

---

# **SnobotSim**

***Release 0.0.0***

**Jan 13, 2020**



<b>1</b>	<b>Getting started</b>	<b>3</b>
<b>2</b>	<b>VS Code Extension</b>	<b>5</b>
<b>3</b>	<b>Standalone Setup Tool</b>	<b>7</b>
<b>4</b>	<b>Gradle Plugin</b>	<b>9</b>
<b>5</b>	<b>Configurating the Simulator</b>	<b>13</b>
<b>6</b>	<b>Custom Simulators</b>	<b>21</b>
<b>7</b>	<b>Motor Model Simulations</b>	<b>23</b>
<b>8</b>	<b>C++ Setup Guide</b>	<b>27</b>
<b>9</b>	<b>Release Notes</b>	<b>29</b>
<b>10</b>	<b>Indices and tables</b>	<b>33</b>



SnobotSim is a simulator for FIRST Robotics Competition (FRC) Robots that use Java.

Please report any simulator related issues to the [main project](#)



### 1.1 Introduction

SnobotSim is a easy to use, customizable, and extendable extension of the WPILib simulation framework. It was designed to require no changes to your robot code, and only minor changes to your build script.

The goal of the simulator is to provide you with a mechanism to visualize your robot components, and run, test, and debug your code. It supports rudimentary physics simulations, and simulations of third party libraries like CTRE, REV Robotics, and NavX sensors. You can simulate joysticks, simulate your autonomous code, and even create unit tests without the complex set up of tools like Mockito.

The simulator is not intended to be a 100% accurate real world simulation, so don't plan on tuning your mechanisms or your path planning with it. If set up properly, it should get you pretty close, but you should always be sure to test your code on your real hardware.

### 1.2 Quickstart Guide

Several tools have been developed to make the setup process as easy and painless as possible. The initial setup is detailed further along in the documentation, but the process is essentially:

1. Install the VS Code extension
2. Run the setup program
3. Run the simulator
4. Customize your robot





Note: This tool is in development and may not function as expected

This is a VS Code Extension that allows you to easily add SnobotSim to your project, as well as simplify the process for upgrading versions and running the simulator from your IDE. If you are not a VS Code user but still want the setup tool, you can use the *standalone tool* instead.

## 2.1 Installing the Extension

The extension is not available on the marketplace yet, so you must install it using a VSIX package. The releases of the extension are stored on the [projects Github page](#).

To install:

1. Open VS Code
2. Select and download the `vscode-RE.LEA.SE.vsix` bundle
3. In VS Code, press `CTRL+SHIFT+P` to open the command palette, and type “Install From VSIX”
4. Select Extensions: Install from VSIX...
5. Navigate to where you downloaded the SnobotSim VSIX package, and select it

## 2.2 Using the Extension

Once the extension has been installed, there should now be some SnobotSim commands available.

To set up your project for the first time, you can right click your `build.gradle` file, and select “Setup Snobot Sim”. This will open the window that will allow you to customize your installation



---

### Standalone Setup Tool

---

Note: This tool is in development and may not work as expected

This is a standalone tool that can add SnobotSim to your existing robot project. It will add the necessary configuration items into your `build.gradle` file, install the necessary config files that the simulator uses to get you up and running as easily as possible.

This is a standalone version of the VS Code plugin outlined in [VS Code Extension](#) portion of the documentation. If your team uses VS Code, it might be better to use the extension rather than the standalone tool

### 3.1 Downloading the tool

The releases are kept on the [projects Github page](#). You should download the latest version that is applicable for your operating system.



The SnobotSimPlugin is a gradle plugin designed to make the simulator easier to use. It provides a easy to use mechanism to pull in all of the dependencies, and provide tasks to start the simulator from the command line.

### 4.1 Manual Installation Notes

Note: You can use the installation tool outlined in *VS Code Extension* to avoid doing the manual installation

To add the plugin to your robot project, update the plugin block of your `build.gradle` file like so

```
plugins {  
    id "java"  
    id "edu.wpi.first.GradleRIO" version "2020.1.2"  
    id "com.snobot.simulator.plugin.SnobotSimulatorPlugin" version "2020-0.0.0" apply_↵  
    ↵false  
}
```

Notice the `apply false` line maker. This means that we are not activating the plugin yet, because we need more information from the rest of the build script first.

