Using FRCSim with C++ and Java

FRC
FIRST® Robotics Competition

Table of Contents

Articles

Installing Ubuntu

At this time, to run FRCSim, it is necessary to install Ubuntu 14.04 (64bit). We recommend following this tutorial: https://help.ubuntu.com/community/Installation/FromUSBStickQuick

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

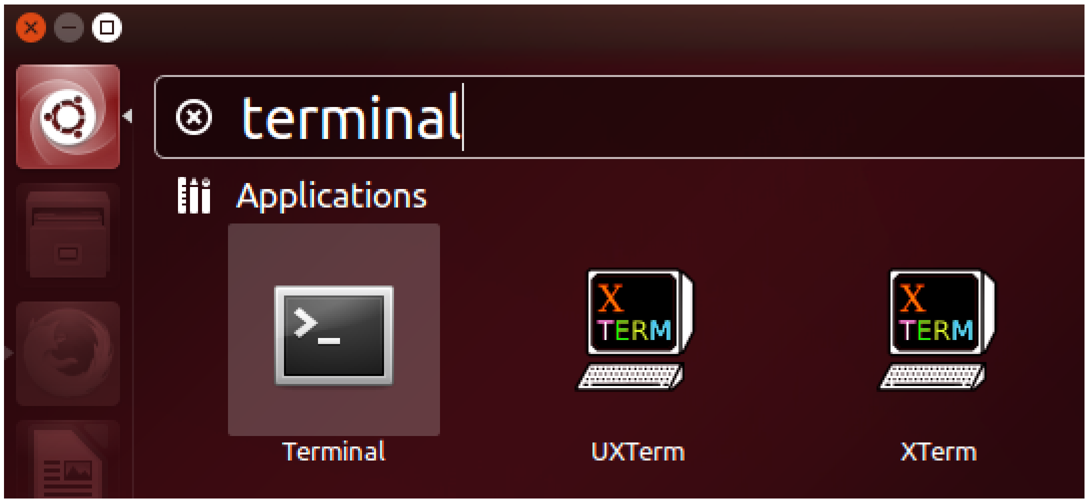
Running Ubuntu on the same computer as Windows

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/WindowsDualBoot.

Installing FRCSim with a Terminal

Open Terminal

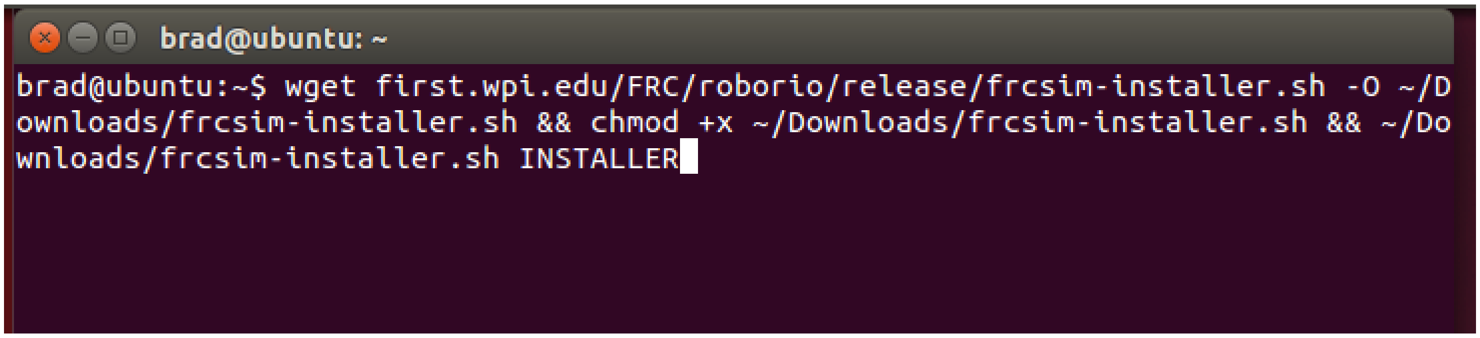


1. Click on the Ubuntu icon (just below the top-left corner)

2. Type “terminal”

3. Click on the terminal application

Run the installer



```
brad@ubuntu: ~
brad@ubuntu:~$ wget first.wpi.edu/FRC/roborio/release/frcsim-installer.sh -O ~/D
ownloads/frcsim-installer.sh && chmod +x ~/Downloads/frcsim-installer.sh && ~/Do
wnloads/frcsim-installer.sh INSTALLER
```

1. Copy the following text (note that the following ine is a single command that you can copy and paste into the terminal window):

```
wget first.wpi.edu/FRC/roborio/release/frcsim-installer.sh -O ~/Downloads/frcsim-installer.sh &&
chmod +x ~/Downloads/frcsim-installer.sh && ~/Downloads/frcsim-installer.sh INSTALLER
```

