# Programming 1
# for FIRST Robotics

< Copper Country Robotics >

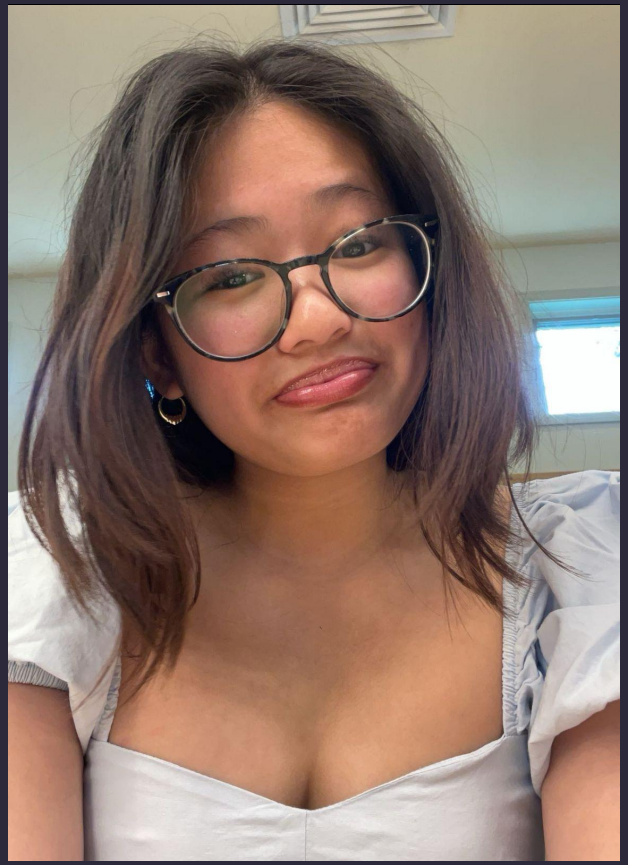
# Will Kanar
## Team 5509

- Been with FIRST for 8 years
- Lead programmer for 3 years
- Has never gotten something to work the first time

# Athena Lieu

## Team 3452

- 6 years in FIRST
- Lead programmer Scouter and Programmer



# WPILib and VS Code

## WPILib

The standard software library provided for teams. Part of this is a modified version of VS Code

## VS Code

A code editor/IDE used by over 14 million programmers worldwide.

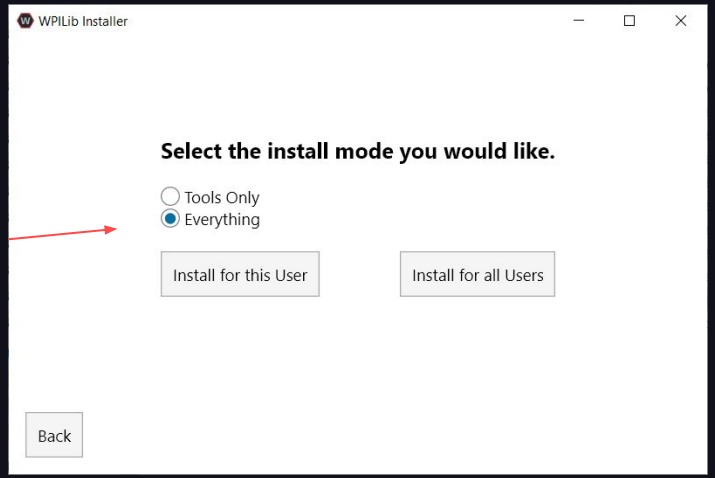
# Where to get help?

