Robotbuilder & Command Based Robot

FRC

FIRST Robotics Competition

GRANITE CITY GEAR HEADS

Team 3244

Robotbuilder & Command Based Robot

About Us

Corey Applegate

- 20 Years Industrial Controls Technician using Rockwell Automation PLC and Motion Control, Kuka Robotics, Fanuc Robotics
- 4 years Mentoring a LEGO Mindstorms Club for 5-6 graders
- 1 season Mentoring VEX
- 1st full year with FIRST was 2016 Stronghold
- Hobbies and skills include
  - 3d Printing, Autodesk Inventor, Raspberry Pi, Arduino

Austin Applegate

- Junior Tech High School
- 1st Year FIRST Stronghold
- Team responsibilities
  - Driver, Build, Some Software
- Hobbies
  - PC builds and Gaming, Football

Team 3244

Robotbuilder & Command Based Robot

What we hope to cover

- Quickly create the backbone of the robot code Using RobotBuilder
- Import project into Eclipse
- Explore the parts of the Project
- Test robot IO
  - without writing a single line of code
- Create commands
  - link Joystick inputs to Subsystems
- Create Methods in Subsystems
  - for the Commands to use
- Test/Tune a PID Subsystem
  - again without writing a single line of code
- Link the setpoints of this PID subsystem to buttons
- Add User code to control the Drivetrain

Team 3244

Robotbuilder & Command Based Robot

Pros and Cons to Command Based programing

Pros

- Can **develop** and **debug** subsystems individually
- Easily create a more **complex autonomous** using command groups
- Well **organized code** (Starting with Robotbuilder)
- Easily **split functions** or subsystem coding responsibilities between multiple students
- **Reuse commands** in both Auto and Teleop

Cons

- **Looks complicated** because there are so many files to create for operator IO, Robot IO, Subsystems, Commands
- Must treat some hardware differently such as limit switches and avoid loops and waits
- Must link and create objects in correct order to avoid errors

Team 3244

Robotbuilder & Command Based Robot

Parts of a Command Based Robot Program
SUBSYSTEMS

- **Different subsets** of the robot.
  IE Drivetrain, manipulators, vision

http://wpilib.screenstepslive.com/s/4485/m/13810/l/241892-what-is-command-based-programming

Team 3244

Robotbuilder & Command Based Robot

Parts of a Command Based Robot Program
COMMANDS

- What do you want your subsystem to do?

http://wpilib.screenstepslive.com/s/4485/m/13810/l/241892-what-is-command-based-programming

Team 3244

Robotbuilder & Command Based Robot

Parts of a Command Based Robot Program

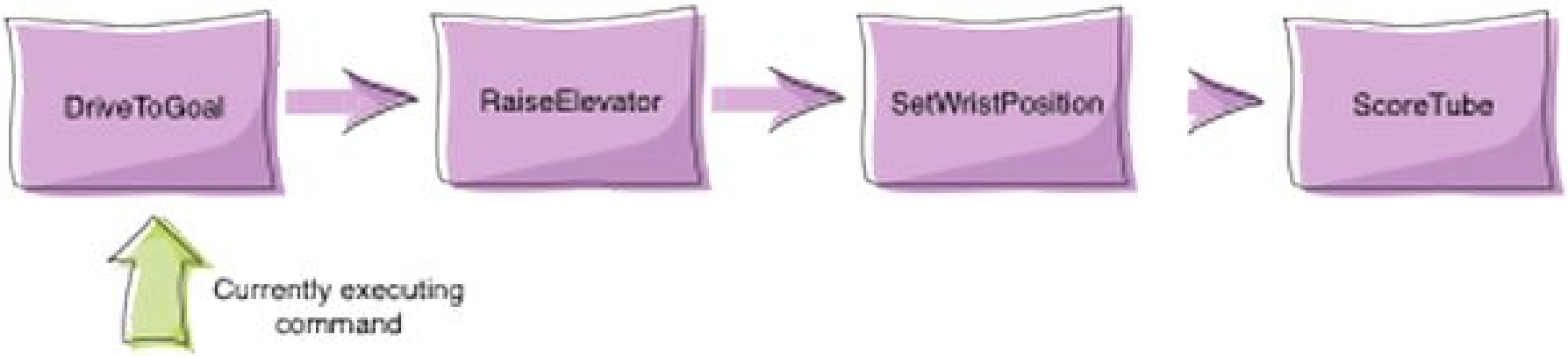
- **Robot**
  - Main program
- **OI**
  - All Joystick controls created and assigned functionality
  - Smartdashboard Buttons
- **RobotMap**
  - Maps all Robot Inputs and Outputs Plugged into the RoboRio

Team 3244

Robotbuilder & Command Based Robot

How Commands work

- Commands let you break up the tasks of operating the robot into small chunks.
  - Each command has an **execute()** method that does some work and an **isFinished()** method that tells if it is done.