2. Paste it into the terminal

3. Hit enter

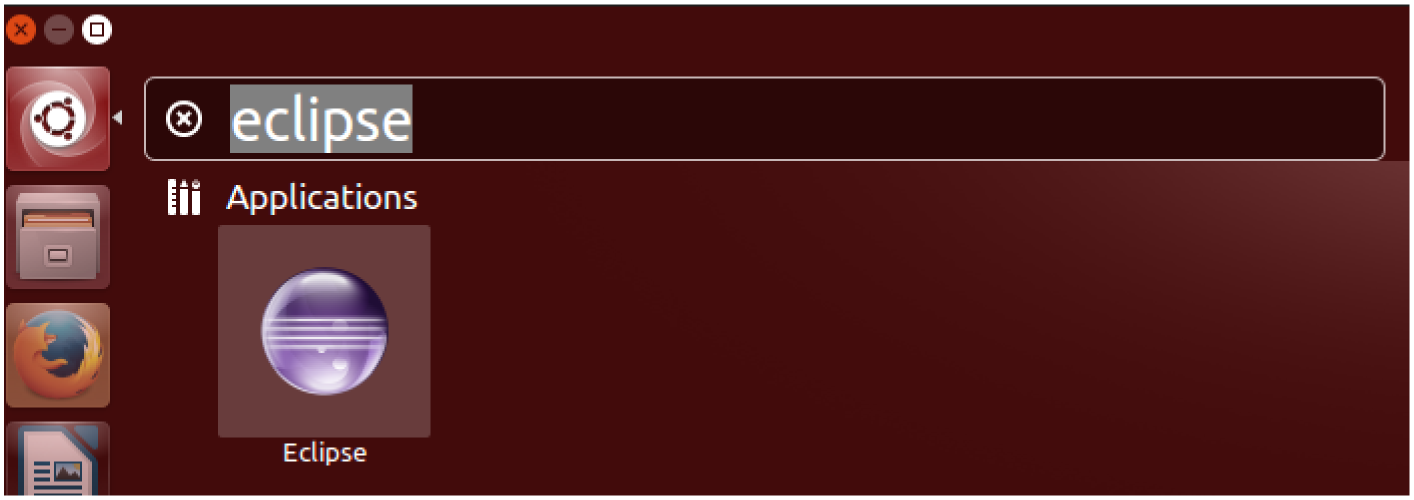
4. Type in your password and wait

5. Installation is successful

Simulating GearsBot with FRCSim

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

Open Eclipse

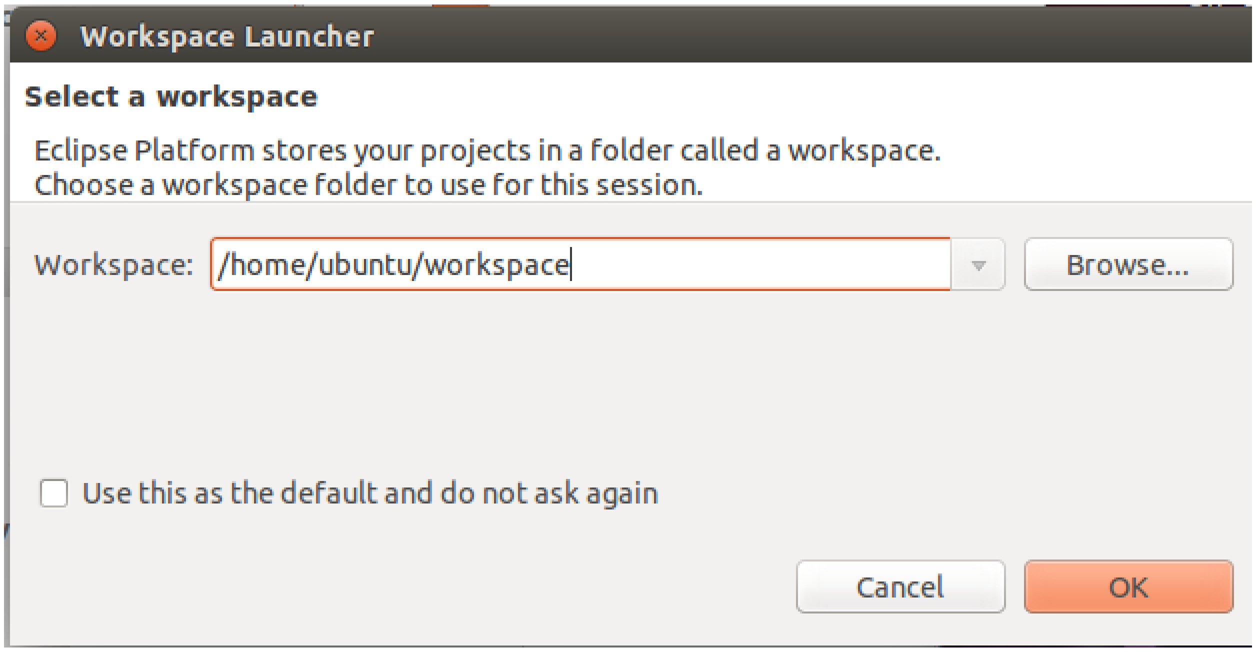


1. Click on the menu

2. Type “eclipse”

3. Click on the eclipse icon

Open the workspace



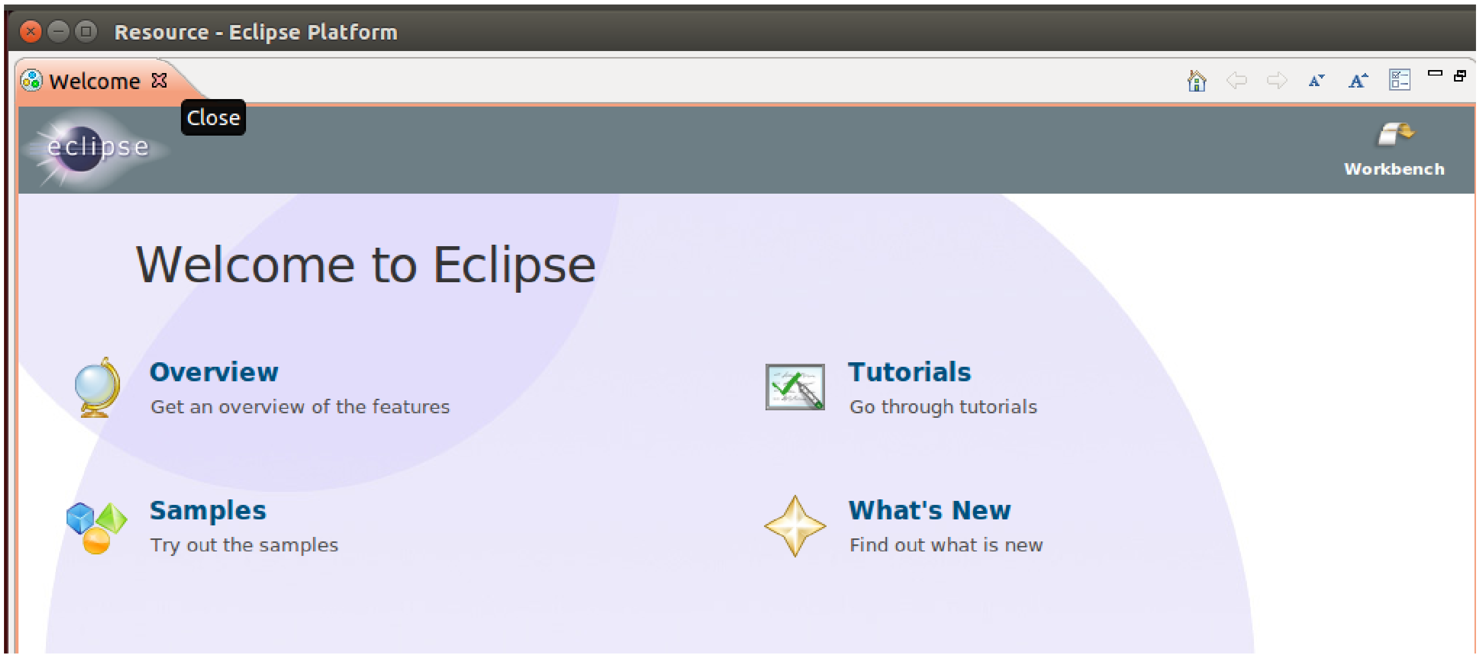
Select a workspace

Eclipse Platform stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: /home/ubuntu/workspace

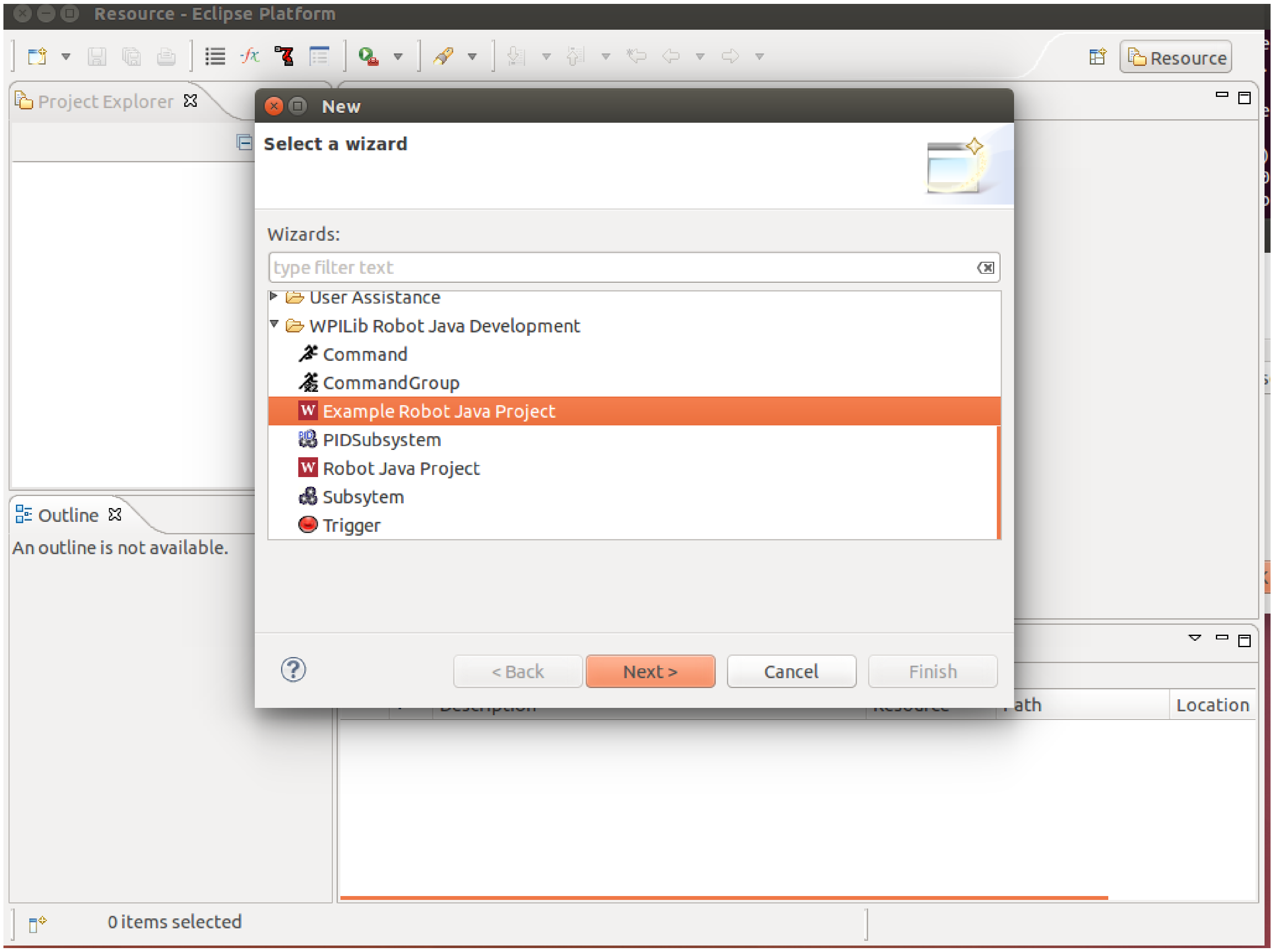
Click OK

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

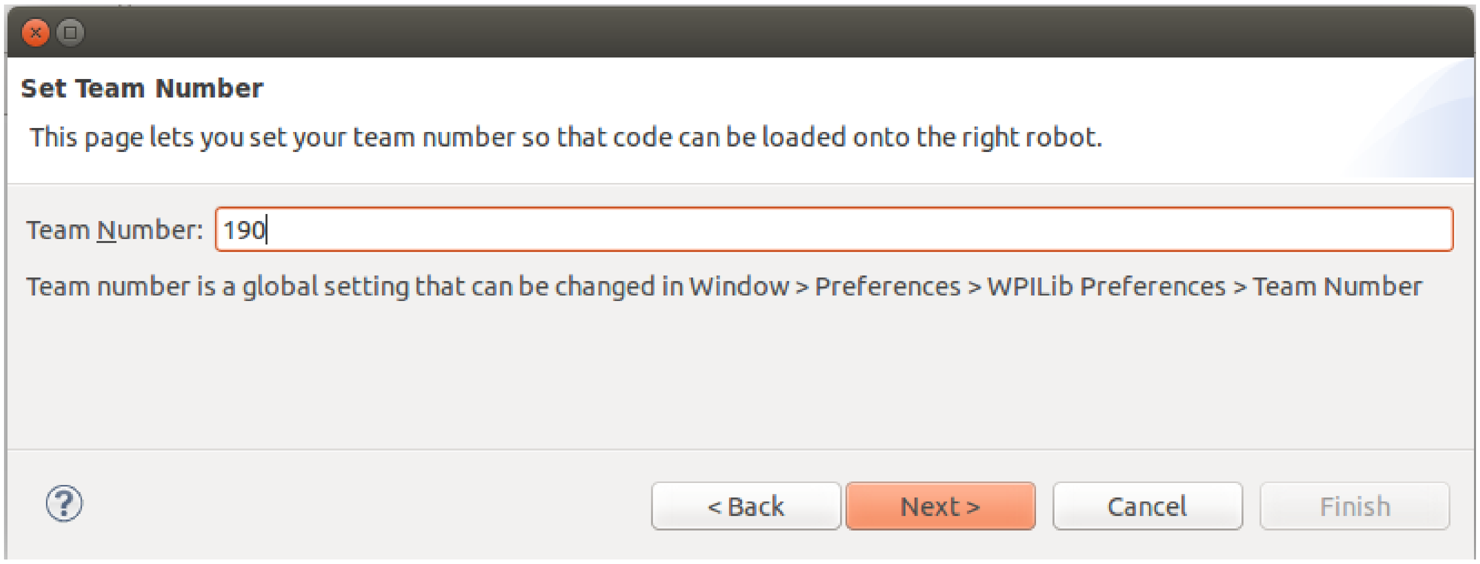
3. If using C++

(a) Expand WPILib Robot C++ Development

(b) Select Example Robot C++ Project

4. Click Next

Set your team number


Set Team Number

This page lets you set your team number so that code can be loaded onto the right robot.