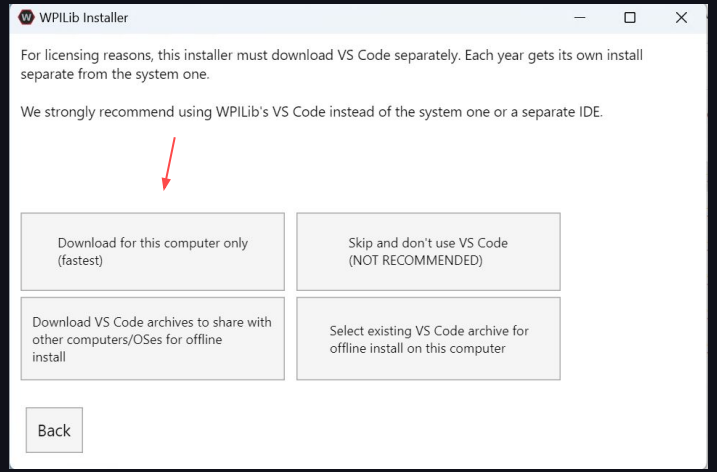
There are many great places.

- Mentors (CCR included!)
- docs.wpilib.org WPI Lib documentation
- Google specific documentation (docs.ctr-electronics.com, docs.revrobotics.com)
- Chief Delphi (chiefdelphi.com)
- Game Manual

# Installation

- WPILib Installation Guide
  - https://docs.wpilib.org/en/stable/docs/zero-to-robot/step-2/wpilib-setup.html



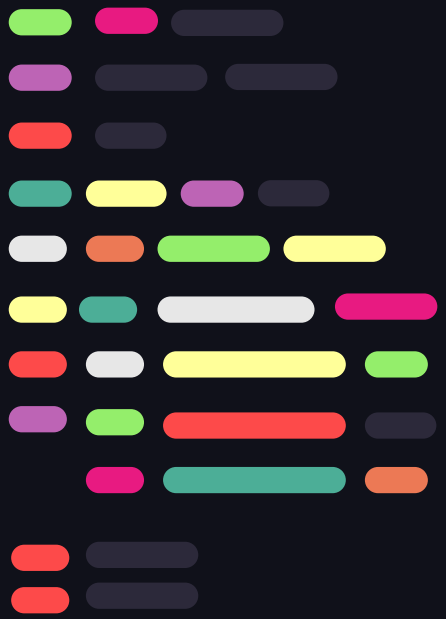




Follow the page above.

# Re: Installation

- **Tips**
  - Remember to uninstall old versions or pin the new version to your taskbar.
  - Make sure to click “Install Everything” and “Download for this computer only” when installing.
  - If NI instruments pops up, just close the tab.