The recommend place to add the activation is between your `repositories` block and your `dependencies` block, like so:

```
// Maven central needed for JUnit  
repositories {  
    mavenCentral()  
}  
  
////////////////////////////////////  
// SnobotSim  
////////////////////////////////////  
apply plugin: com.snobot.simulator.plugin.SnobotSimulatorPlugin  
apply from: "snobotsim/snobot_sim.gradle"
```

(continues on next page)

(continued from previous page)

```
////////////////////////////////////  
  
// Defining my dependencies. In this case, WPILib (+ friends), and vendor libraries.  
// Also defines JUnit 4.  
dependencies {  
    compile wpi.deps.wpilib()  
    compile wpi.deps.vendor.java()  
    nativeZip wpi.deps.vendor.jni(wpi.platforms.roborio)  
    nativeDesktopZip wpi.deps.vendor.jni(wpi.platforms.desktop)  
    testCompile 'junit:junit:4.12'  
}
```

## 4.2 Running the simulator

The easiest way to run the simulator is from the command line. Open a command window in your root directory (the directory that has the main `build.gradle` script), and run the following command, `./gradlew runJavaSnobotSim` and you should see the the simulator GUI open on your computer

Notes:

- Windows users don't need to put the `./` in front of `gradlew`, only Linux users do
- C++ teams should use `./gradlew runCppSnobotSim`. For more specific C++ notes and exceptions, see [C++ Setup Guide](#)

## 4.3 Release Notes

### 4.3.1 Current Release

#### 2020-0.0.0

- 2020 Kickoff Release
- Includes a new task, `updateSnobotSimConfig`, which will download the latest SnobotSim config file

### 4.3.2 Previous Releases

#### 2019-2.0.0

- Added ability to simulator REV SparkMax motor controllers

#### 2019-1.0.0

- Updates to make the build script easier to set up and use

#### 2019-0.2.0

- Updates required for the 2019.1.1 WPLib release

## **2019-0.0.0**

- Initial beta release for the 2019 season





---

### Configuring the Simulator

---

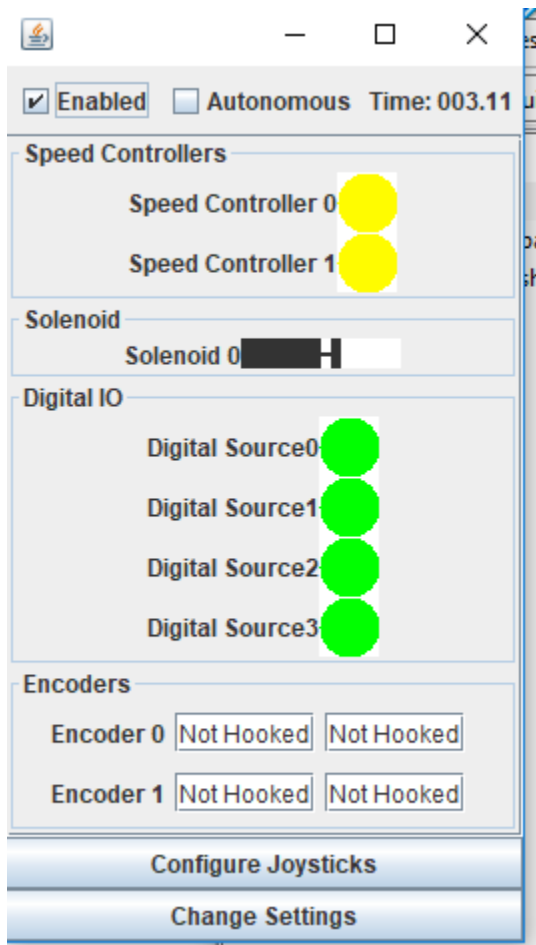
Once you have the added necessary simulator files to your project, you are ready to run it and customize it for your teams robot.

If you have installed the simulator on top of a brand new repository, you won't see much going on and won't have any options to configure. Alternatively, if you are running for the first time with an already completed robot, you might see a large number of sensors and actuators that you can set up.

For this portion of the documentation, we will be using the "Example Robot" that comes packaged with the tool.