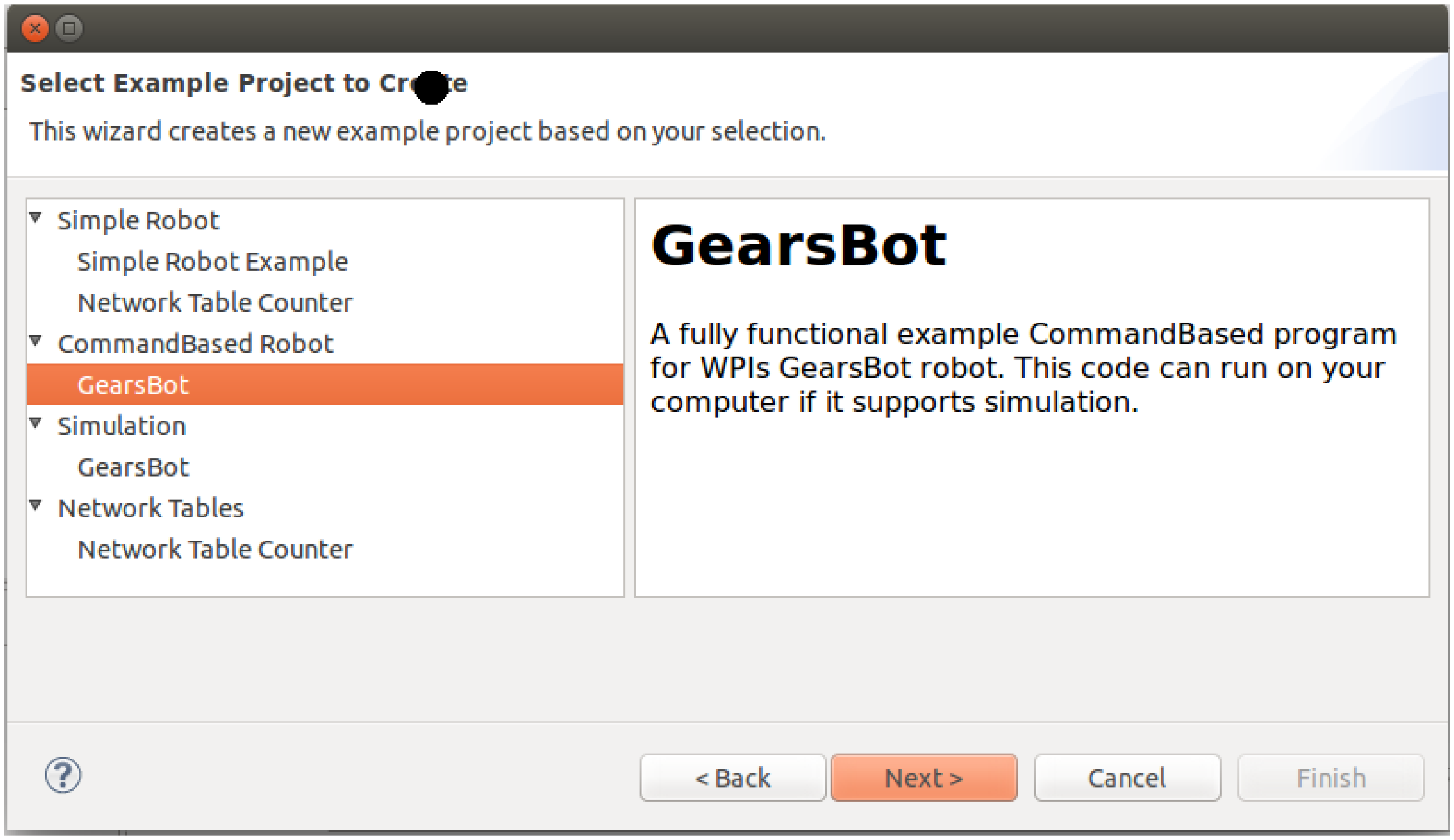
Team Number: 190

Team number is a global setting that can be changed in Window > Preferences > WPILib Preferences > Team Number

1. Enter your team number

2. Click Next

Choose the GearsBot example


Select Example Project to Create

This wizard creates a new example project based on your selection.

- Simple Robot
  - Simple Robot Example
  - Network Table Counter
- CommandBased Robot
  - GearsBot
- Simulation
  - GearsBot
- Network Tables
  - Network Table Counter

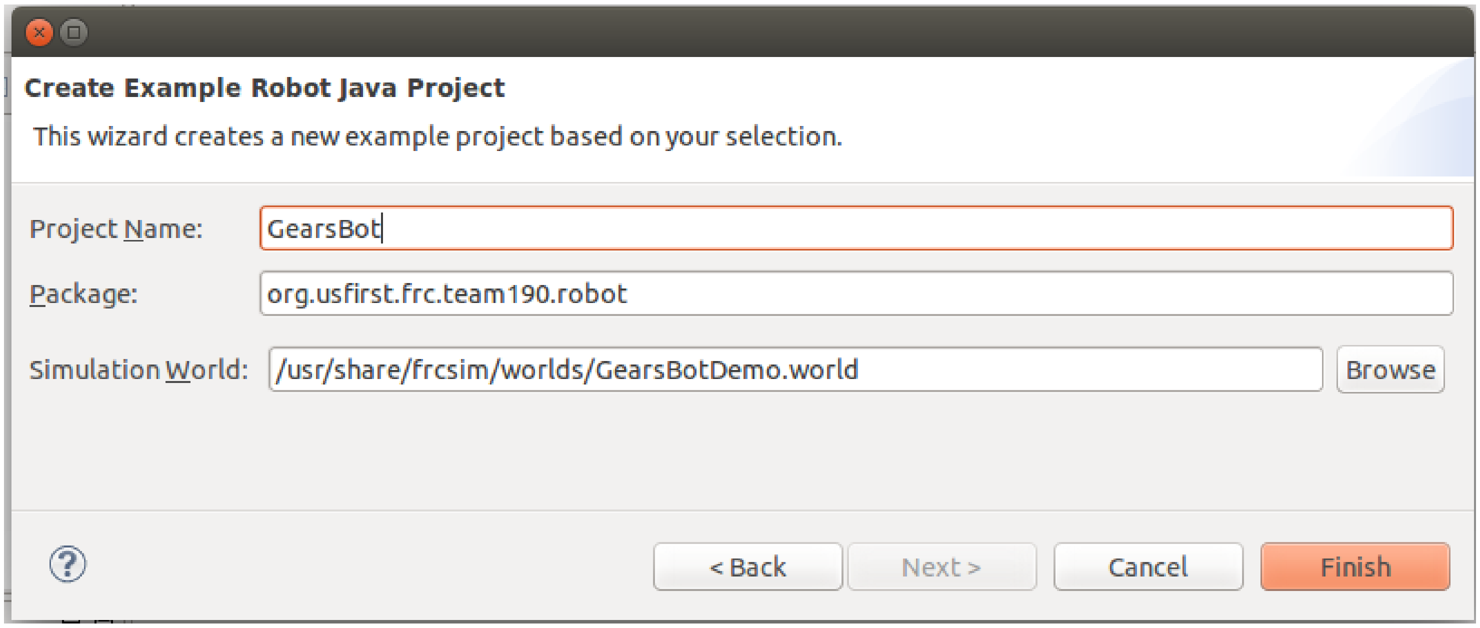
GearsBot

A fully functional example CommandBased program for WPIs GearsBot robot. This code can run on your computer if it supports simulation.

1. Select the GearsBot example

2. Click Next

Naming the project


Create Example Robot Java Project

This wizard creates a new example project based on your selection.

Project Name: GearsBot

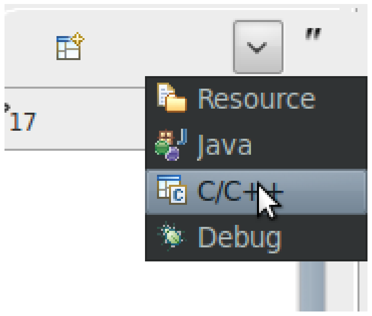
Package: org.usfirst.frc.team190.robot

Simulation World: /usr/share/frcsim/worlds/GearsBotDemo.world

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

2. Click Finish

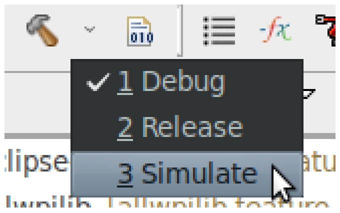
Switching to the C++ Perspective (for C++ programs only)



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

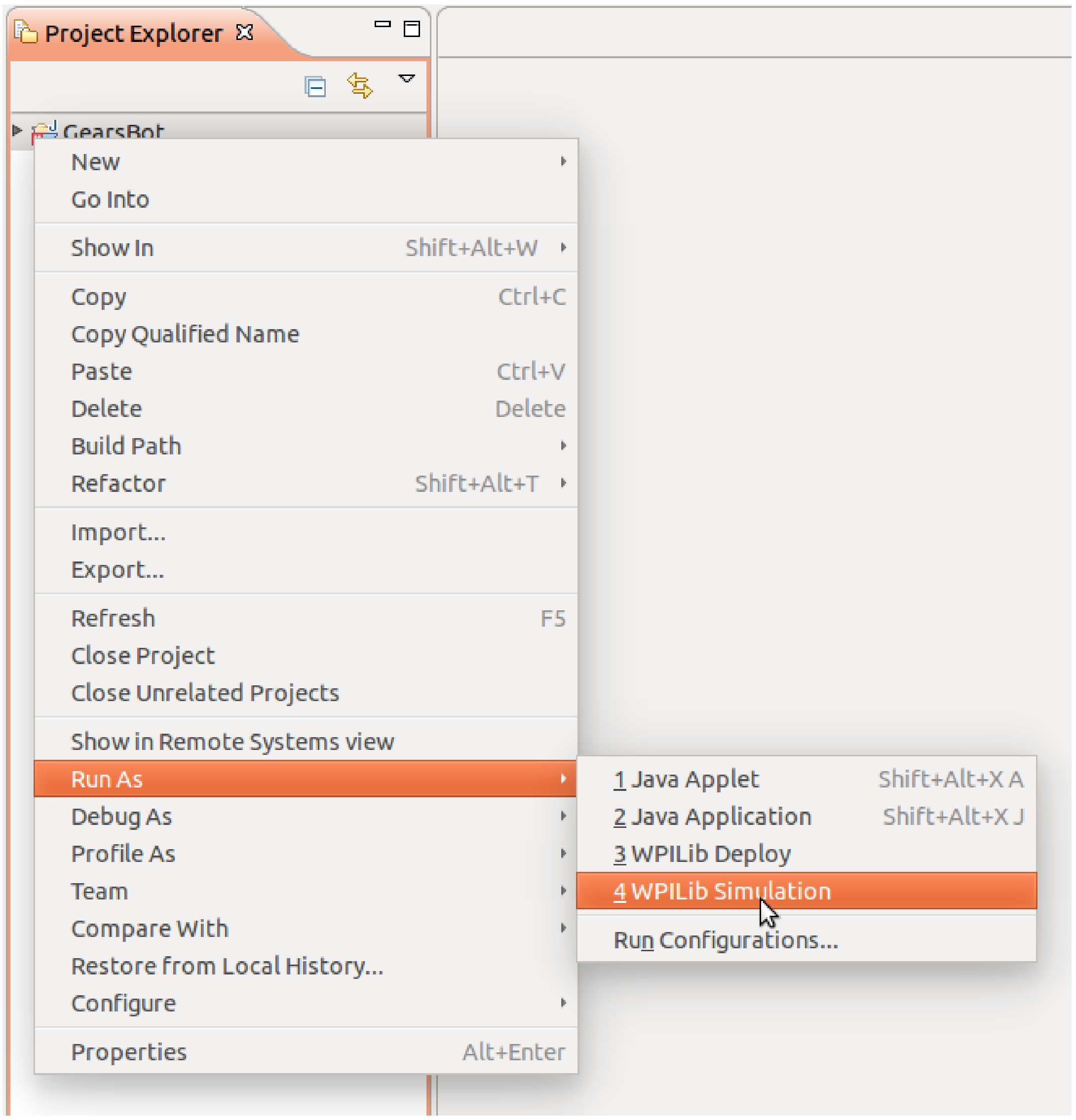
1. In the upper right corner click on the arrow.
2. Select C/C++

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 Simulate to set the default for your robot simulation project.