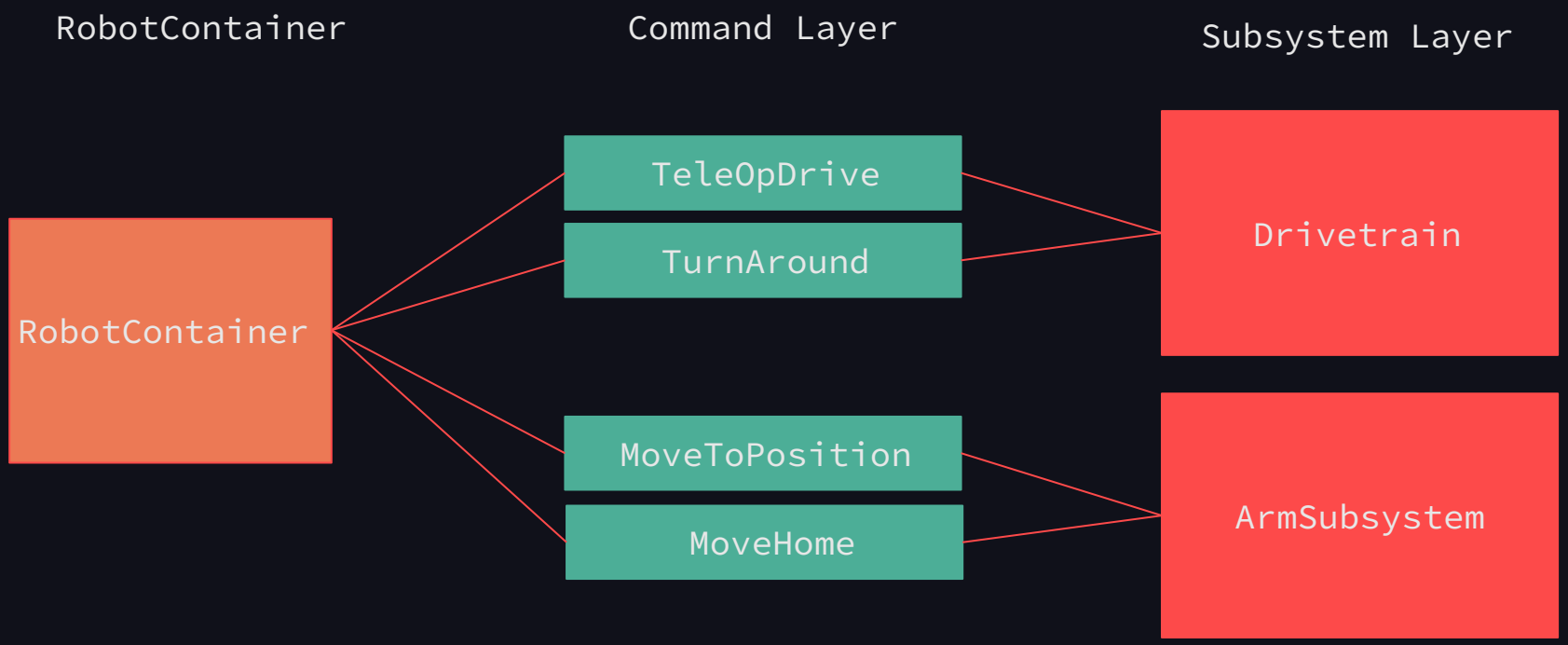
# CAN IDs



## Phoenix Tuner X and REV Hardware

Software to to update, configure, analyze, and control your devices

# Command Based

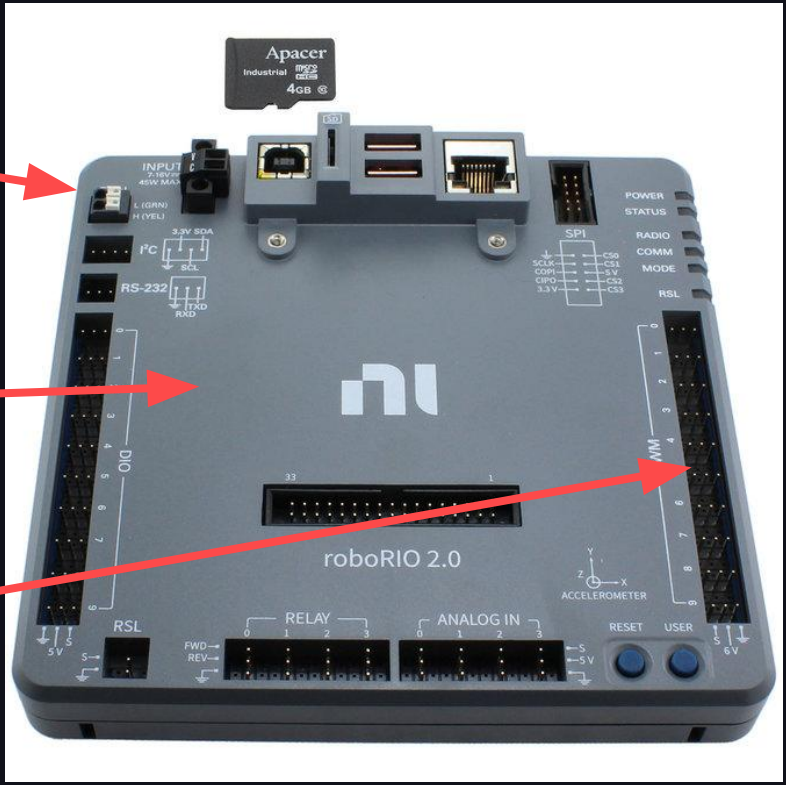


# The Brain of the Beast

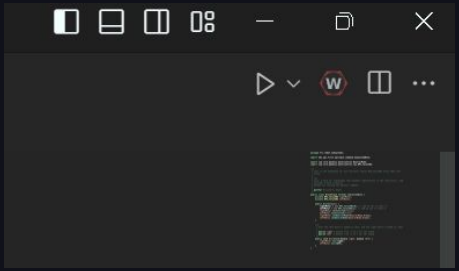
CAN Output

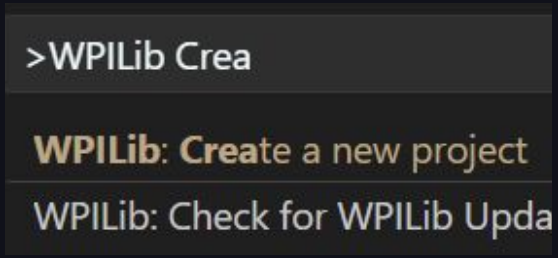
Frisbee

PWM

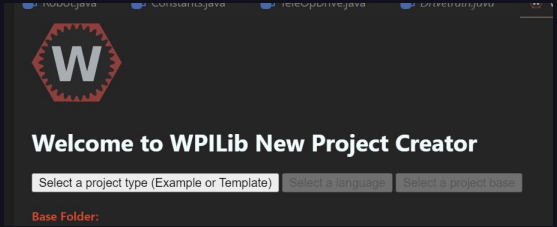


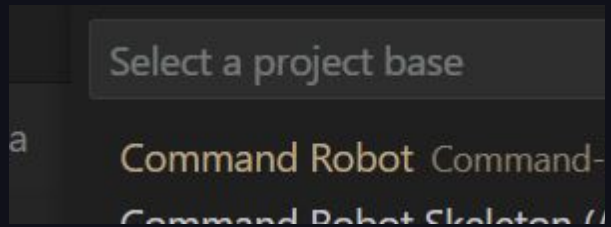
# Creating Project in WPILib






- Open WPILib VSCode