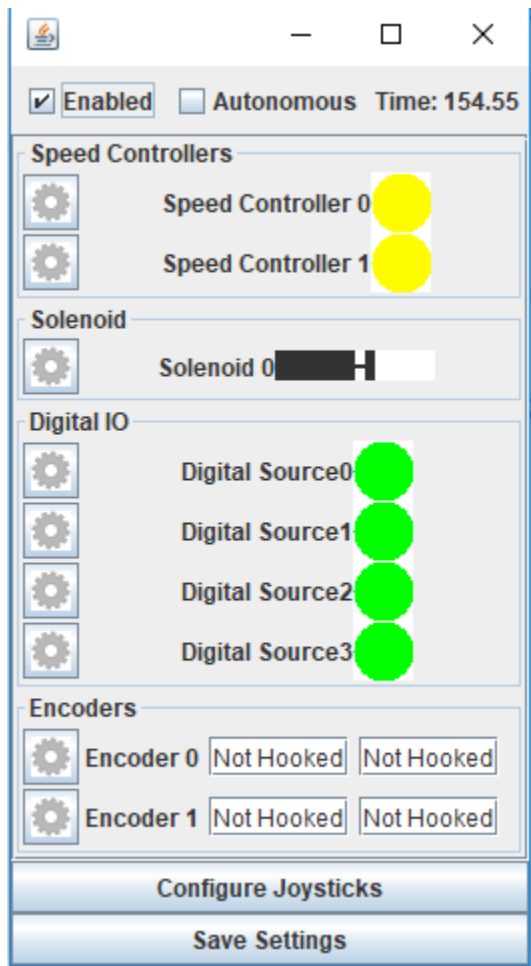
#### 5.1 Simple Customization

The first time that you run the simulator with the example robot, you should see a screen like this

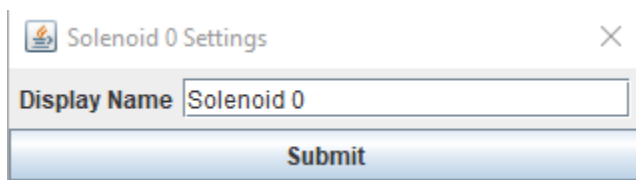


You can now change your robots state into autonomous or disabled modes, see the elased “Match Time”, and configure the components settings.


If you select the “Change Settings” button, you will be open to changing the settings for all of the loaded components. Simply click on the gear next to the component to see what options are available.



Most of the settings are very simplistic; you can only set a custom display name for the component. Here is an example of the dialog that allows you to set a new name for a solenoid



There are several options for how you can configure the connected motor settings for a speed controller, outlined *Motor Model Simulations*

 Speed Controller 0 Settings ✕

Display Name

Simulation Type Static Load ▼

---

Simulator Parameters

Load

---

Motor Information

Motor Type CIM ▼

Transmission

Num Motors  ▲ ▼

Gear Ratio

Efficiency

Motor Parameters

Inverted ☐

Brake ☐

Nominal Voltage

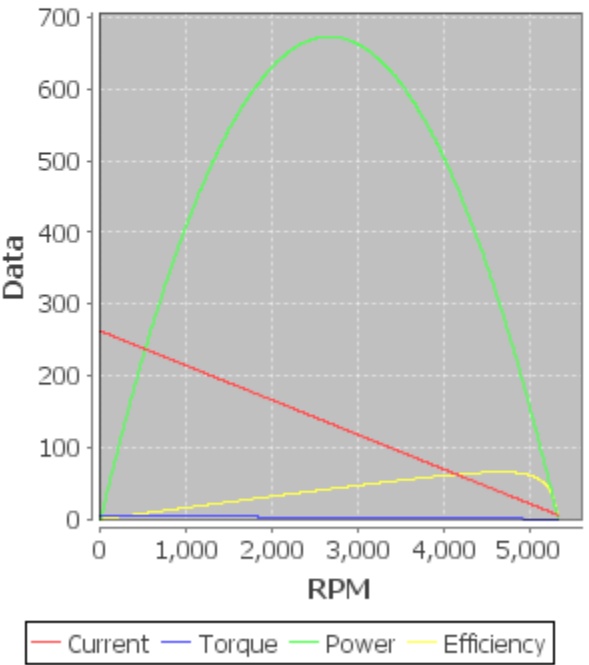
Free Speed RPM

Free Current

Stall Torque

Stall Current


**CIM**



— Current — Torque — Power — Efficiency

**Submit**

The encoder settings allow you to hook up an encoder to a speed controller as a feedback device. This is very useful for simulating things like your drivetrain/elevator/arm/etc