Running the simulator

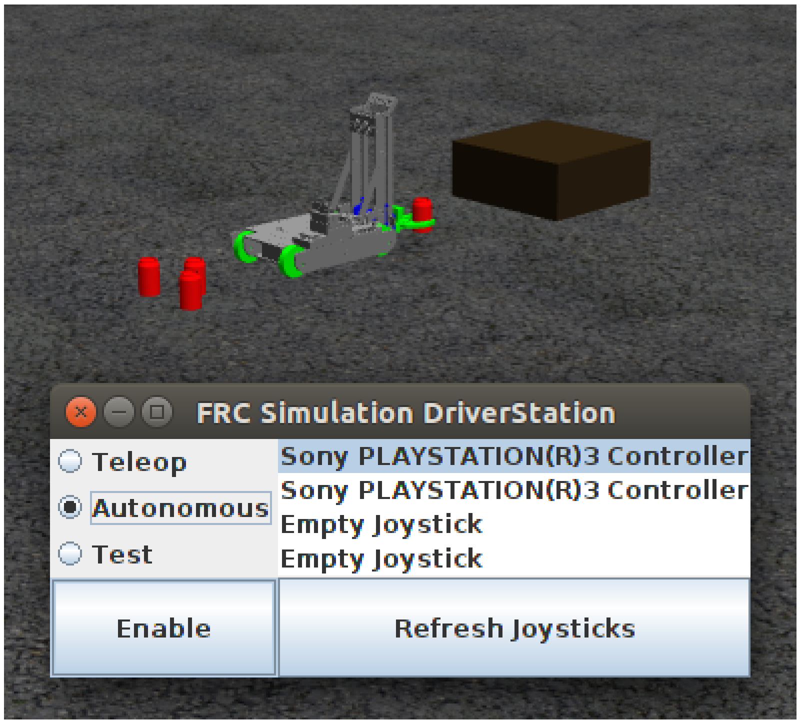


1. Right click on the project

2. Select Run As

3. Click WPILib Simulation

Running Autonomous


FRC Simulation DriverStation
Teleop
Autonomous
Test
Sony PLAYSTATION(R)3 Controller
Sony PLAYSTATION(R)3 Controller
Empty Joystick
Empty Joystick
Enable
Refresh Joysticks

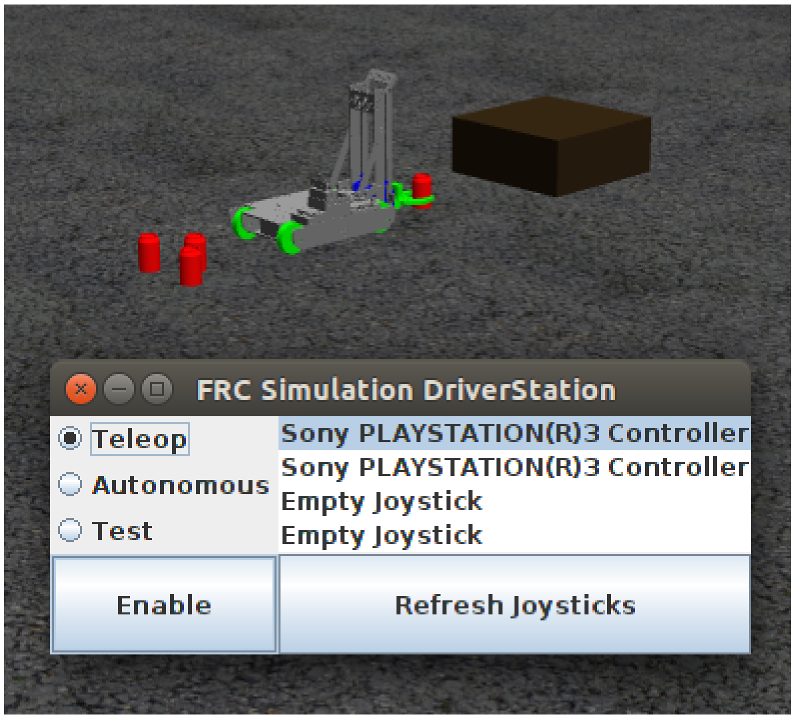
1. Make sure the simulator is running

2. Select Autonomous

3. Click Enable

4. You should observe something like this video.

Using a PS3 joystick to drive

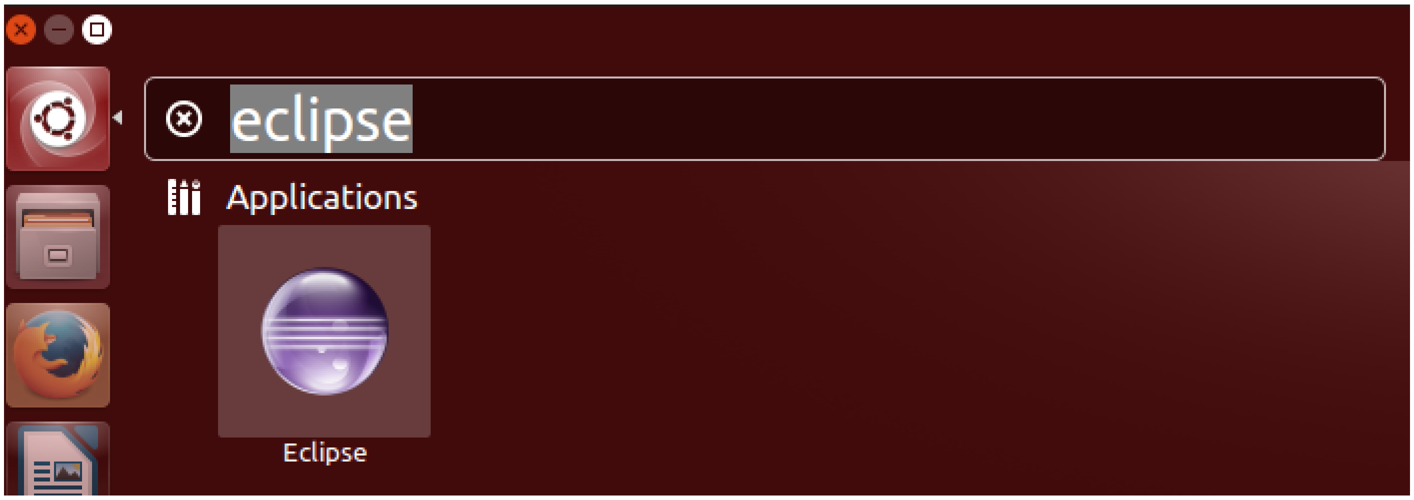

FRC Simulation DriverStation
Teleop
Autonomous
Test
Sony PLAYSTATION(R)3 Controller
Sony PLAYSTATION(R)3 Controller
Empty Joystick
Empty Joystick
Enable
Refresh Joysticks