- Click this logo in the top right
- Type, “Create” and Click on “Create a new project”
- Click “Select a project type”
- Select “Template”, then “java”, then “Command Robot”
- Select a folder and give your project a name.



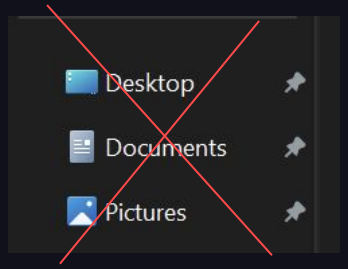



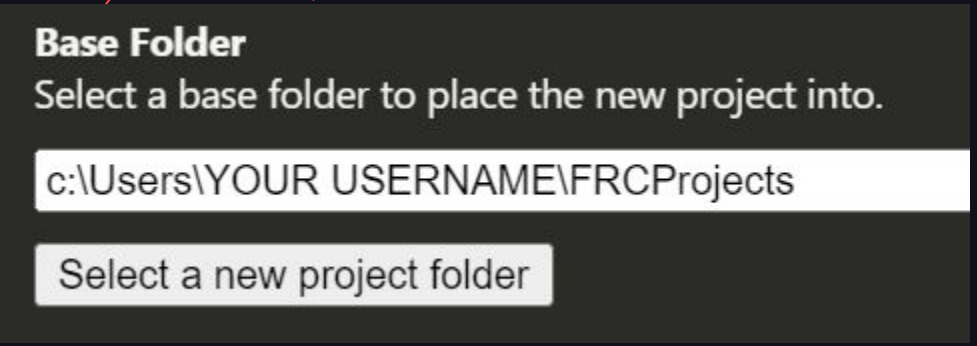

*

# Prepping the Project


OneDrive and WPILib are *not* friends

These may be in OneDrive




Try your own folder, like “FRCProjects”.

