

Robot Software

- [Getting Started with Robot Development](#)
 - [What you need to install](#)
 - [Learning Robot Code](#)
 - [Resources](#)
 - [Getting started with a Romi](#)
- [Configuring a Robot](#)
 - [Setting steer offsets for a Mk 4i drivetrain](#)
 - [Setting the magnet offset for a CANcoder](#)
 - [Flashing an Radio \(pre-2024 Champs Radios\)](#)
 - [Testing Robot Code](#)
- [Git 101](#)

Getting Started with Robot Development

This chapter details what software you need to install to write software for our robots provides starting points for learning on your own.

What you need to install

Git

Git is a popular version control system (VCS) for developing software. If you are interested in becoming a Software Engineer or writing code in general, learning to use git is an extremely important skill. You can [download git here](#). When installing, just choose all the default recommended options.

We use GitHub to host and collaborate on our software projects, so you will need to make a GitHub account on github.com. Once you have an account, post your GitHub username in the #controls channel of our discord and we can add you to our GitHub organization which you can find [here](#) along with every software project our team has ever worked on.

WPILib

WPILib is the suite of software, known as a library that we use to actually control the components on the robots. [WPILib has their own guide](#) for setting up your development environment, however since their guide is designed for everybody across FRC, it has a lot of information not relevant to our team. Following the guide below should get you set up for developing robot programs for YETI. If you run into any issues in your setup, you can ask questions in the #controls channel of the YETI discord, and check out the WPILib guide as it may have the solution to your problem.

NOTE: This guide is for Windows, macOS, and Linux computers. You will **NOT** be able to write robot programs with ChromeOS.

Download the latest version of the installer from the [WPILib Github](#). Scroll down to Downloads and download the appropriate installer for your operating system.

Read this if you use a mac

Before installing WPILib, mac users will need to install XCode Command Line Tools. These are tools developed by Apple for C++ development which WPILib requires to run. To do so, open the Terminal app on your mac and run the following command

```
xcode-select --install
```

You may need to run this command administrator privileges, in which case run the following command and enter your password when prompted. **Note:** When typing in your password in the terminal, it will look like nothing is happening, but this is just the terminal version of how websites show dots instead of letters in password fields.

```
sudo xcode-select --install
```

Next, you need to install the appropriate installer for the processor your mac uses, either an Intel or M series (arm64) chip. If you are unsure which your mac uses, do the following:

1. Click the apple logo menu in the top left of your screen
2. Click about this mac
3. If it says you have an Apple M1/M2/etc. chip, download the arm64 installer. otherwise download the Intel version

1. Open the file you downloaded
2. Run the WPILibInstaller
3. Click **Start**
4. Choose **Everything**
5. Select the top right option that says **Skip and don't use VS Code**

Why the option that says not recommended?

Selecting **Everything** will install all the development tools you need to code an FRC robot. The way you actually use those tools to write your code is up to you. All the other options will install a dedicated instance of VS Code, a very popular code editor that can be used to write just about any kind of program. If you already have VS Code installed, it will still install a new version. This is one reason I do not recommend this path, as it can be confusing what version of VS Code you are using.

The other reason is because here on Yeti, we use IntelliJ to write our robot code. This is because IntelliJ is specifically designed for developing Java programs, and so has many useful features that VS Code lacks out of the box. Additionally, I have found that Java development in VS Code is simply much buggier because it is not specifically designed for it while IntelliJ is.

Install for this User or Install for all Users? (doesn't apply to mac)

If you share your computer with anyone else, for example a parent, you should install for this user. If your computer is just yours, you should install for all users. The reason being is that installing for all users requires administrator privileges, which you may not have if this is not your computer. The difference does not matter too much, but installing for all users may create less problems in the future in terms of other software dependencies or updates.

IntelliJ

Here on YETI, we use IntelliJ to write our robot code. This is because IntelliJ is specifically designed for developing Java programs, and so has many useful features that VS Code lacks out of the box. Additionally, I have found that Java development in VS Code is simply much buggier because it is not specifically designed for it while IntelliJ is.

Install

1. [Download IntelliJ](#) **Community Edition (NOT Ultimate)**
 1. Scroll down a little on this page to get to the community edition install
 2. Run the installer and install IntelliJ
 3. Open IntelliJ
 1. When you open IntelliJ, it will begin processing your workspace. The progress bar is in the bottom left.