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

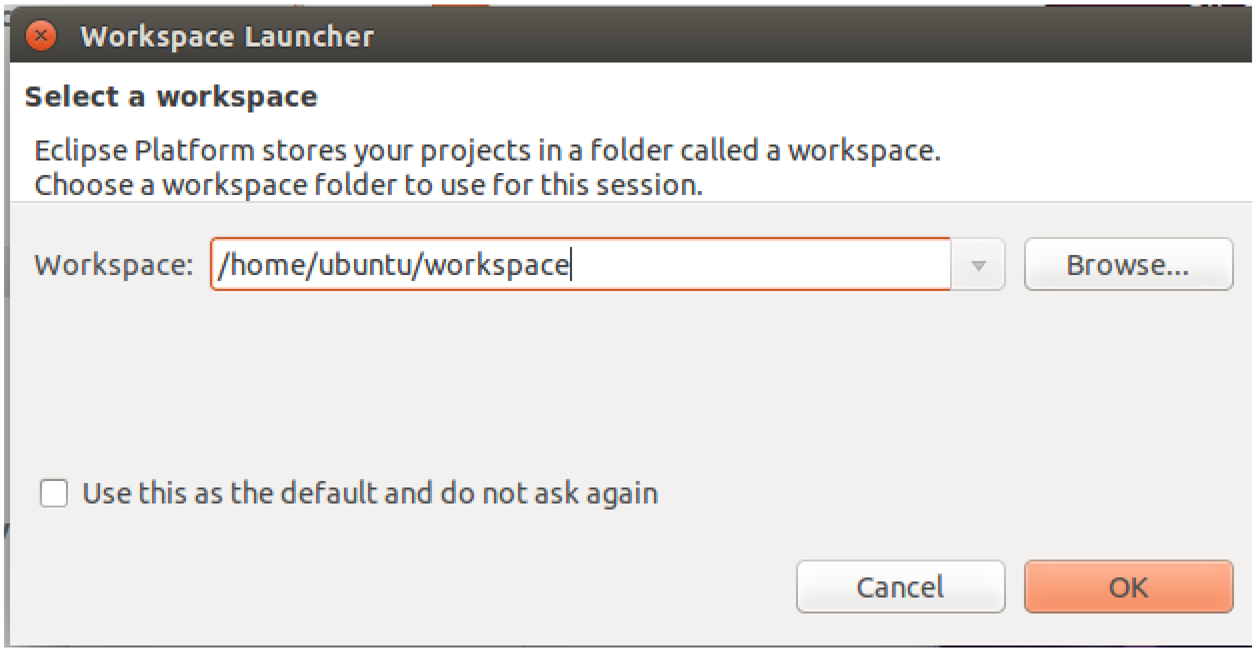


1. Click on the menu

2. Type “eclipse”

3. Click on the eclipse icon

Open the workspace


Workspace Launcher

Select a workspace

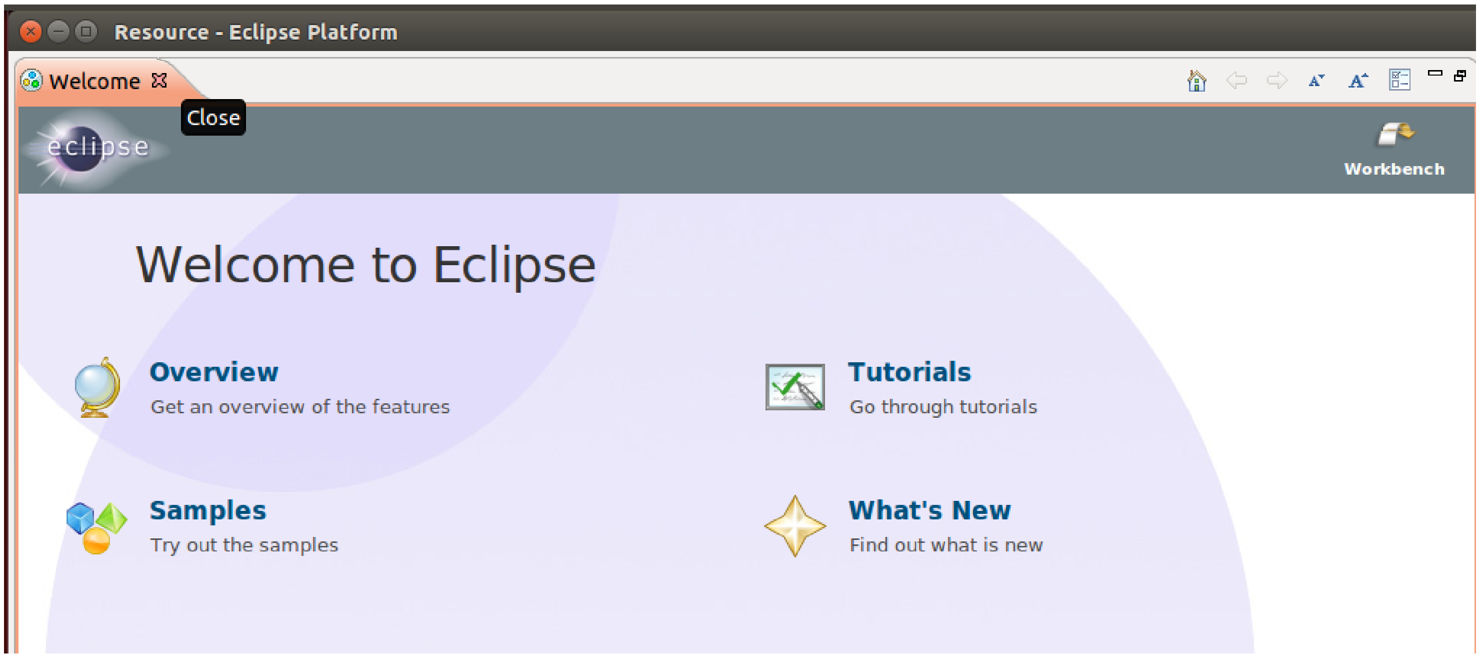
Eclipse Platform stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: /home/ubuntu/workspace

☐ Use this as the default and do not ask again

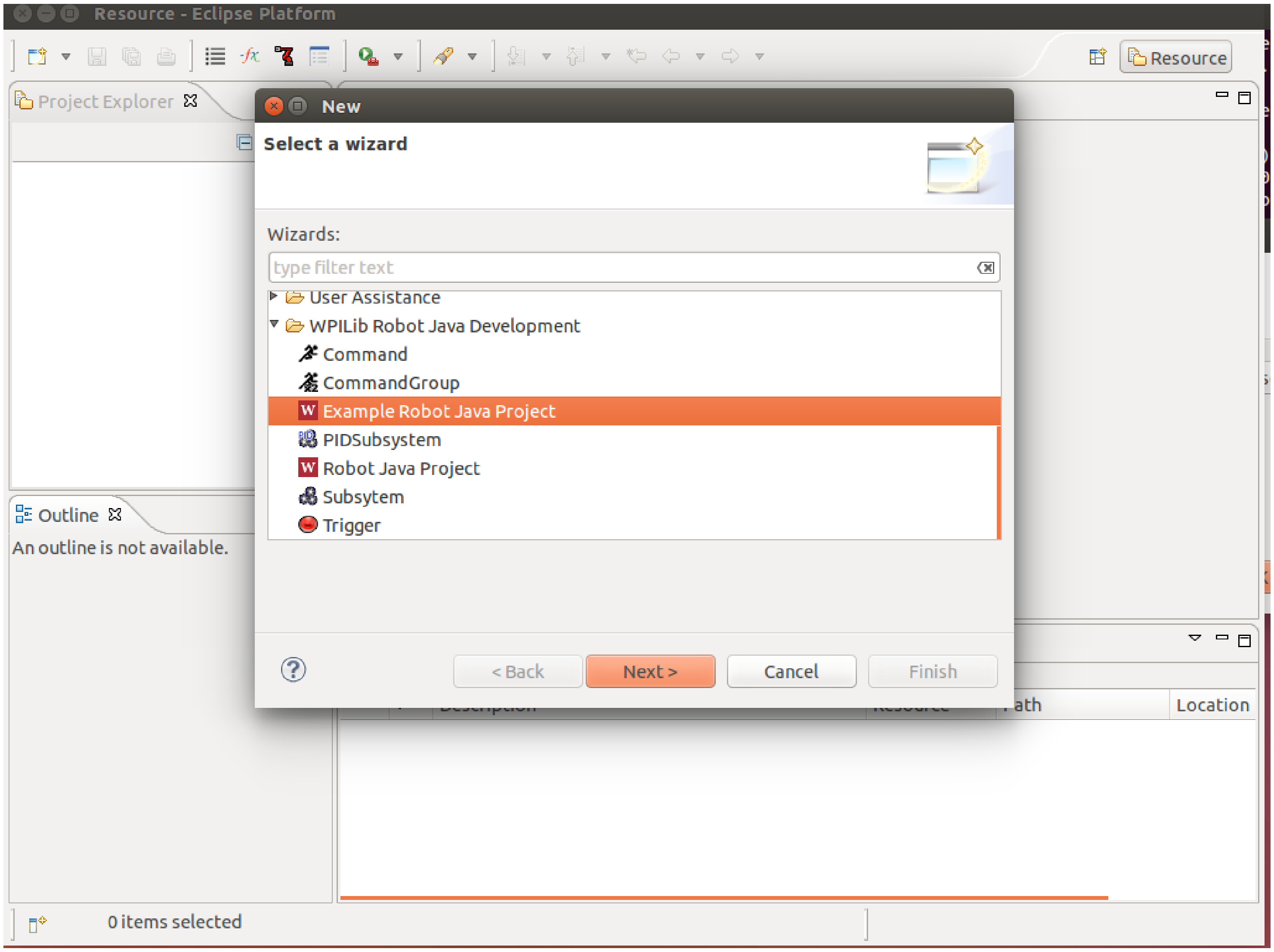
Click OK

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

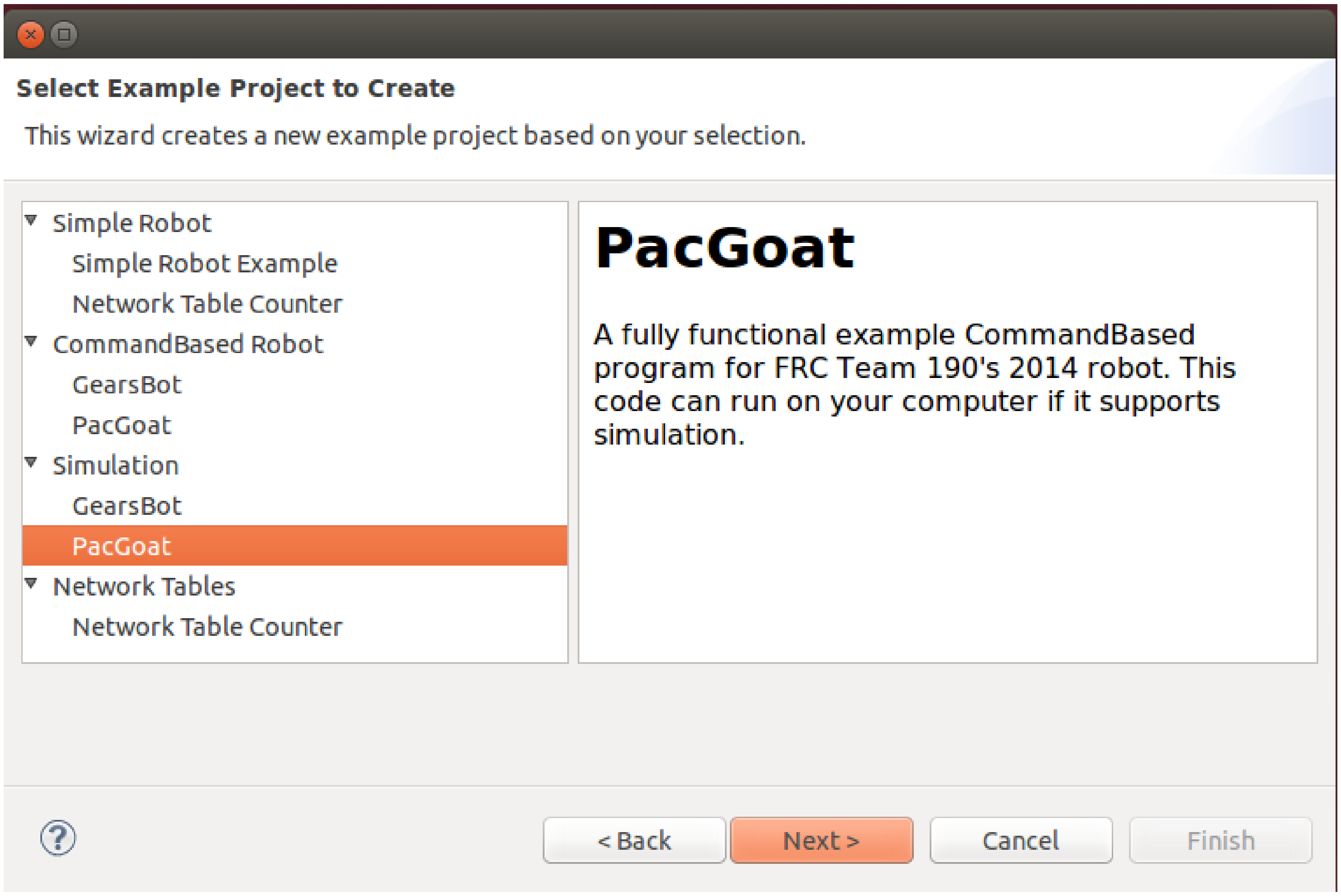
3. If using C++

(a) Expand WPILib Robot C++ Development

(b) Select Example Robot C++ Project

4. Click Next

Choose the PacGoat example



Select Example Project to Create

This wizard creates a new example project based on your selection.