Remember where you put it!

# Welcome to WPILib New Project Creator

template | java | Command Robot

**Base Folder**
Select a base folder to place the new project into.

c:\Users\Will\Documents

Select a new project folder

**Project Name**

RunThrough

Name

**Create a new folder?** ☑
This creates a new folder at Base Folder\Project Name. Highly recommended to be checked. O

**Team Number**

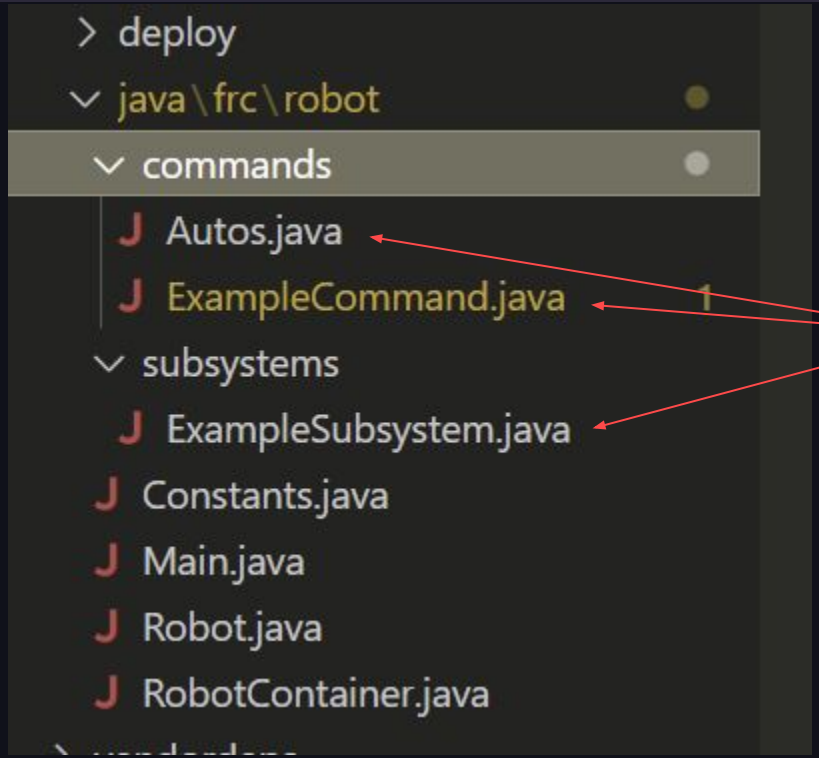
0

Team Number

**Enable Desktop Support** ☐
This is needed for simulation and unit testing support, however there are some cases where thi
"WPILib: Set Desktop Support" at any time.

Generate Project

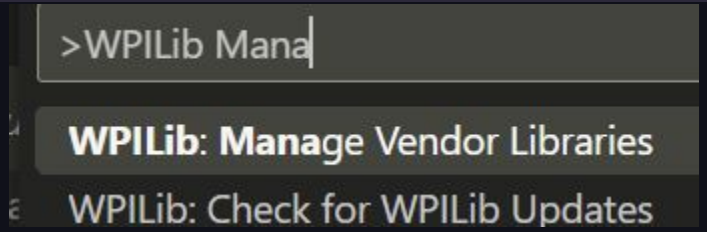
GO!

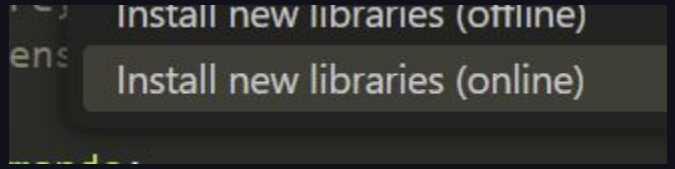



You can use right click to delete these if you want. We won’t use them today

https://maven.ctr-electronics.com/release/com/ctre/phoenix6/latest/Phoenix6-frc2024-latest.json