Setup

Java

1. Open settings
2. Expand **Build, Execution, and Deployment** on the left
3. Expand **Build Tools**
4. Select **Gradle**

5. Set **Distribution** to **Wrapper**
6. In the **Gradle JVM** dropdown, select **Download JDK**
 1. Set **Version** to **17**
 2. Set **Vendor** to **Amazon Corretto**
 3. Click **Download**

FRC Plugin

1. Open settings
2. Select **Plugins** on the left
3. Select the **Marketplace** tab at the top
4. Search for and install the FRC plugin

Next steps

You can view a list of [additional resources here](#), including links to learn Java. For a guide for learning robot code, we have a roadmap for the [FRC Ladder series here](#).

If you are interested in getting started developing for a Romi robot to practice robot code, we have a guide for setting that up [here](#).

Learning Robot Code

FRCLadder

[FRCLadder](#) provides a series of videos that together serve as a great introduction into robot code. We've selected some below that are particularly relevant to the programming YETI does.

Note: The hardware and APIs available have changed quite a bit since some of these videos were published. However, the concepts and theory behind them remain the same.

Introduction to FRC Programming and Basic Drivetrain Code

https://www.youtube.com/embed/ihO-mw_4Qpo?si=23MMYKgGsUODgW9I

How PIDs Work and How to Implement Them

Part 1: Dead Reckoning, Bang Bang, and using kP

<https://www.youtube.com/embed/jlKBWO7ps0w?si=FYxEzyTxH8kPpNjk>

Part 2: Using kI and kD

<https://www.youtube.com/embed/Z24fSBVJeGs?si=0kUkjiBfnQ8hoZy5>

Supplemental: A practical example of tuning a PID loop and how each value affects movement

<https://www.youtube.com/embed/qKy98Cbcltw?si=qPtB1oUdjG7YQsqq>

Resources

Java

Java is the programming language we use to write the code that enables the robot to do anything useful or interesting. You have know how to write software in Java to be able to program our robots.

If you don't have any experience programming, it can feel daunting. You have to learn how to think in a specific way using a language that you don't know yet. Fortunately, there are many resources available. The [free Java course on CodeAcademy](#) is a great way to learn Java at your own pace. You'll start from the basics and learn everything you need to know to program effectively.

Git

Git is a version control system for tracking changes across files in a project (aka repository). It is what enables collaborative software development and is used by every programmer. Github is a website that hosts git repositories. There a lots of programs and apps to manage git for you, but my preferred way is the command line. [CodeAcademy has a good course on git as well.](#)

<https://www.youtube.com/embed/hwP7WQkmECE?si=onkfGbzHPzW0qgIt>

You can access all of the code YETI has ever written on our [GitHub page](#).

Some repositories worth looking at:

- [2024 robot code](#)
- [Scouting site](#)
- [The YETI Wiki you're reading this on!](#)

WPILib

- [What is WPILib?](#)
 - [the same content but in pdf](#)
- [WPILib Java API](#)

Also highly recommended is the [Command-Based Programming](#) for getting familiar with the concepts and APIs. Specifically these pages;

- [Commands](#)
- [Command Compositions](#)
- [Organizing Command-Based Robot Projects](#)

My presentation on some basic WPILib concepts

[A textbook on robot development](#) by team TER3M Robotics

Control Theory

[Teaching Rocks to Think](#) is a great blog about [Programming applied to FRC](#).

There are five blog posts pertaining to control theory.

- [Systems & Control Engineering](#)
- [Mathematical Models of the World](#)
- [PID & Controller Design](#)
- [Tuning PID](#) - This one is really cool! It has an interactive guide to tuning a PID loop
- [Supplemental - Vertical Arm](#)

Getting started with a Romi

Romis are small robots that you program the same way you program the big robots we build for the season. they are a great way to get familiar with how commands and subsystems work. you can read more about romis [here](#).

to get started programming a romi, make sure you have [installed all the tools](#) you need for robot development.

we have a starting template for programming the romi on our github [here](#). to get started with it, follow these steps:

1. click the **use this template** dropdown in the top right, then **Create a new repository**
2. make sure your personal github account is selected as the **owner**
3. give the repository a name. anything your heart desires
4. click **create repository**
5. you now have your own personal romi project

setting up on your computer

1. you should now be on the page for your own romi repository. copy the url of this page
2. open intellij
3. open the **file** menu in the top left
4. under the **new** dropdown, select **Project from Version Control**
5. paste in the url of your repository
6. choose where you want your project to be on your computer
7. click **clone**
8. in the new window, wait for the project to finish loading. the loading bar is in the bottom right
9. test that your project is set up correctly by clicking the dropdown in the top right next to the green play button and selecting **Build & Run Romi via Simulate Java**
10. click the green play button
11. a new purple and black window should open up. if it doesn't or you see an error message, put a message in the `#controls` channel to ask for help