- Simple Robot
  - Simple Robot Example
  - Network Table Counter
- CommandBased Robot
  - GearsBot
  - PacGoat
- Simulation
  - GearsBot
  - PacGoat
- Network Tables
  - Network Table Counter

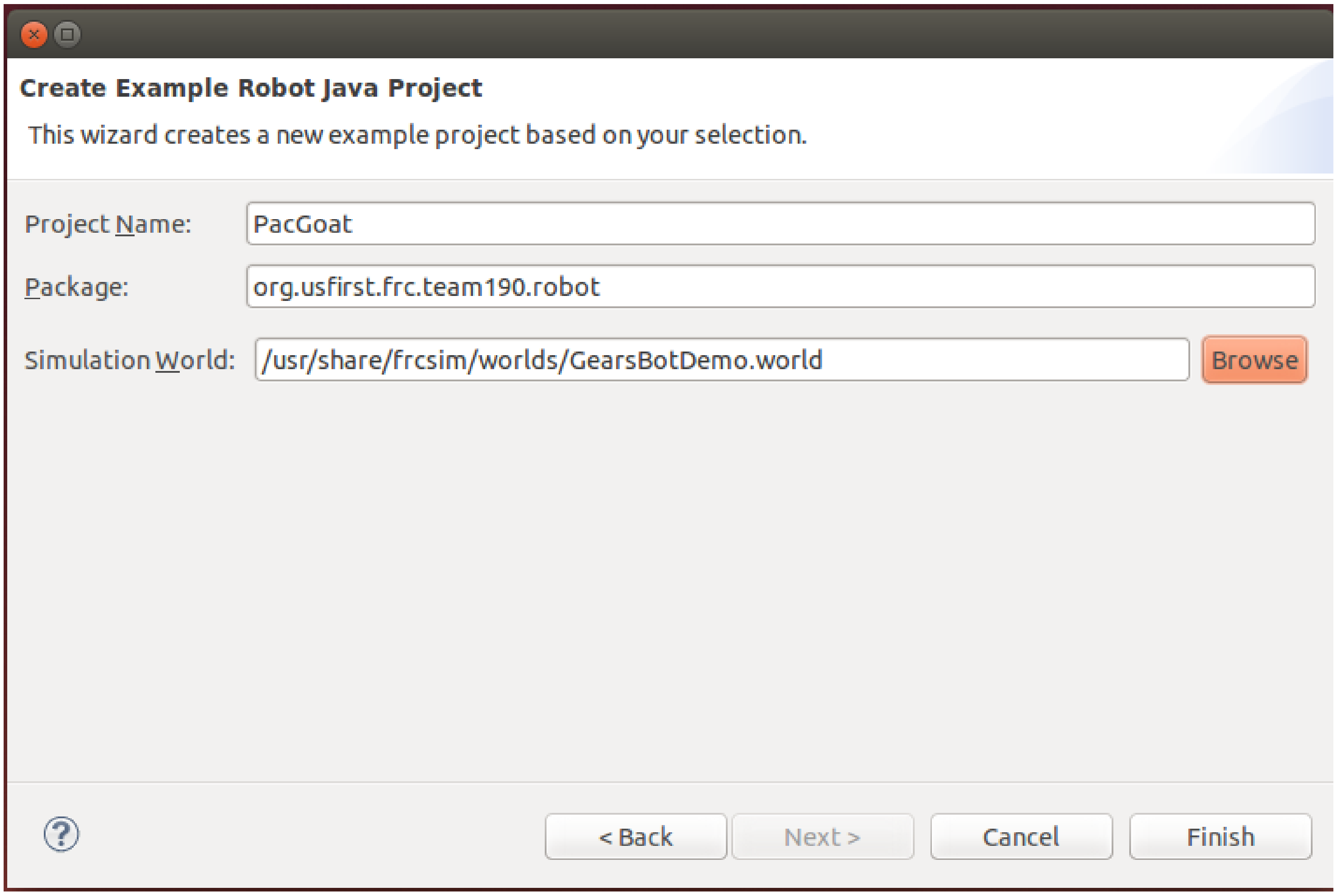
PacGoat

A fully functional example CommandBased program for FRC Team 190's 2014 robot. This code can run on your computer if it supports simulation.

1. Select the PacGoat example

2. Click Next

Naming the project


Create Example Robot Java Project

This wizard creates a new example project based on your selection.

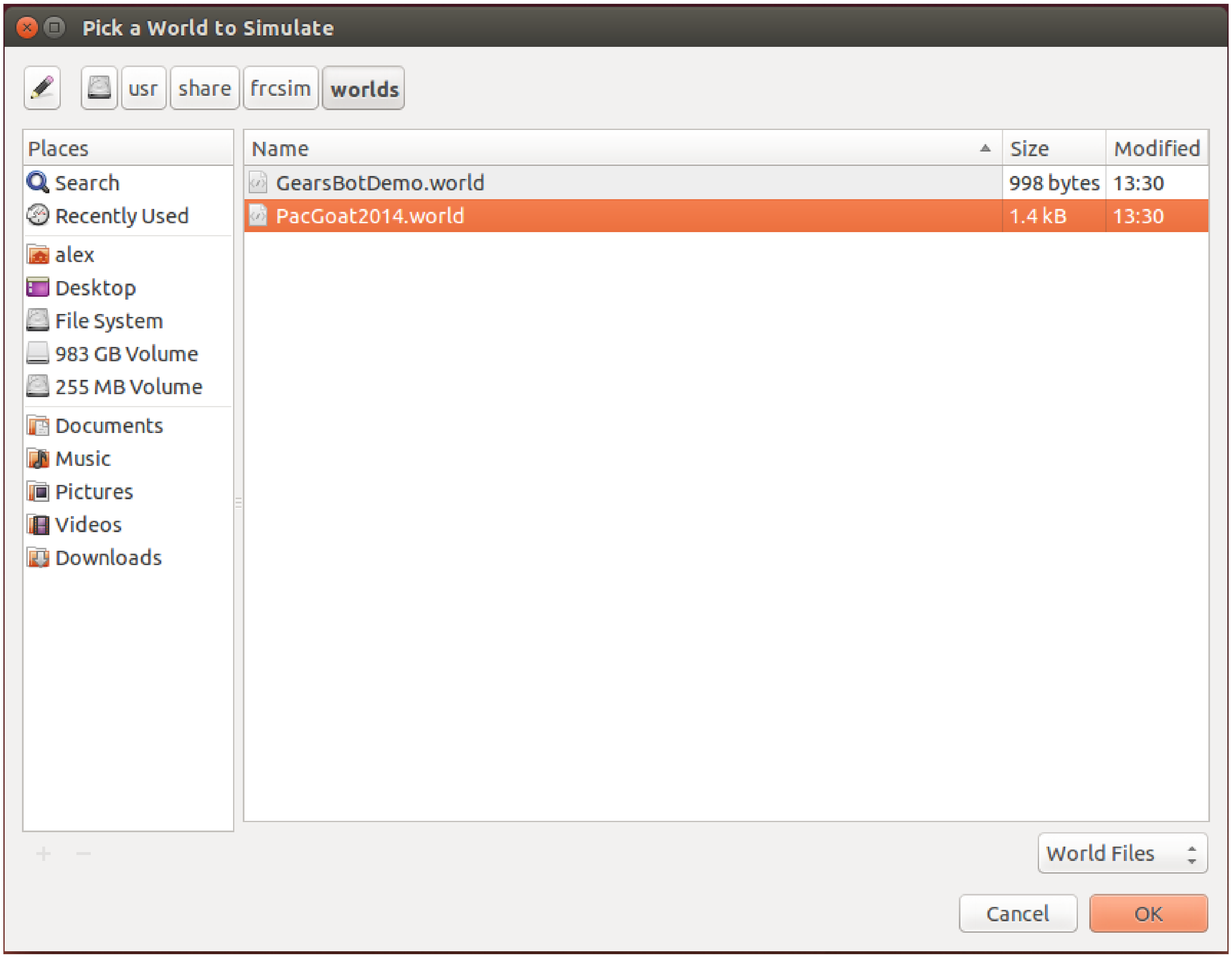
Project Name: PacGoat

Package: org.usfirst.frc.team190.robot

Simulation World: /usr/share/frcsim/worlds/GearsBotDemo.world

Type in a project name such as “PacGoat”

