Armada Robotics
FRC 2508
Programming Handbook

Table of Contents

Overview

Introduction

Welcome to the programming handbook! This handbook will provide resources for learning to code(we will be using Java), setting up and installing the necessary libraries for FRC, and learning FRC specific libraries. Because there is much that could be discussed for any one topic mentioned in this book, resources will be provided for many of these topics. Many of the topics mentioned in this book are well documented and additional resources and documentation will be provided in each section.

Setup

Git

Git is a version control system that is used to manage code. The Git installer can be downloaded from this webpage. While it is only strictly necessary to install Git itself, it can be easier to use a GUI while getting started with Git. This maps Git commands to buttons in an application so all commands are handled behind the scenes. A list Git GUI clients can be found here, with GitHub Desktop and GitKraken being two of the more popular clients.

Java

Java is the programming language that will be covered in this handbook. It is also possible to program the RoboRIO in other languages such as C++, LabView, and more recently Python and Kotlin. OpenJDK is installed with the WPILib installer so it is not necessary to install anything specific to Java.

FRC Game Tools

The FRC Game Tools package is only necessary to develop code if you are using LabView. It is a package that is managed by National Instruments who make the RoboRIO and LabView. It is necessary to drive the robot, as the package includes the FRC Driver Station which is the program used to control the robot, but it is not necessary to install if you are only developing code in C++ or Java. If you do need to install it, the installer can be downloaded from here.

WPILib

WPILib is the main library used to control the RoboRIO. It provides the main functions that are called during operation and offers many helpful classes for controlling robots. Its installer can be found on the releases page for the WPILib here. The installer will install Visual Studio Code, a code editor, the library itself, and OpenJDK which is necessary to run Java. More detailed instructions can be found here.

Phoenix

Phoenix is the library used for controlling CTRE devices such as Talons, Victors, etc. The installer can be downloaded from this page under the "Tech Resources" tab. Once a robot project is created, you will have to initialize Phoenix as a vendor dependency to use it.

Vendor Libraries

Vendors often provide both an online and offline installation method for their libraries. It is recommended that the offline installation method is used as it does not require an active internet connection while using the project, while the online method does. The offline installation methods for Phoenix, although it still has to be initialized into any project that wishes to use them. By using the keyboard combination Ctrl+Shift+P inside of the installation of VSCode the Command Palette will appear. Typing "Manage Vendor Libraries" and pressing return will allow you to modify the vendor libraries installed in the current project. To initialize a vendor dependency, click on "Install new libraries", using the offline mode to install offline libraries and the online mode to install libraries with an internet connection. Libraries can also be updated by using the respective option or be removed by using the “Manage current libraries” option. Additional Instructions may be found here.

Java

Resources

A basic understanding of Java is strongly recommended before using any FRC specific libraries. Having this understanding will allow for easier comprehension of the working principles of the libraries themselves which will allow for easier use of the library. W3schools and Codecademy both provide decent Java tutorials and there are many video tutorials out there such as this one, typically, newer is better. Don't be afraid to harness your inner Google-fu and search for help. Repl.it is also a useful site for testing code or expanding on examples given in tutorials without needing to install Java on your own computer. The best way to understand a concept is to play around with it, so don't be afraid to have some fun with some test projects.