Configuring a Robot

Setting steer offsets for a Mk 4i drivetrain

1. Put the robot up on blocks
2. Begin with the bevel gear (the black ring on one side of the wheel) on the modules all facing inward
 1. Use a straight edge to ensure the modules on each side are all facing the same direction
3. Deploy code that sets the angle offset to `Rotation2D.fromDegrees(0.0)`
4. Open the driver station and pull up the labview (default) dashboard. go to the variables tab
5. The value you are looking for could have any name. It will be derived from something like `Rotation2D.getDegrees()`
6. Set this value to the angle offset for each module exactly
 1. e.g. `Rotation2D.fromDegrees(12.3243)`
7. Deploy code
8. Test
9. If a wheel is spinning the opposite direction of what it should be, add 180.0 to the angle offset

Setting the magnet offset for a CANcoder

1. Move the encoder's mechanism to the desired home position
2. Reset the magnet offset of the CANcoder to 0 in phoenix tuner
3. Apply config (hit the button with the down arrow next to **CANcoder Configs**)
4. Hit the green refresh button
5. Check the absolute position of the CANcoder
 1. This will appear in a table under **Self Test**
6. Set the magnet offset to the absolute position multiplied by -1
7. Apply config
8. Hit the green refresh button
9. Confirm that the absolute position now reads 0
 1. If it is still significantly far off from 0, adjust the magnet offset number until the absolute position gets as close to 0 as possible. Remember to hit apply config and the green refresh button every time you change the magnet offset
10. Set this new value in the code
11. Reboot the robot
 1. This is needed so the updated values in the CANcoder are properly distributed to the other CAN devices
12. Confirm that the absolute position still reads 0 at the desired position

Flashing an Radio (pre-2024 Champs Radios)

Prerequisites

- Windows PC with WPILib Tools Installed

Steps

1. Windows search for "Manage Network Adapter settings"
2. Disable Wifi
3. Connect the Ethernet from the port closest to the power (18-24v POE, image included below) to the Radio Power Module (RPM). Connect the RoboRio side of the RPM to the computer.



4. Open the Radio Configuration Utility application
5. Set the team number to 3506 (or the number of the team you are configuring the radio for)
6. Set the robot name. This is optional, but HIGHLY recommended. The name, if specified, will be put in the Wifi network name, and makes it easier to differentiate between robots.
7. Go back to Network Adapter settings and re-enable Wifi
8. Reboot the robot

If you encounter any issues, check out the "Programming your Radio" guide on WPILib (<https://docs.wpilib.org/en/stable/docs/zero-to-robot/step-3/radio-programming.html>)

Testing Robot Code

Testing code before merging it is important to ensure it works as intended. This page discusses how to checkout the correct branch, deploy the code, test it on the robot, and merge changes.

Checking out your branch:

Go to the terminal in IntelliJ by clicking on this icon in the lower-left corner of your screen:



When open, type "git fetch origin" to fetch to your local repository. Then, type "git checkout [branch name]" and insert the name of the branch that contains the code you want to test. It should look like this:

```
SHIKHAs-MacBook-Air:RoboWS shikhagaurav$ git fetch origin
SHIKHAs-MacBook-Air:RoboWS shikhagaurav$ git checkout [branch-name]
```

You should now be on the correct branch. To ensure all of your changes are on it, type "git pull" to update your branch. It should look like this:

```
SHIKHAs-MacBook-Air:RoboWS shikhagaurav$ git pull
Updating 46b32d3..c1e0a29
Fast-forward
 src/main/java/frc/robot/RobotCommands.java | 32 ++++++
 src/main/java/frc/robot/RobotContainer.java | 35 ++++++
 src/main/java/frc/robot/subsystems/ArmSubsystem.java | 4 ++
 3 files changed, 44 insertions(+), 27 deletions(-)
```

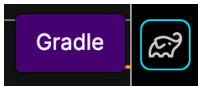
Congrats! You now have all the code that needs to be tested.

Deploying the code:

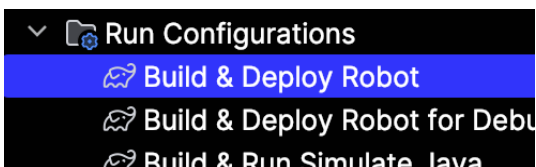
Now, you need to deploy the code to the robot. First, connect to the robot's WiFi. For our team, the network name will be in this format: 3506_robotName. Once connected, go back to IntelliJ and look for this in the top-right portion of your screen:



Ensure the dropdown is set to "Build and Deploy Robot." If that option is unavailable, you may have to scroll through the Gradle string which you can find by clicking this icon:



You'll find the Build and Deploy Robot option in the Run Configurations folder:



