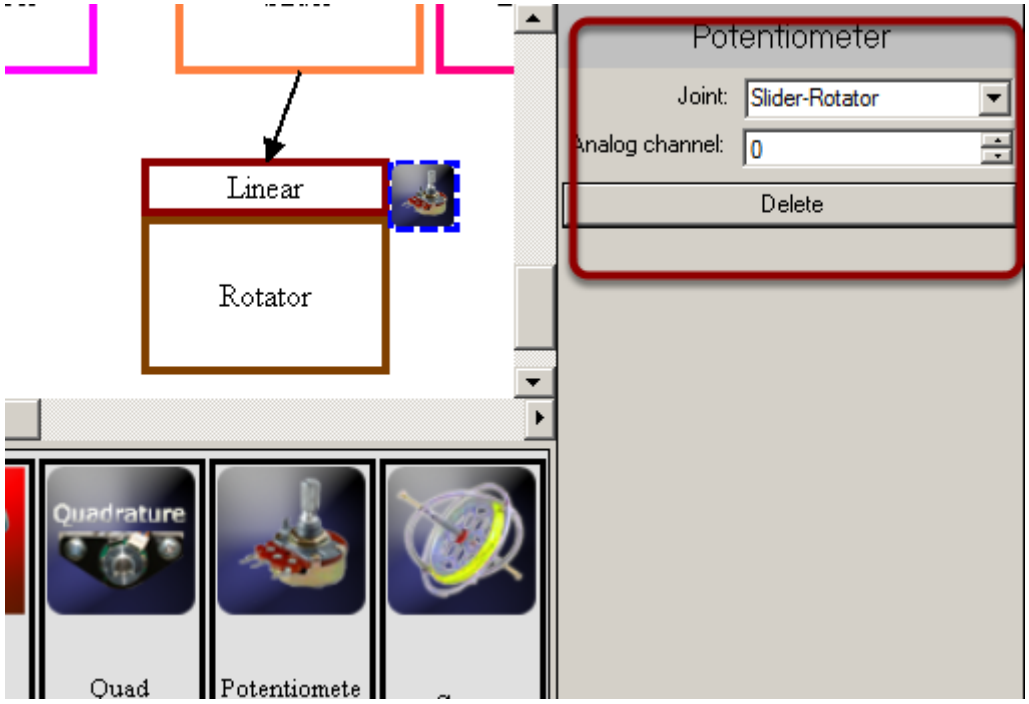
Also, the model should first have all of its links and joints defined. A tutorial on how to do this can be found here under the "Define Joints" section: http://wpilib.screenstepslive.com/s/4485/m/23353/l/349026-exporting-a-solidworks-model-to-gazebo. Once those have been defined, open the Manage Robot window.

Add Attachments



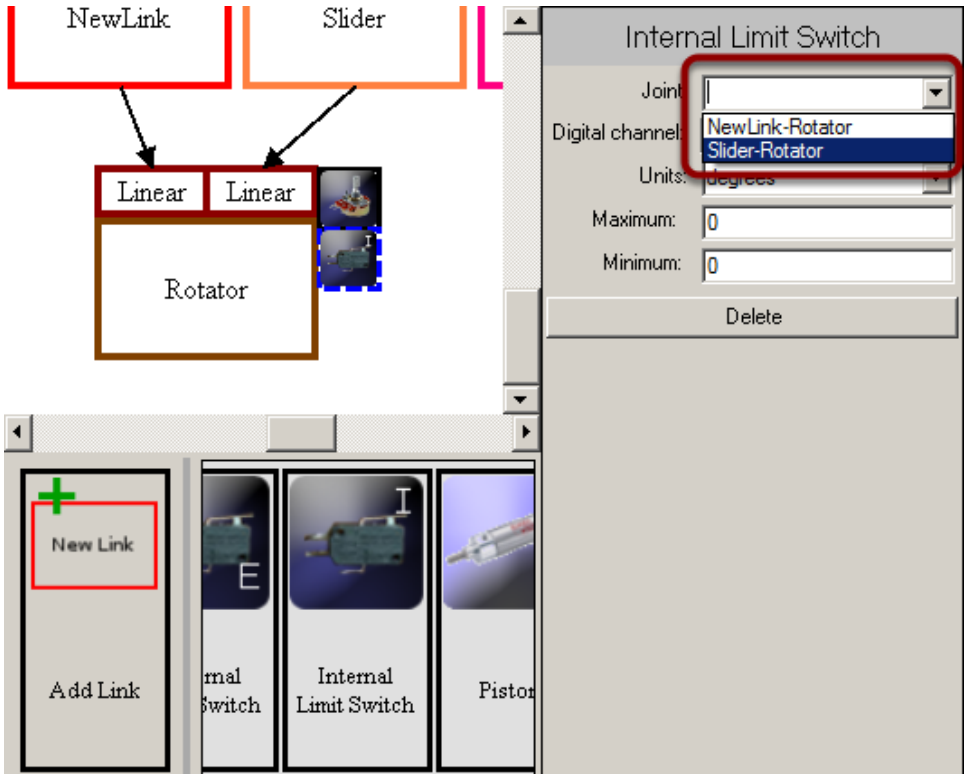
To add an attachment, simply drag the desired type of attachment from the bottom bar onto the link you want to apply it to. If an attachment affects a joint, it will affect the joint between the link it is on and its parent link, and it cannot be put on the base link. Attachments that affect joints are: simple motor, quadrature encoder, potentiometer, internal limit switch, and piston.