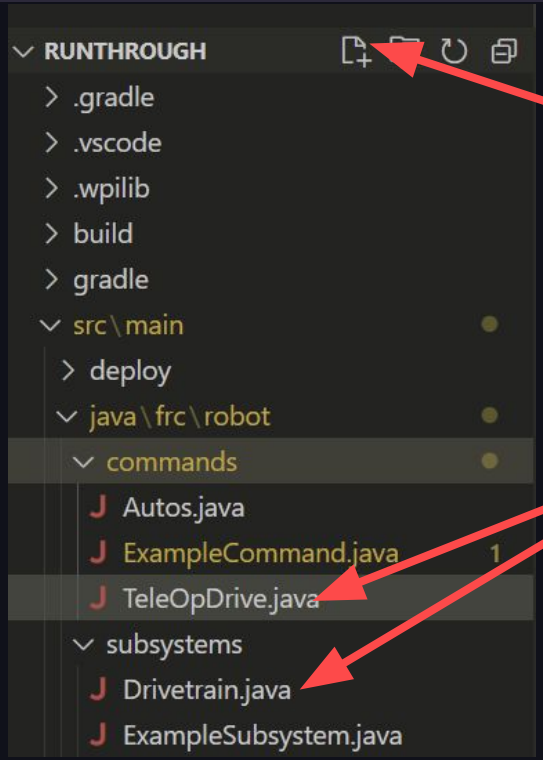
https://maven.ctr-electronics.com/release/com/ctre/phoenix/Phoenix5-frc2024-latest.json




https://docs.wpilib.org/en/stable/docs/software/vscode-overview/3rd-party-libraries.html

# Creating Command and Subsystem




Click on new file

To make "TeleOpDrive.java" and "Drivetrain.java

Put them in "commands" and "subsystems", respectively.

# In the Subsystem

```java
package frc.robot.subsystems;

import com.ctre.phoenix.motorcontrol.can.WPI_TalonSRX;

/**
 * This is the Subsystem for out two-motor {@link WPI_TalonSRX Talon SRX} tank
 * drive.
 */
public class Drivetrain extends SubsystemBase {
    private WPI_TalonSRX rightMotor;
    private WPI_TalonSRX leftMotor;
}
```

These usually appear automatically

Creates a right and left motors

# In the Subsystem

```
public class Drivetrain extends SubsystemBase {
    private WPI_TalonSRX rightMotor;
    private WPI_TalonSRX leftMotor;
    public Drivetrain(){
        rightMotor = new WPI_TalonSRX(0); // CAN ID set in Tuner X
        leftMotor = new WPI_TalonSRX(1); // CAN ID set in Tuner X
        rightMotor.setInverted(false);
        leftMotor.setInverted(true);
        rightMotor.setNeutralMode(NeutralMode.Brake);
        leftMotor.setNeutralMode(NeutralMode.Brake);
    }

}
```

Tells the robot how to construct the Drivetrain

The IDs for out motor controllers

Invert one motor, since it's facing the other way.

Have the wheels hold their position at rest

# In the Subsystem

```
public class Drivetrain extends SubsystemBase {
    private WPI_TalonSRX rightMotor;
    private WPI_TalonSRX leftMotor;
    public Drivetrain(){
        rightMotor = new WPI_TalonSRX(0); // CAN ID set in Tuner X
        leftMotor = new WPI_TalonSRX(1); // CAN ID set in Tuner X
        rightMotor.setInverted(false);
        leftMotor.setInverted(true);
        rightMotor.setNeutralMode(NeutralMode.Brake);
        leftMotor.setNeutralMode(NeutralMode.Brake);
    }
    public void driveTank(double right, double left) {
        rightMotor.set(right);
        leftMotor.set(left);
    }

}
```