Select the world



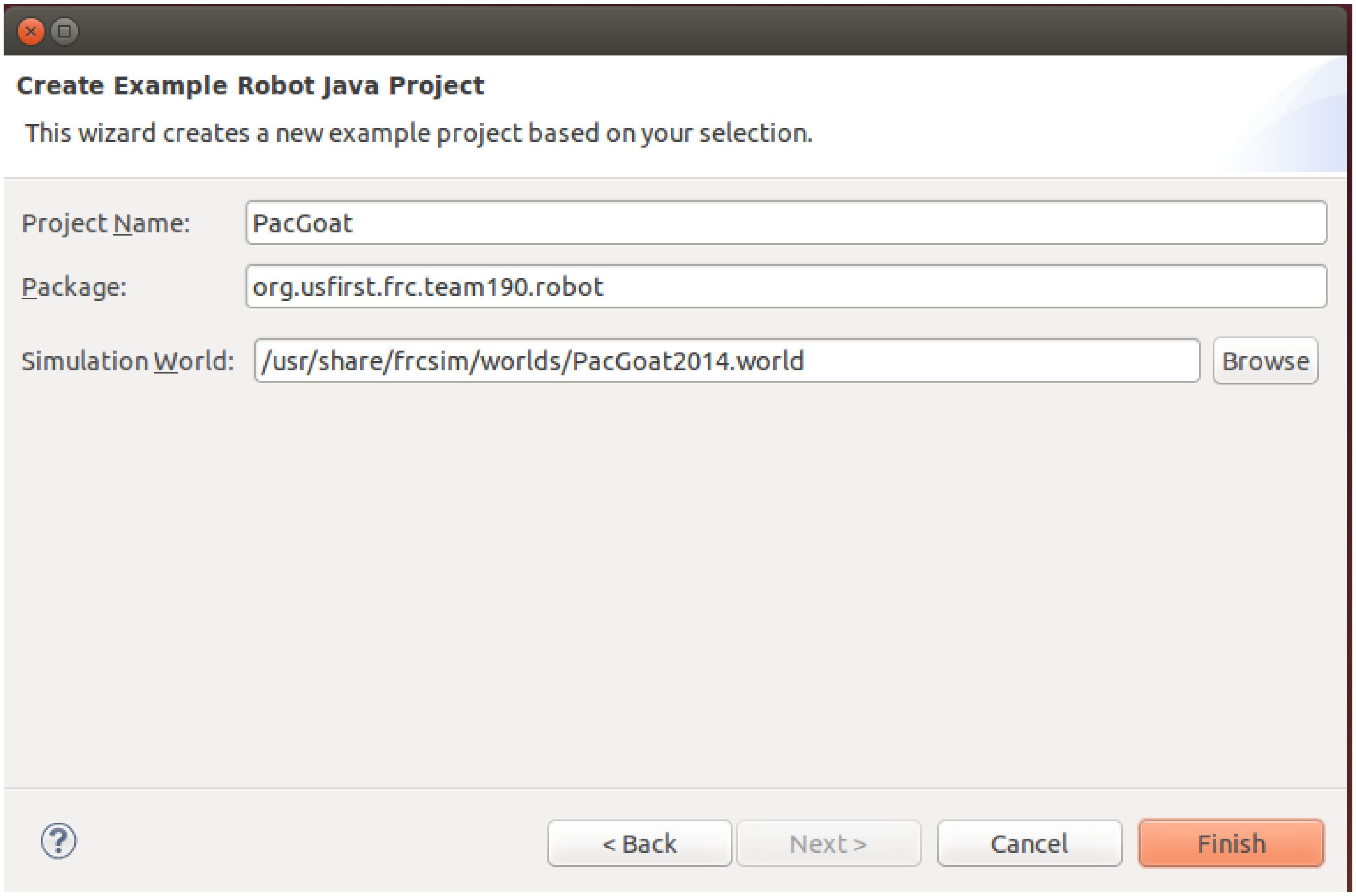
Name	Size	Modified
GearsBotDemo.world	998 bytes	13:30
PacGoat2014.world	1.4 kB	13:30

1. Click “Browse”

2. Select “PacGoat2014.world”

3. Click “OK”

Finish


Create Example Robot Java Project

This wizard creates a new example project based on your selection.