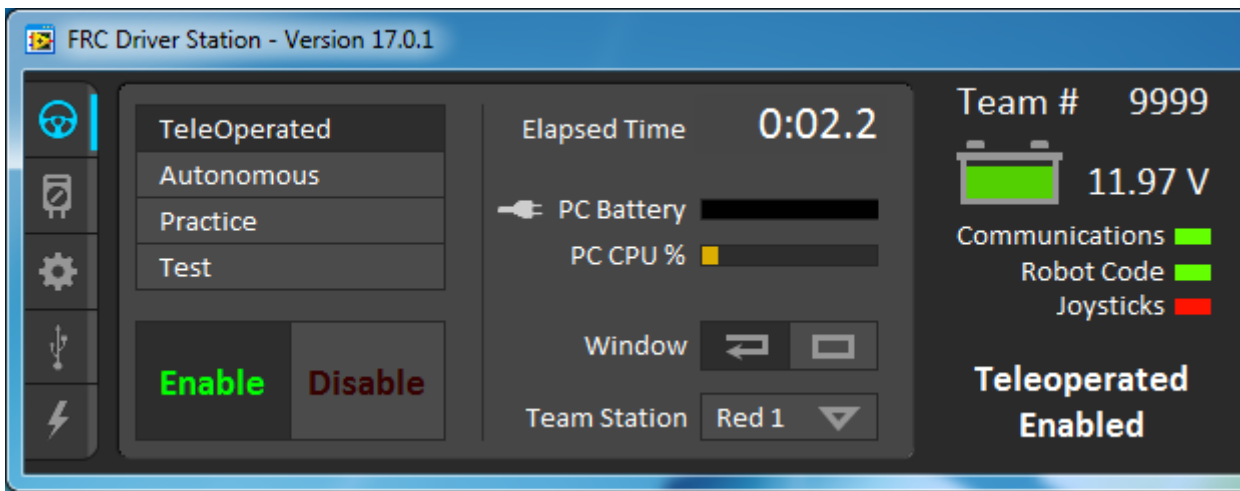
Once you've selected Build and Deploy Robot, hit the green play button. After about 30 seconds, the deployment will be complete.

Note that the Communications and Robot Code bars will turn red on the FRC Driver Station Application while deploying, but will return to green after deployment is complete. Also, if the Communications and Robot Code bars begin to cycle red and green, it means your code is crashing and there is an issue.

Hooray! The code is on the robot!

Test it out:

To run the code on the robot, go to the FRC Driver Station application. It should look something like this:



Ensure that the settings are correct. All three of the bars (Communications, Robot Code, and Joysticks) should be green. The battery should be sufficient. Ensure that you are on the correct drive setting (TeleOperated, Autonomous, Practice, and Test).

If it is in TeleOperated mode, you will need to manipulate the controller to test. If it is on Autonomous mode, ensure that the robot is placed correctly and the correct autonomous program is selected prior to enabling. Hit the green enable button to start the robot and test.

If it does not work as intended, then you will need to change some code and try again.

Nice! Your code has been tested and verified!

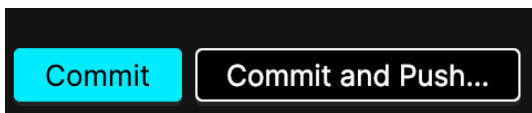
Merging changes:

Now that you've tested out your code and it works, you'll need to get it merged to make sure it's part of the **official robot code**™.

Begin by committing and pushing your changes. You can do this from the terminal, but it's easier to do through the IntelliJ menu bar. Click on this icon:

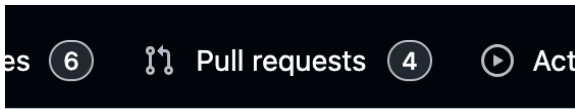


This will show you all of your changes. Type a commit message that details the changes made and their purpose. Make sure your changes are checkboxed. Then hit the commit and push button:



If a box pops up telling you that there are warnings, don't worry about it. However, if the box says that there are errors, then there are problems in your code that you need to fix (and maybe test) before pushing.

Your changes should now be pushed. Go to GitHub and open the repository. Then go to the Pull Requests tab:



Hit the New Pull Request Button in the top right:



This bar should pop up:



Keep the base branch development, but select the dropdown on the compare branch and change it to the branch with the the changes on it. Then hit the Create Pull Request Button and fill in the necessary information. Once completed, submit your pull request.

Once checks are completed, it may mention merge conflicts. GitHub provides ways to resolve these, so make sure those are fixed and any changes are pushed.

Once a mentor has reviewed it, they may request changes that you will need to go in and complete (and maybe retest). Once it is approved, it will be merged and will be part of the **official robot code™**.

Yay! Good job testing!

Git 101

This chapter will go over two main things:

1. What are Git and Github? What are they useful for and how are they used in the real world?
2. How do we use Git on YETI?