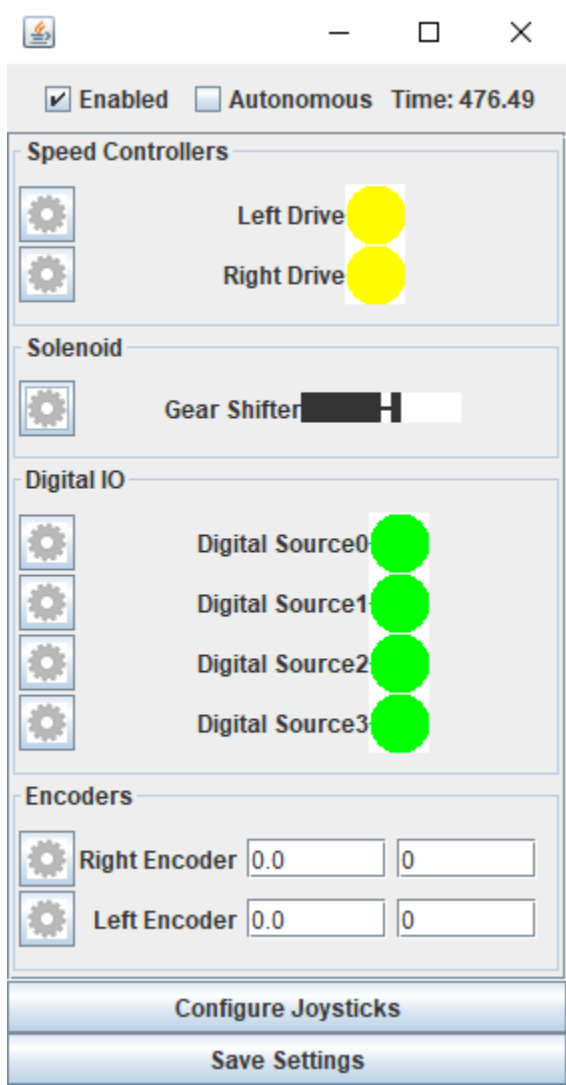
 Encoder 0 Settings ✕

Display Name

Connected SC Speed Controller 0(0) ▼

**Submit**

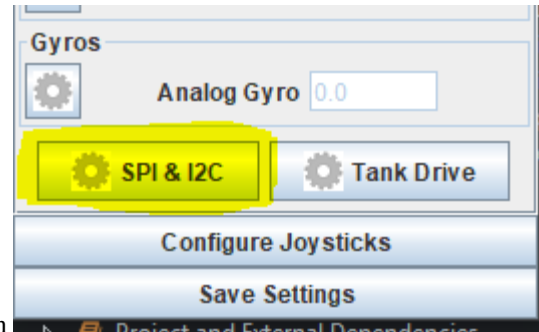
Once you have renamed the components and hooked everything up as you would like, you should see something like this. Not that the encoders now display count and distance rather than “Not Hooked Up”



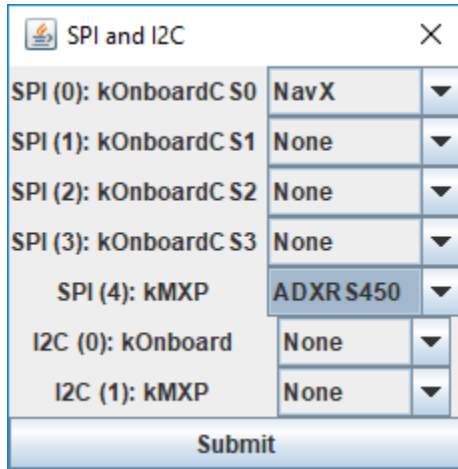
Once you are all set, you can click the “Save Settings” button to persist the changes to disk, and leave edit mode. Now the next time you start it up, you should see the new names you gave the components

## 5.2 I2C and SPI simulators

Due to the highly de-coupled nature of SnobotSim, the simulator cannot determine what sort of device you have connected when you create an I2C or SPI device, so it must be set up manually. Note, in almost all cases this information must be known on startup, so after setting up the simulator you must save your changes and restart the simulator.



To configure a I2C or SPI device, click on the *SPI & I2C* button



Currently the simulator natively supports the following devices:


- ADXL345 Accelerometer (Comes with WPILib, either I2C or SPI)
- ADXL362 Accelerometer (Comes with WPILib, SPI only)
- ADXRS450 Gyroscope (Comes with WPILib, SPI only)
- NavX (KauaiLabs simulator, in the I2C or SPI configurations, no USB)

You must find the port you need, and set the appropriate simulator. Note: You are probably setting up the device with WPILib's enumerations, but if you hover over the constant, IntelliSense will tell you what internal ID it represents. For example, SPI.kMXP is port 4

If you have a custom SPI/I2C device or are using something the simulator doesn't currently support, you can add your own simulator. This will be discussed later in the documentation

## 5.3 Tank Drive Simulator

SnobotSim provides a very simplistic tank drive simulator, assuming that you are running with an independant skid steer robot, with a gyro measuring yaw feedback. Note: the current implementation has very little science behind it, and is mostly used for very rough testing purposes.