A method (function) to apply motor values

Two decimal inputs. Power from -1 to 1 for the left and right

Apply these values to each motor

# In the Command

```
package frc.robot.commands;

import edu.wpi.first.wpilibj2.command.Command;
import frc.robot.subsystems.Drivetrain;

public class TeleOpDrive  extends Command{
    private final Drivetrain m_drivetrain;
    public TeleOpDrive(Drivetrain subsystem) {
        m_drivetrain = subsystem;
        addRequirements(m_drivetrain);
    }
}
```

Define Command constructor

Use the Drivetrain provided to the Command

Make the Command require the Drivtrain

# In the Command

```
import frc.robot.RobotContainer;
public class TeleOpDrive  extends Command{
    private final Drivetrain m_drivetrain;
    public TeleOpDrive(Drivetrain subsystem) {
        m_drivetrain = subsystem;
        addRequirements(m_drivetrain);
    }
    @Override
    public void initialize() {
        m_drivetrain.driveTank(0,0);
    }
    @Override
    public void execute() {
        m_drivetrain.driveTank(RobotContainer.getDriverControllerAxis(1),
RobotContainer.getDriverControllerAxis(5));
    }
}
```

New import

Runs Once at the start. Sets the speeds to 0.

The execute() repeats.

Sets the motor speeds to out controller's stick values. We will finish this later.

# In RobotContainer.java

```
public final Drivetrain m_drivetrain = new Drivetrain();
. . .
private static final CommandXboxController m_driverController = new
CommandXboxController(
      OperatorConstants.kDriverControllerPort);
. . .
public RobotContainer() {
    // Configure the trigger bindings
    configureBindings();
    m_drivetrain.setDefaultCommand(new TeleOpDrive(m_drivetrain));
}
public static double getDriverControllerAxis(int axis) {
    return m_driverController.getRawAxis(axis);
}
```

Create our Drivetrain (Line 24)

Add the word “static” (Line 26)

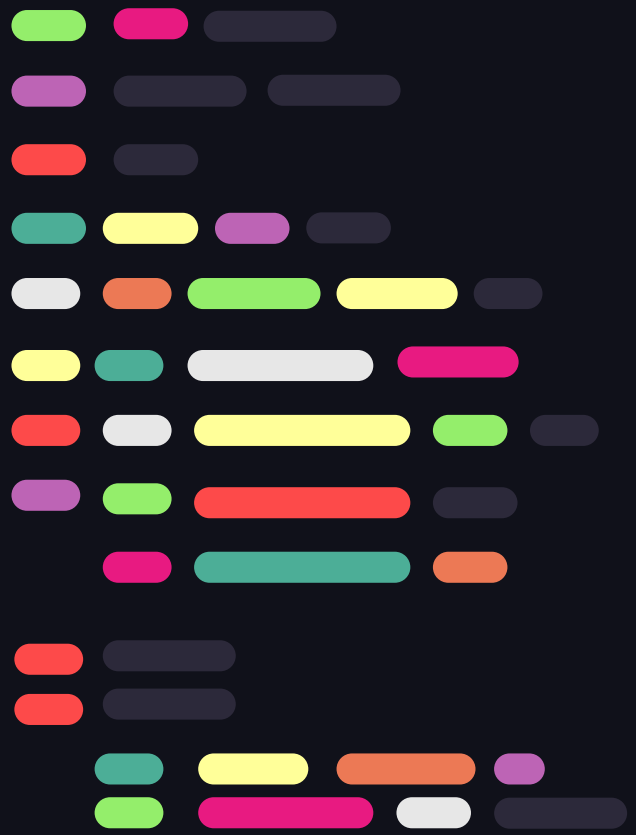
Add this line to make TeleOpDrive the default (Line 33)

Add this method to allow us to grab stick inputs (Line 35)

The method returns a decimal from -1 to 1 for the axis provided here

# Building a Running

- Click the “W” icon at the top right
- Type “Build” and click on “Build Robot Code”
- Read the terminal to fix errors
- Turn the robot on and connect to its wifi
- Click the “W”, Type “Deploy” and click on “Deploy Robot Code”



# Questions?