Edit Attachment Properties



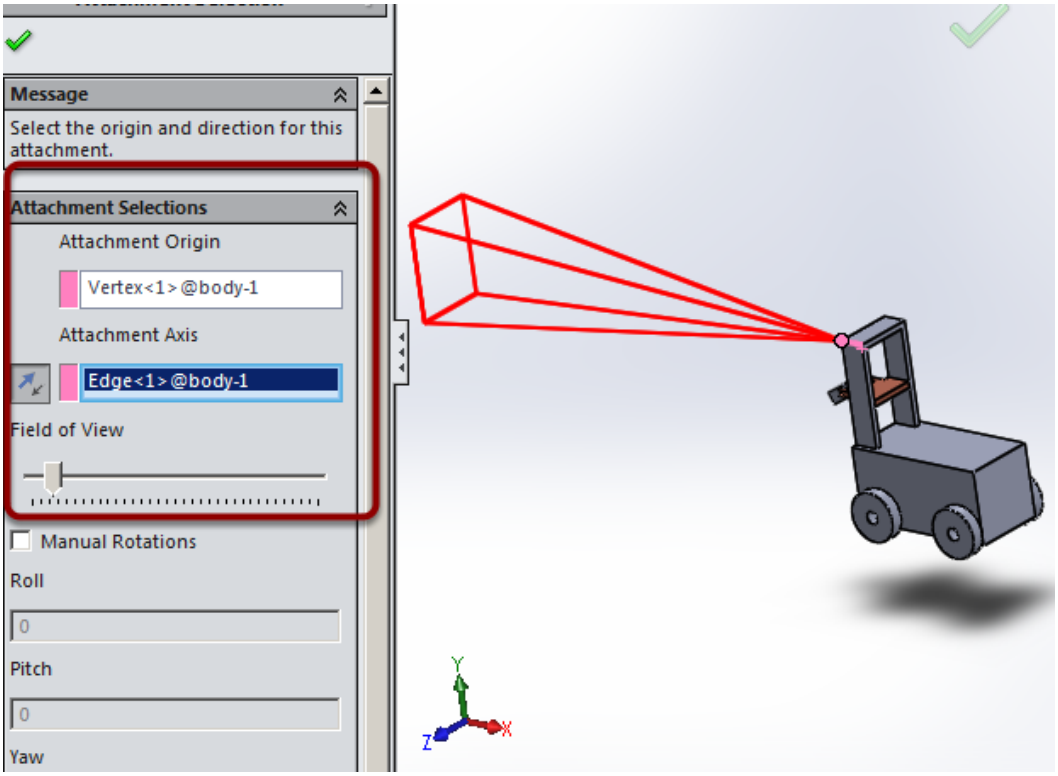
To edit the properties of an attachment, simply click on the desired one in the graph window. A property page will appear on the right side of the window with boxes to edit the various properties for each attachment. The attachment can also be deleted here by pressing the Delete button.

Attachments for joints



Some attachment types must be associated with a specific joint chosen from the parent joints of the link it is on. To do so, select the name of the joint in the form *ParentJoint-ChildJoint* in the dropdown box.

Set the Attachment Position



For some attachments (rangefinder and camera), you must specify the point on the robot where it is located and a radius/field of view by clicking the "Attachment Pose" button, which opens the Solidworks window where you can select a point on the model.

To define an attachment's location, click on any vertex or reference point in the physical configuration model (not the collision models).This point will be the location that the attachment is located at.

To define the direction it looks, you can click on any edge or axis and toggle the direction button as necessary, or you can manually specify the rotations to be applied. The rotations are applied in the order of roll, then pitch, then yaw. Once a location and direction are specified a preview will appear showing the view of the attachment. The field of view of the attachment can also be adjusted using the slider.

List of currently supported attachments

The currently supported attachments, their properties, their units, and other details are shown in the table below. Note: Any joint attachment applied to a joint with more than one axis of movement will only effect the first axis of the joint.

TABLE: Currently supported attachments

	NAME	PROPERTIES (UNITS)	Notes*
JOINT ATTACHMENTS	Simple Motor	PWM channel	
		Multiplier (Newtons or Newton-meters)	the motor's output to the joint will range from -(Multiplier) to +(Multiplier)
	Quad Encoder	A Output channel	
		B Output channel	
		Pulses per rev/m	
	Potentiometer	Analog channel	
	Internal Limit Switch	Digital channel	
		Units	Defaults to degrees
		Maximum (the units specified)	
		Maximum (the units specified)	
	Piston	Channel	
		Direction	Defaults to forwards
		Forward Force (Newtons)	
		Reverse Force (Newtons)	
LINK ATTACHMENTS	Gyro	Analog channel	
		Axis	Defaults to pitch
	Rangefinder	Analog channel	
		Radius (meters) - the radius of the cone of view at 1 meter distance	Can also set with slider (click Attachment Pose)
		Minimum distance (meters)	
		Maximum distance (meters)	
		Location	
		Attachment position and axis or Manual rotation (degrees)	(click Attachment Pose)
	Camera	Horizontal Field of View (m)	Can also set with FOV slider (click Attachment Pose)
		Attachment position and axis or Manual rotation (degrees)	(click Attachment Pose)
	External Limit Switch	Digital channel	

*All default values are 0 unless otherwise noted.

Video Tutorial

A video on tuning motors in Gazebo can be found here:

https://www.youtube.com/playlist?list=PL9HqYJ1lkIKW21ULeI4mUjaj0UIzRJhWf