 Tank Drive Settings
 ×

Add Simulator

Left Motor	Speed Controller 0(0)	▼
Right Motor	Speed Controller 1(1)	▼
Gyro	Analog Gyro(0)	▼
Turn Gain	33.33	

Remove

Submit changes





---

## Custom Simulators

---

There are several times when you might want to add a custom simulator to your robot, such as

- Simulating limit switches being triggered when your elevator encoder reaches a certain height
- Simulate non-supported SPI/I2C devices like a PixyCam or Dotstar LED strips
- Simulate your camera ICD to fake data based on your robot position (i.e. faking the Limelight NetworkTable or the NetworkTable/socket messages coming from a co-processor)
- Simulating delayed sensor responses (i.e. When I press my “intake ball” button, it takes .25 seconds before my sensor gets tripped)

### 6.1 Examples

Some of these are outdated and are using the old SnobotSim API, but the logic behind them should still hold

#### 6.1.1 Camera Simulators

- [STEAMworks UDP Camera Simulator](#)
- [DeepSpace Limelight Simulator](#)

#### 6.1.2 Sensor Feedback Simulators

- [Potentiometer Simulator](#)
- [NavX Pitch Simulator](#)

#### 6.1.3 SPI/I2C

- [Dotstar LED simulator](#)



---

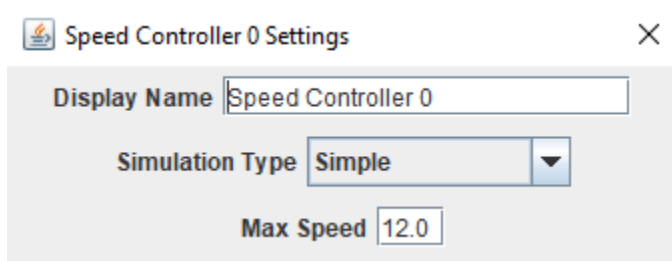
## Motor Model Simulations

---

There are a handful of pre-canned motor model simulations. They are not intended to be real-world accurate, but can do a decent enough job to simulate things such as your drivetrain, or an arm that rotates and has to fight gravity.