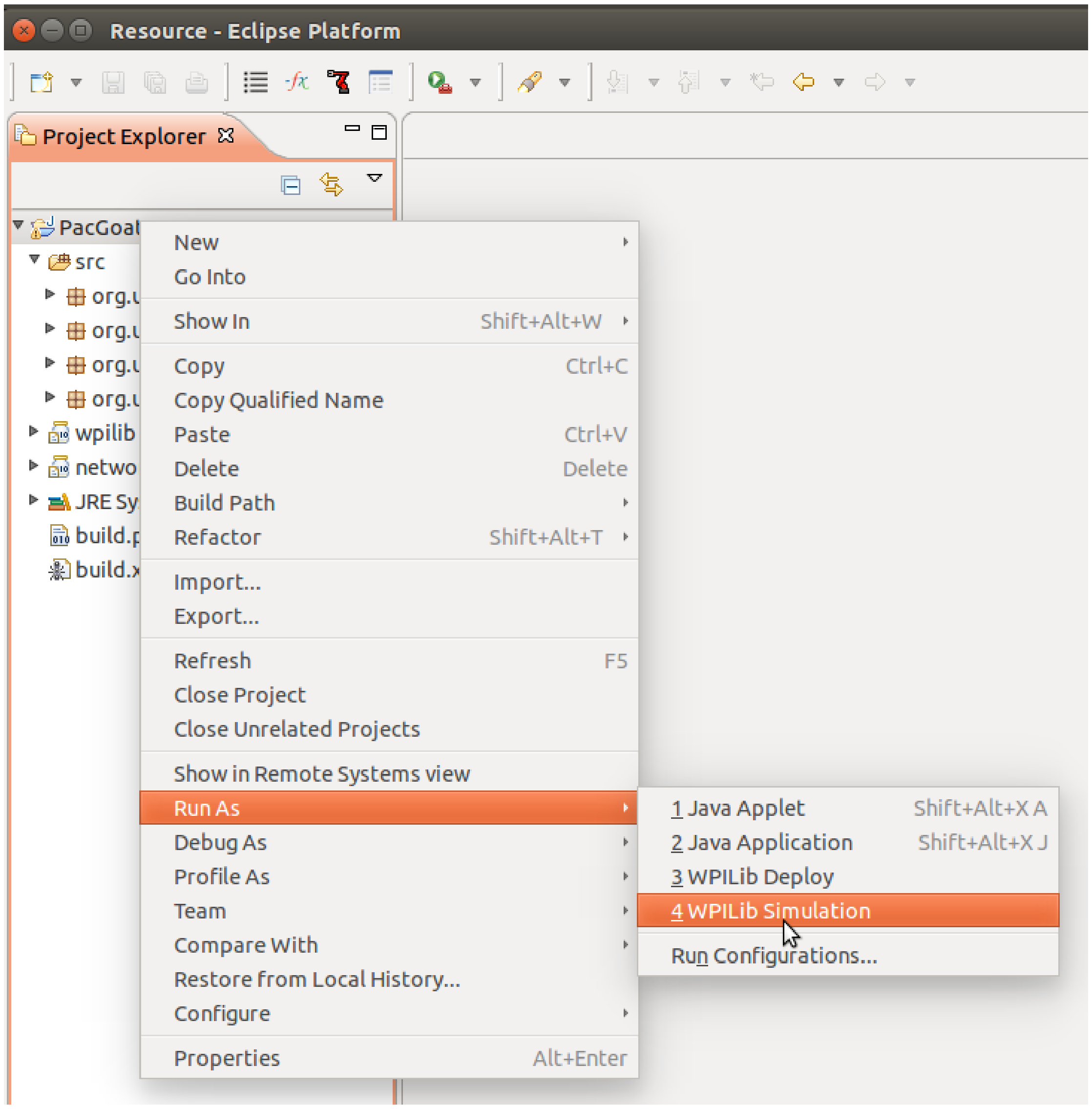
Project Name: PacGoat

Package: org.usfirst.frc.team190.robot

Simulation World: /usr/share/frcsim/worlds/PacGoat2014.world

Press Finish

Running the simulator

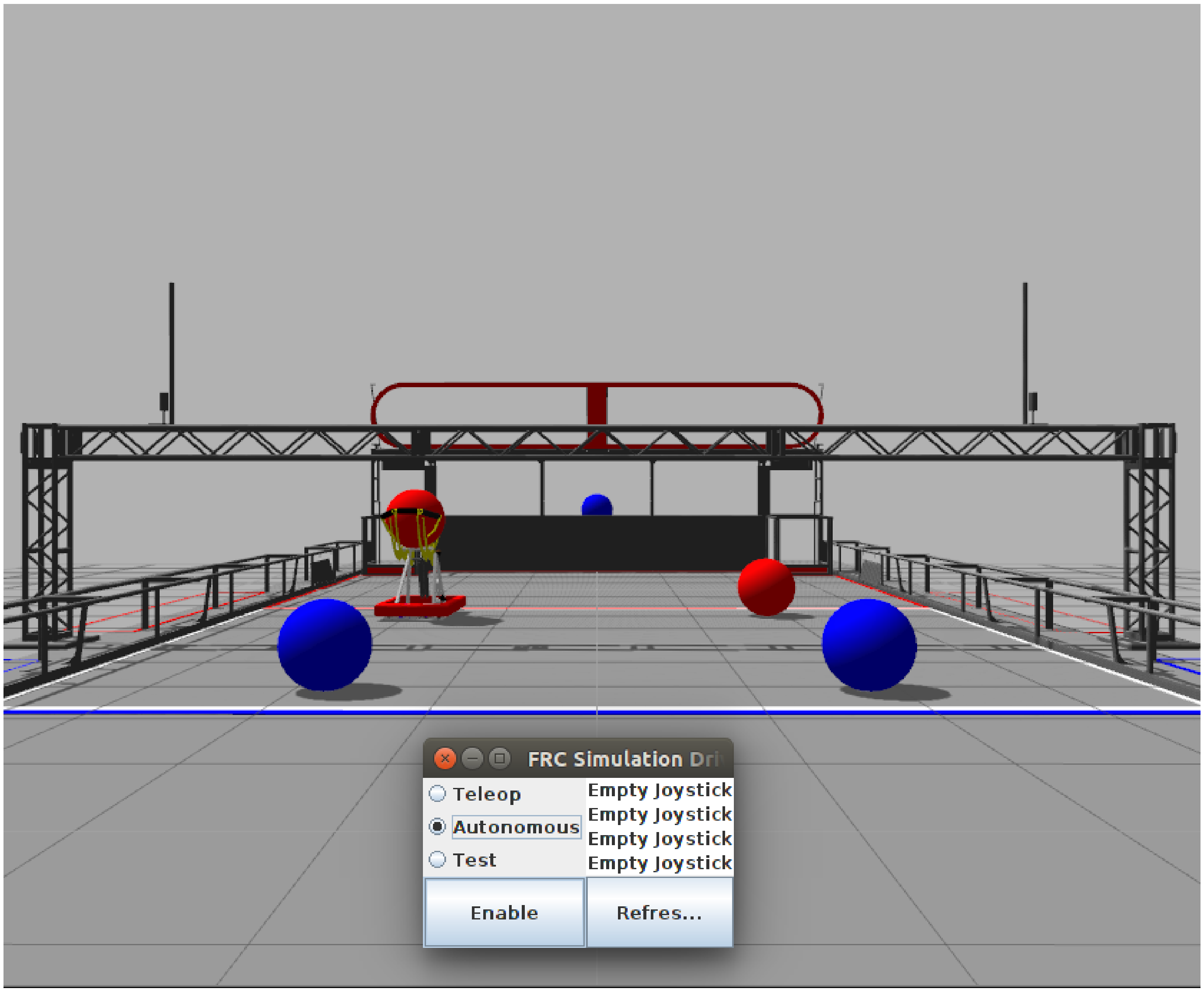


1. Right click on the project

2. Select Run As

3. Click WPILib Simulation

Running autonomous

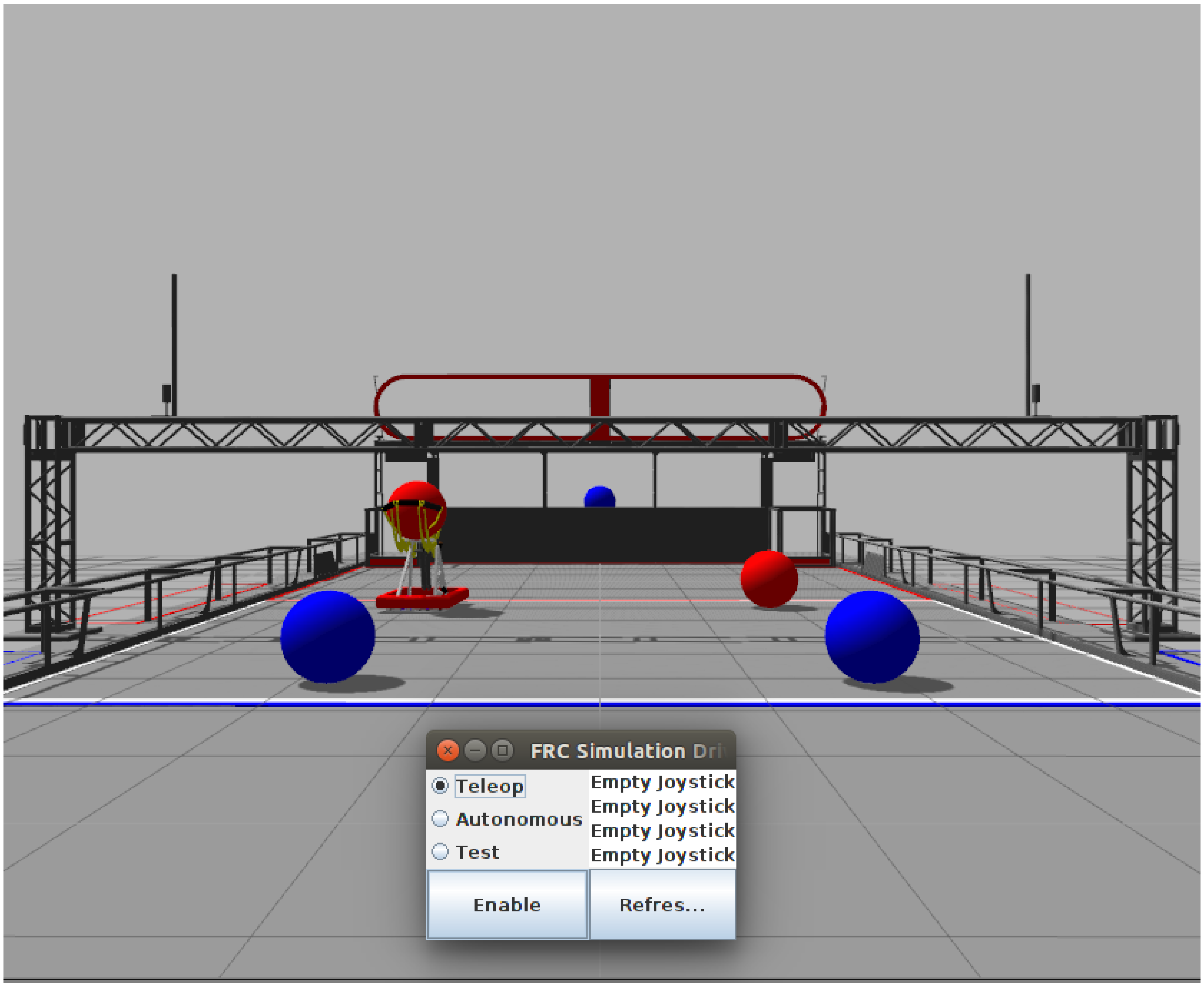


1. Make sure the simulator is running

2. Select Autonomous

3. Click Enable

Using a PS3 joystick to drive



1. Plugin a joystick (preferably a PS3 SIX AXIS/DUALSHOCK)

2. Start the simulator

3. Select Teleop

4. Click Enable

5. 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
- Drive in a square
- Arrange the sodas in a square • Whatever else you can think of

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

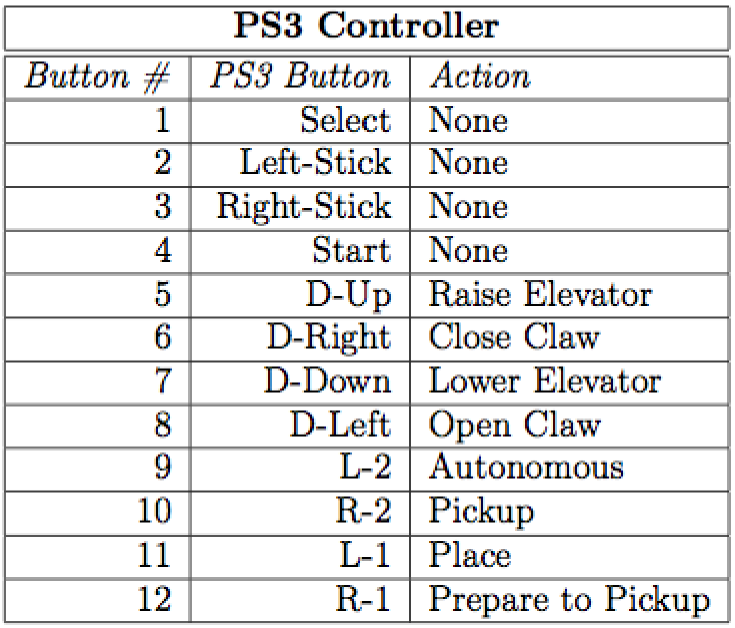
21converting the readings from the real robot to these units the code can be written at a higher level that runs in both simulation and on the real robot. The second major difference is that the PID values are different on the real robot and on the simulated robot. While it would be ideal if they could 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

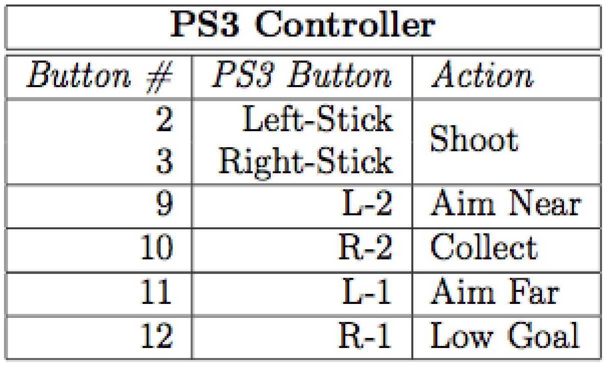
Digital		
Pin	*Subsystem*	*Sensor*
1	Drive Train	Left Encoder $_{B}^{A}$
2		
3		Right Encoder $_{B}^{A}$
4		

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.

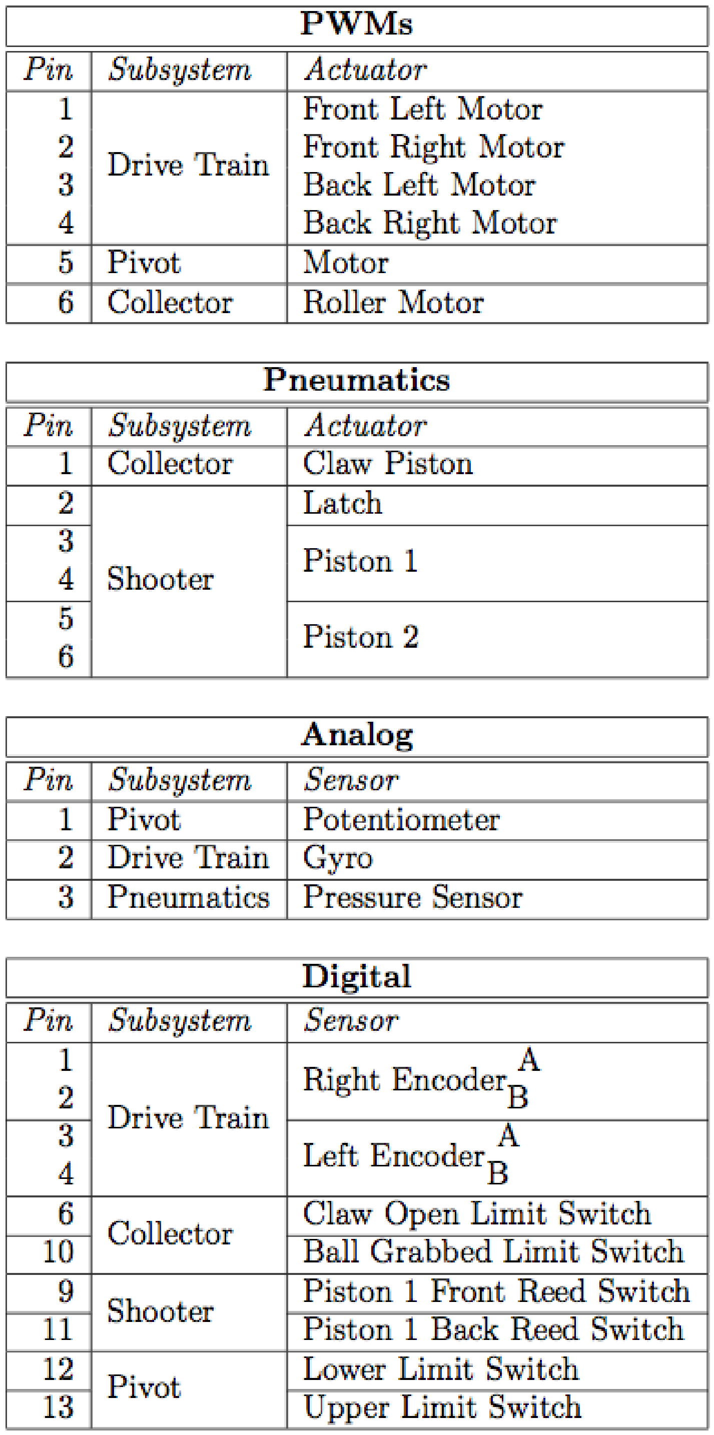
Operator interface



PS3 Controller		
Button #	*PS3 Button*	*Action*
2	Left-Stick	Shoot
3	Right-Stick	
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 4		Piston 1
5 6		Piston 2

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

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