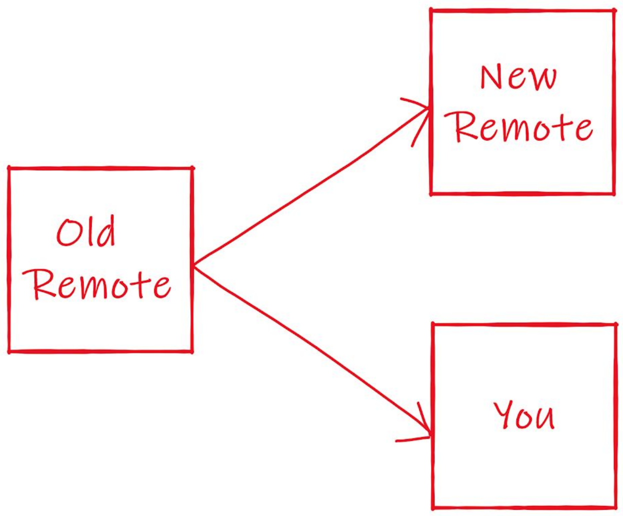
Git

Introduction

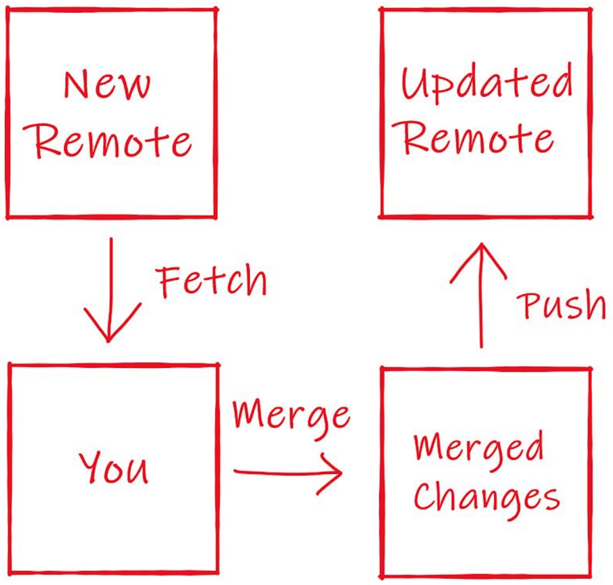
You can get much more done when multiple people are working on the same project, but one of the major challenges when coding as a team is how to have multiple people work on the same project at the same time. Luckily, computer programmers have been dealing with this problem for a long time, well before the internet was created. The solution to this problem is called a version control system. Version control is the concept of versioning every copy of the software, so that, for example, if two people made changes to the same version of the software, their changes could be marked as two separate versions and then combined, or "merged", into one copy. Version control is a way of managing different copies or versions of software and how they interact with each other. Git is a version control system that was invented in 2005 and has become overwhelmingly popular. Git allows for different versions of code to be separated from each other using things called "branches". For example, one branch might have a feature that is experimental and may break the program at times, so it might be desirable to separate this from the main development of the stable version. These branches can be combined or "merged" and if two branches cannot be combined automatically by analyzing the changes between them, it is called a "merge conflict" and must be resolved manually by a human. When developing code with Git, you must first "stage" your changes which tells Git which changes you want to "commit". A commit is the new version of the software, and it can be thought of as an update from the old version. By choosing parts of code to commit, you can organize your commits by what has been updated, and stating what has been updated in a "commit message". In addition, all commits have a "commit hash" that identifies them. In Git, you change branches by "checking out" a new branch. The current commit that is checked out is called the HEAD.

GitHub

An initial point of confusion in version control can often be the role of a hosting service, such as GitHub. While Git is the software that is used for version control, GitHub provides a central place where everyone can access your code. When developing in Git, there are two versions of the repository at any given time. There is the local version on your computer, and there is the remote version that is stored in GitHub's servers. This gives everyone working on the project access to a central remote repository that everyone can see. This concept also transfers over to branches. Every branch has a local branch and a remote branch. For example, you might update a configuration for your project and store that in your local version, but not upload that into the central server. This would mean that the remote branch remains unchanged while the local branch has been modified. When you first commit a new version, this makes a commit to your local branch but does not affect the remote repository on GitHub. This is because you have not synchronized your changes with the remote on GitHub. To do this, you must first "push" your commits to the remote on GitHub from your computer. If someone has made changes to this remote repository and you would like to add them to your



local branch, you must "fetch" from the remote. If someone has made changes and pushed them to the remote and you have also made changes to your local branch, this presents a problem because there is now no easy way for you to push to the remote branch. The solution to this problem is to first fetch the changes from the remote branch and merge them with your changes, then to push this merged version to the remote branch. This is so common that there is a command, "pull", that fetches then merges your changes.



Additional Resources

This was an extremely basic high-level introduction to Git, and there is much more to learn that was not covered. GitHub lists some resources here which also include some interactive activities such as this. And, of course, there are numerous videos on the topic.

WPILib

Introduction

WPILib is the main library that allows for control of the RIO and other control system components. This includes utility functions, helper classes, etc. WPILib is extremely well documented, so this handbook will not cover the usage of it, but will instead provide its documentation which is frequently updated as the library itself updates.

Documentation

The documentation for WPILib can be found here although it is recommended to start with the WPILib Overview and then move to the other programming related topics. The Javadoc generated documentation for all packages, classes, and methods provided in WPILib can be found here. Command-Based programming will be used in all robot projects, and it is recommended to cover the other topics under "WPILib Advanced Programming". It is also crucial to have an understanding of the components in the Control System and their respective purposes.

Vendor Libraries

Introduction

As mentioned earlier, Vendor Libraries allow for the control of components provided by a Vendor. Examples include Phoenix and Rev Robotics' libraries. Once installed, they have to be initialized in the project that wishes to use them before they can be utilized.

Documentation

Documentation for Vendor Libraries is usually easily found. The documentation for Phoenix can be found here. Phoenix also has documentation generated by Javadoc Annotations. CTRE provides examples of the Phoenix library in use with both C++ and Java.

Additional Resources

Chief Delphi

Chief Delphi is an unofficial *FIRST* forum used for everything from showcasing new designs to getting help with programming issues. Because it is such an active forum, it is very likely if you run into an issue that someone else may have had a similar issue, which is why it is recommended to scour the forum before creating a new thread, although if it is an obscure issue or related to a newly released feature there may not be any threads covering that issue. It can be extremely helpful if there is sparse documentation around a certain feature, although that is becoming less rare as documentation improves.

Glossary

Term	Definition
CSA	Control Systems Advisors are volunteers that help with diagnosing issues related to control systems at events and wear bright orange hats
EPR	Edges Per Revolution is a measurement of the number of raw encoder units measured per revolution
FMS	The Field Management System is the system that manages the field during competition
LED	Light Emitting Diodes are small, inexpensive, and power-efficient, so they are used in a wide range of devices
LabView	A visual programming language made by NI
Limelight	An all-in-one vision system
MXP	The Modular Expansion Port is the large expansion port on the front of the RoboRIO
NI	National Instruments is the company that makes the RoboRio and LabView
PID	Proportional Integral Derivative control is a type of control loop that uses proportional, integral, and derivative feedback to cause a system to move to some desired state (e.g. a position, velocity, etc.)
SPI	Serial Peripheral Interface, a specification used to communicate with devices, some Gyros or IMUs use this specification

VSCode	Visual Studio Code, The application used to write and deploy robot code
Vendor Library	A code library provided by a Vendor often for the purpose of controlling components sold by that vendor
Vision	Short for computer vision, meaning any image recognition system running on the robot, typically uses special tape to identify key locations on the field