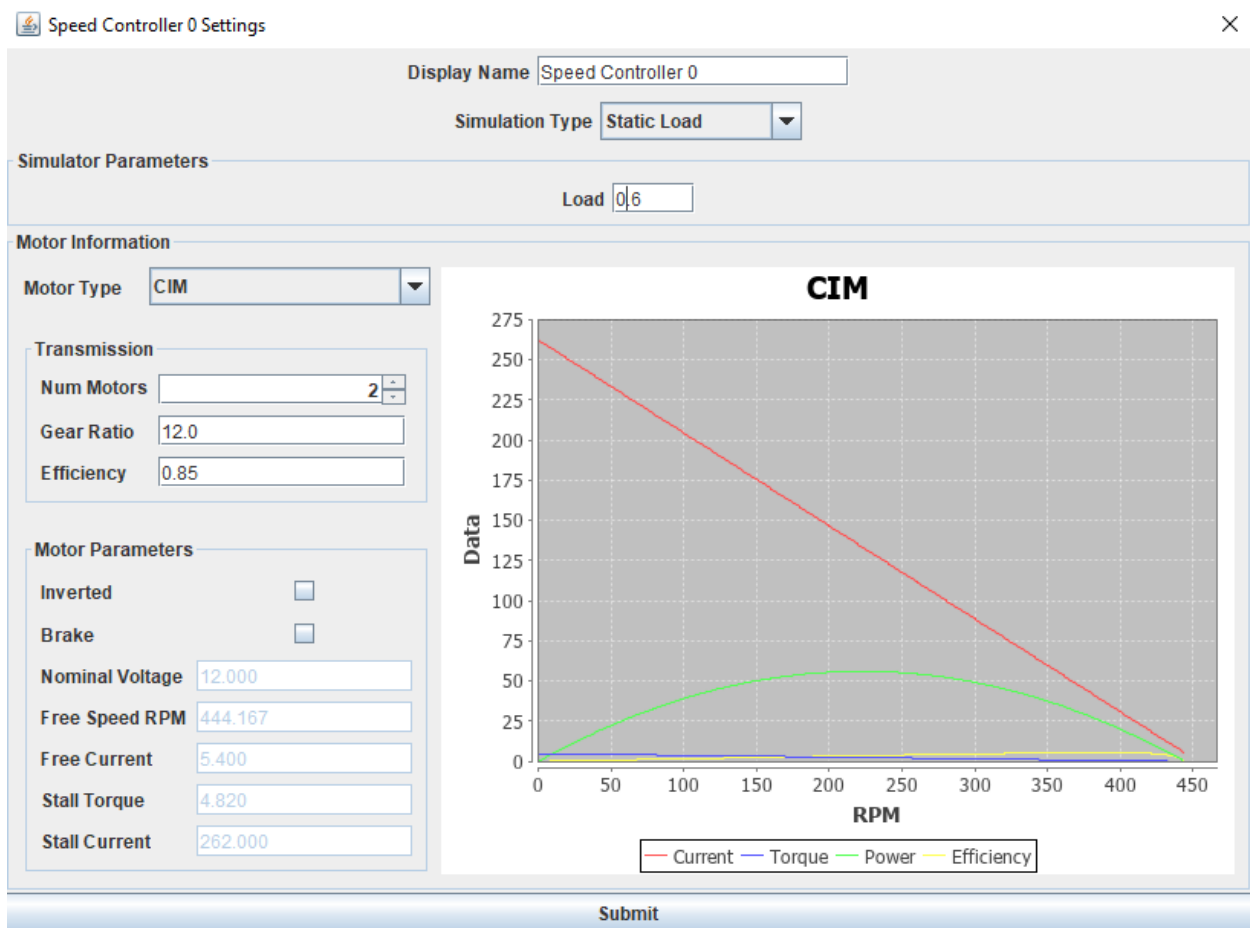
### 7.1 Linear Simulator

This is the simplest simulator type available. It assumes instantaneous acceleration such that  $motor\_speed = (voltage / 12) * MAX\_SPEED$ , where max speed is a configurable value in the units of your choice




### 7.2 Static Load Simulator

This motor model simulates a motor with some applied load. It does not account for additional forces like springs or gravity, and can be used to simulate horizontal linear actuators, or even your drive train motors.



## 7.3 Gravity Load Simulator

This model simulates something that has to fight gravity in one direction, like an elevator. It is not meant to be used with an arm where the load applied by gravity varies by the angle of the arm, only constant forces.

 Speed Controller 0 Settings ✕

Display Name

Simulation Type

Simulator Parameters

Load

Motor Information

Motor Type

Transmission

Num Motors

Gear Ratio

Efficiency

Motor Parameters

Inverted ☐

Brake ☐

Nominal Voltage

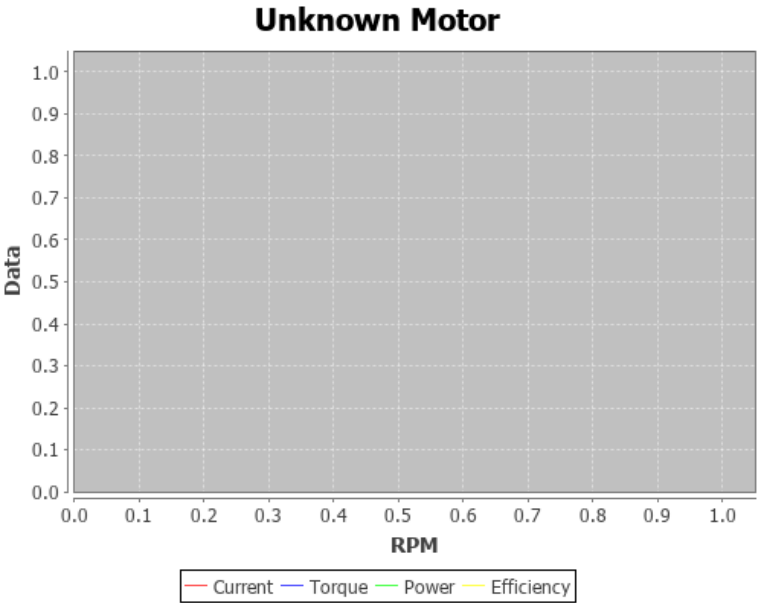
Free Speed RPM

Free Current

Stall Torque

Stall Current

**Unknown Motor**

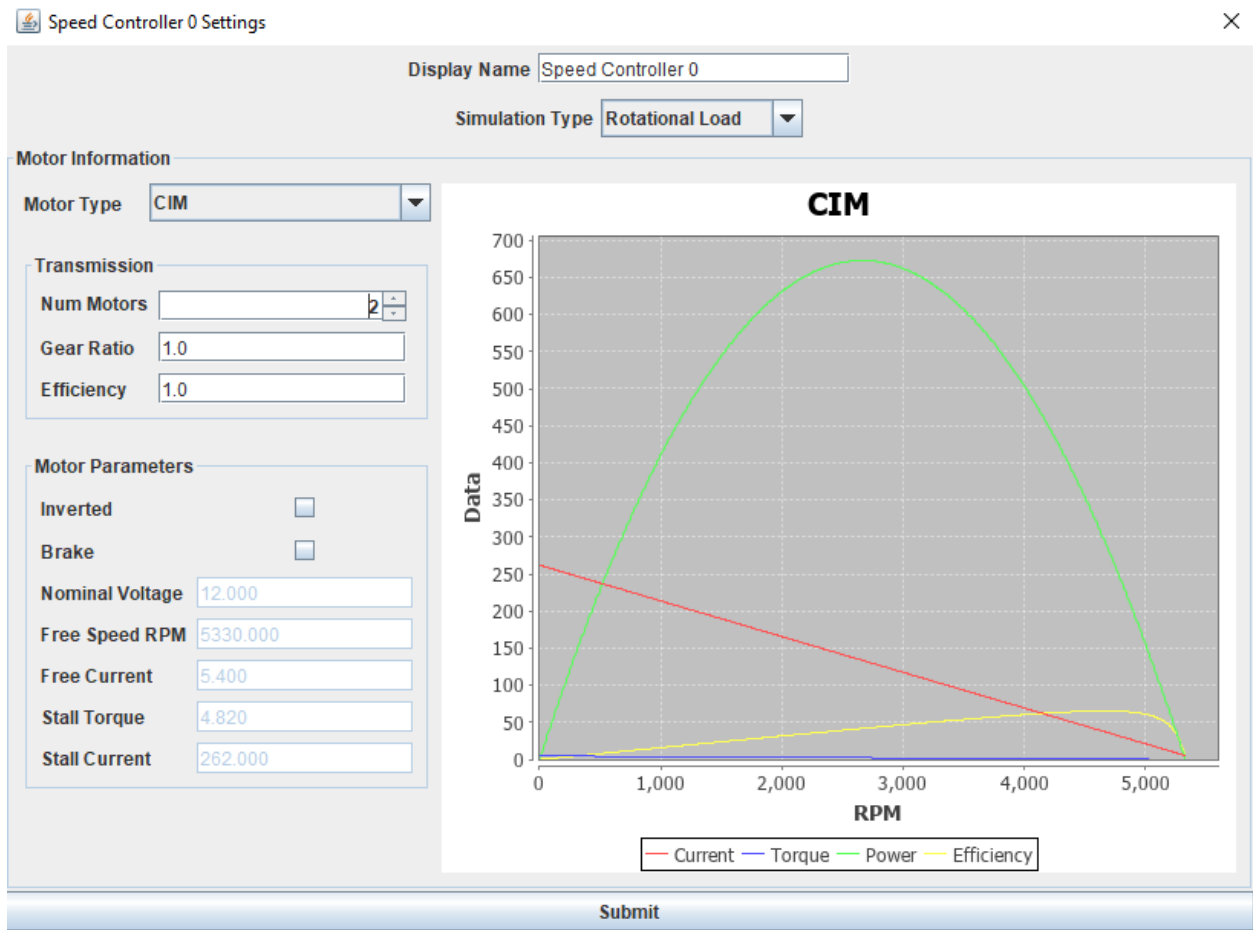


Submit

## 7.4 Rotational Load Simulator

This model simulates a rotating arm, where gravity affects the load through its travel.

Note: THIS IS NOT FULLY SUPPORTED



## CHAPTER 8

---

### C++ Setup Guide

---

// TODO





## 9.1 Current Release

### 9.1.1 2020-0.0.1

Kickoff release

- Added super quick and dirty simulator for the Spark Max. Basically just does voltage percentage simulation, but they will now show up on the GUI. Also supports follow mode, and following a TalonSRX
- Requires [2020-0.0.0 Plugin](#) to use downstream

#### **Known Issues**

- No C++ simulations
- Standard warning about potentially wrong / missing features in CTRE simulation
- REV completely changed their API. Most of the “common” features should be supported, but your mileage may vary

## 9.2 Previous Releases

### 9.2.1 2019-1.0.0

Added simulation for REV CANSparkMax

- Added super quick and dirty simulator for the Spark Max. Basically just does voltage percentage simulation, but they will now show up on the GUI. Also supports follow mode, and following a TalonSRX
- Requires [2019-2.0.0 Plugin](#) to use downstream

#### **Known Issues**

- C++ simulations with a gui broke during GradleRIO alpha testing. C++ teams, I know I've said it for a year, but you should be able to use this before seasons end
- CTRE simulation is hot off the presses. Unit tests work, but new features and functions are probably not supported.
- No smart features of the REV speed controller are supported.

### 9.2.2 2019-0.0.1

First release of the 2019 build season

- Updated to work with wpilib 2019.1.1
- Updated to work with CTRE-Phoenix 5.12.0
- Updated [examples](#) to work with the kickoff release of VsCode/wpilib/CTRE/NavX
- Requires [2019-0.2.0 Plugin](#) to use downstream

#### Known Issues

- C++ simulations with a gui broke during GradleRIO alpha testing. C++ teams, I know I've said it for a year, but you should be able to use this before seasons end
- CTRE simulation is hot off the presses. Unit tests work, but new features and functions are probably not supported.

### 9.2.3 2019-0.0.0

*2019 Beta Version*

- Updated to work with wpilib 2019
- Added some [example](#) on how to use the simulator. Note: these are a work in progress.

*Important notes*

- Requires [2019-0.0.0 Plugin](#) to use downstream
- wpilib is still in beta as well. This means that there might be issues in their software which can cause confusion when using the simulator. Also, since the real “full” beta hasn't been released, I don't think you will be able to run this version of wpilib on a real robot. This is meant for simulation purposes only.
- 3rd party libraries (like CTRE and NavX) have not released their 2019 libraries. There might be a lot of things that have to change to work with those once they get released.

### 9.2.4 2018-2.0.0

- Modified the native library loading scheme to help support [HAL Extensions](#). Also means that mac builds now work
- Added command line option to disable the driver station simulation. \* Opens the door to being able to use an extension to talk to the real driver station instead. This extension is in the works and should be part of the 2019 wpilib release
- Note: Requires the [2018-2.0.0 Plugin](#) to use downstream

#### Known Issues

**CTRE**

- Voltage and AppliedThrottle control modes are in beta support. This means that that they *should* work as expected
- MotionMagic, Position, and Velocity control modes are in alpha support. This means that they *may, sort of, sometimes* work as expected. They are tested against, but may not anywhere near the same as the real world
- Everything else is unsupported. Attempting to use other functions will simply log a warning. For the average team, these features will not hinder the majority of your pre-robot testing

## 9.2.5 2018-1.0.1

- Changed to Year-Sem.Ver.2 versioning system
- Removed the eclipse boilerplate. All Hail GradRIO.
- Moved internal sources around to fit into the standard gradle format (src/main/java vs src)
- Externalized CTRE simulator core code. This should make it easier to pull in different version of the library, and avoid unsatisfied link errors
- Updated the [SnobotSimulatorPlugin](#) to work with the code refactoring. To use this release, you must use a plugin version of 2018-1.0.0 \* This plugin upgrade allows you to run the simulator from the command line with `gradlew runSnobotSim`
- Updated the configuration file format to know more about the simulator objects. Should make it easier to create custom overrides. \* There is a utility that will load the “v0” config file, but it will pop up with a warning. It is recommended that you re-save your config to force it into the “v1” state
- Simulator components get loaded on startup, rather than dynamically as the robot starts. \* Note: You might get a popup warning now when you start the simulator. It loads things first, then checks to make sure they were initialized. If there is a mismatch, expected things might not show up
- Started preparation to migrate to WPI’s 2019 library (while maintaining 2018 support)

### Known Issues

#### CTRE

- Voltage and AppliedThrottle control modes are in beta support. This means that that they *should* work as expected
- MotionMagic, Position, and Velocity control modes are in alpha support. This means that they *may, sort of, sometimes* work as expected. They are tested against, but may not anywhere near the same as the real world
- Everything else is unsupported. Attempting to use other functions will simply log a warning. For the average team, these features will not hinder the majority of your pre-robot testing

## 9.2.6 0.8.0

---

**Note:** Improvements also include 7.1, 7.2, 7.3 because I forgot

---

- Fixed issues with match time and the frequency loops were being called
- Added Linux version of the Eclipse Boilerplate. Due to compilation flags being different on build machines vs. your machine, your mileage might vary
- CTRE 5.2.1.1

- Added static analysis tools to the build process (PMD, checkstyle, fixbugs)

## **9.2.7 0.7.0**

First release of the 2018 Season. Anything releases before this were experimental and not recommended for usage.

### **Known Issues**

#### **CTRE**

- Voltage and AppliedThrottle control modes are in beta support. This means that that they *should* work as expected
- MotionMagic, Position, and Velocity control modes are in alpha support. This means that they *may, sort of, sometimes* work as expected. They are tested against, but may not anywhere near the same as the real world
- Everything else is unsupported. Attempting to use other functions will simply log a warning. For the average team, these features will not hinder the majority of your pre-robot testing

#### **Simulator**

- Tank drive simulator, I2C, and SPI built in simulations do not have GUI support, so you must update the config file manually

## CHAPTER 10

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`