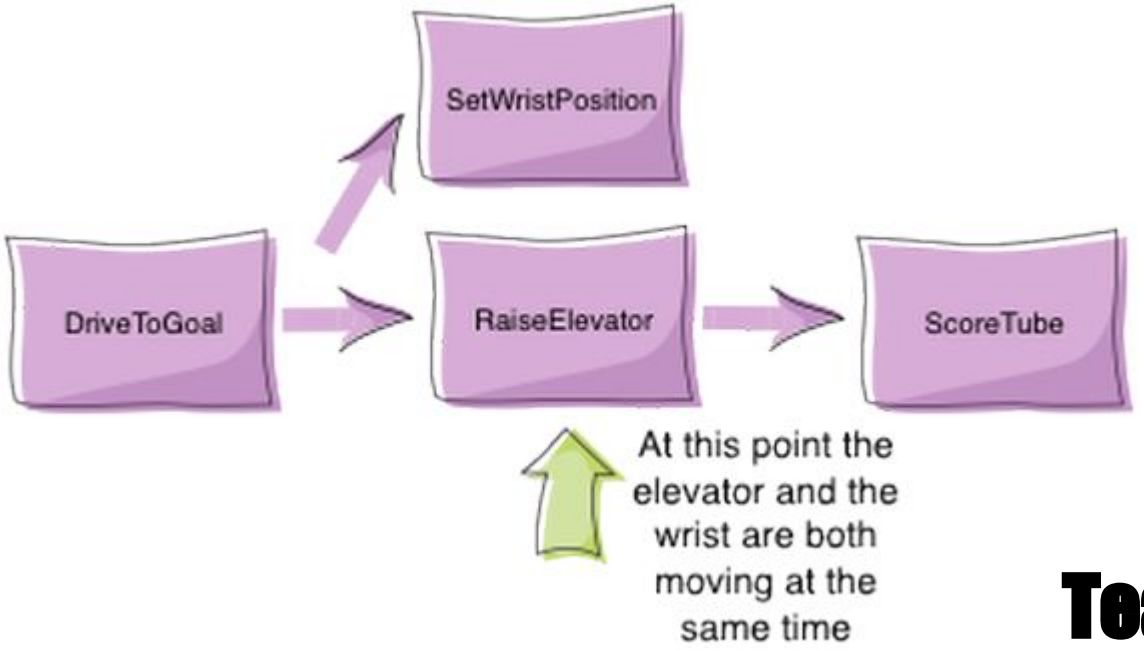
http://wpilib.screenstepslive.com/s/4485/m/13810/l/241892-what-is-command-based-programming

Team 3244

Robotbuilder & Command Based Robot

Concurrency

- Sometimes it is desirable to have several operations happening concurrently. In the previous example you might want to set the wrist position while the elevator is moving up. In this case a command group can start a **parallel command** (or command group) running.



Team 3244

http://wpilib.screenstepslive.com/s/4485/m/13810/l/241892-what-is-command-based-programming

Robotbuilder & Command Based Robot

How is Robotbuilder going to help us?

- **Menus and GUI's** to build our subsystems and assign hardware to these systems
- **Error checking** to be sure required components are added to special types of systems or commands
- **Drop menus** populate with only valid options for component
- **Creates** all the associations and **many useful code entries** for all devices attached to the robot.
- **Without a line of code written** we can test components connected to the robot
- **Create a wire map** for the Electricians.

Team 3244

Robotbuilder & Command Based Robot

Start Eclipse and open Robot builder

1. Start Live Demo
2. There are a few set up steps before robot Programing

Team 3244

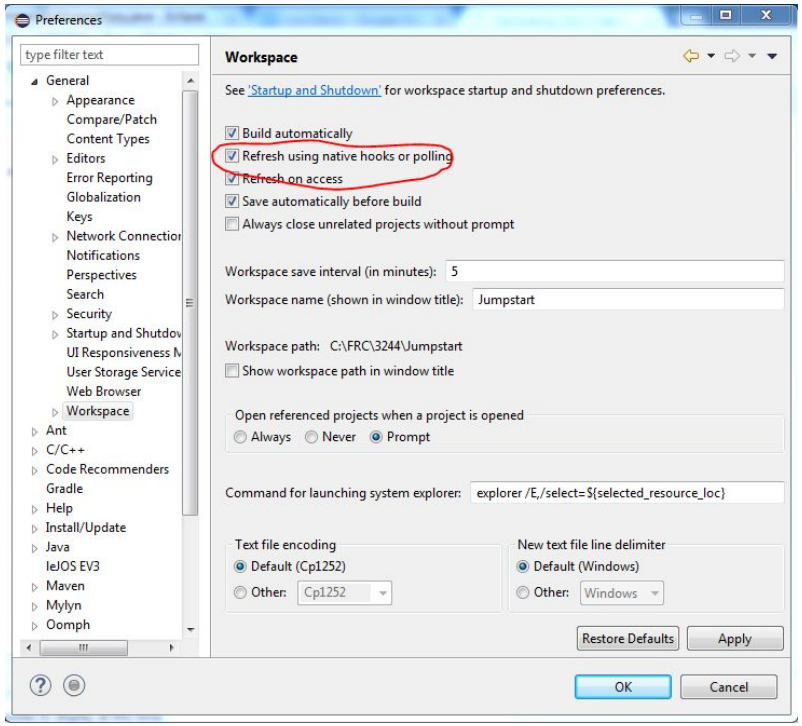
Robotbuilder & Command Based Robot

Live Demo

Start Eclipse

1. After installing Eclipse per the Wpilib.Screenstepslive instuctions a new setting must be set.
   a. Windows>Preferences>General>Workspace

   This setting will allow Eclipse to refresh automatically each time you return to Robotbuilder to add or change settings after the initial project import.

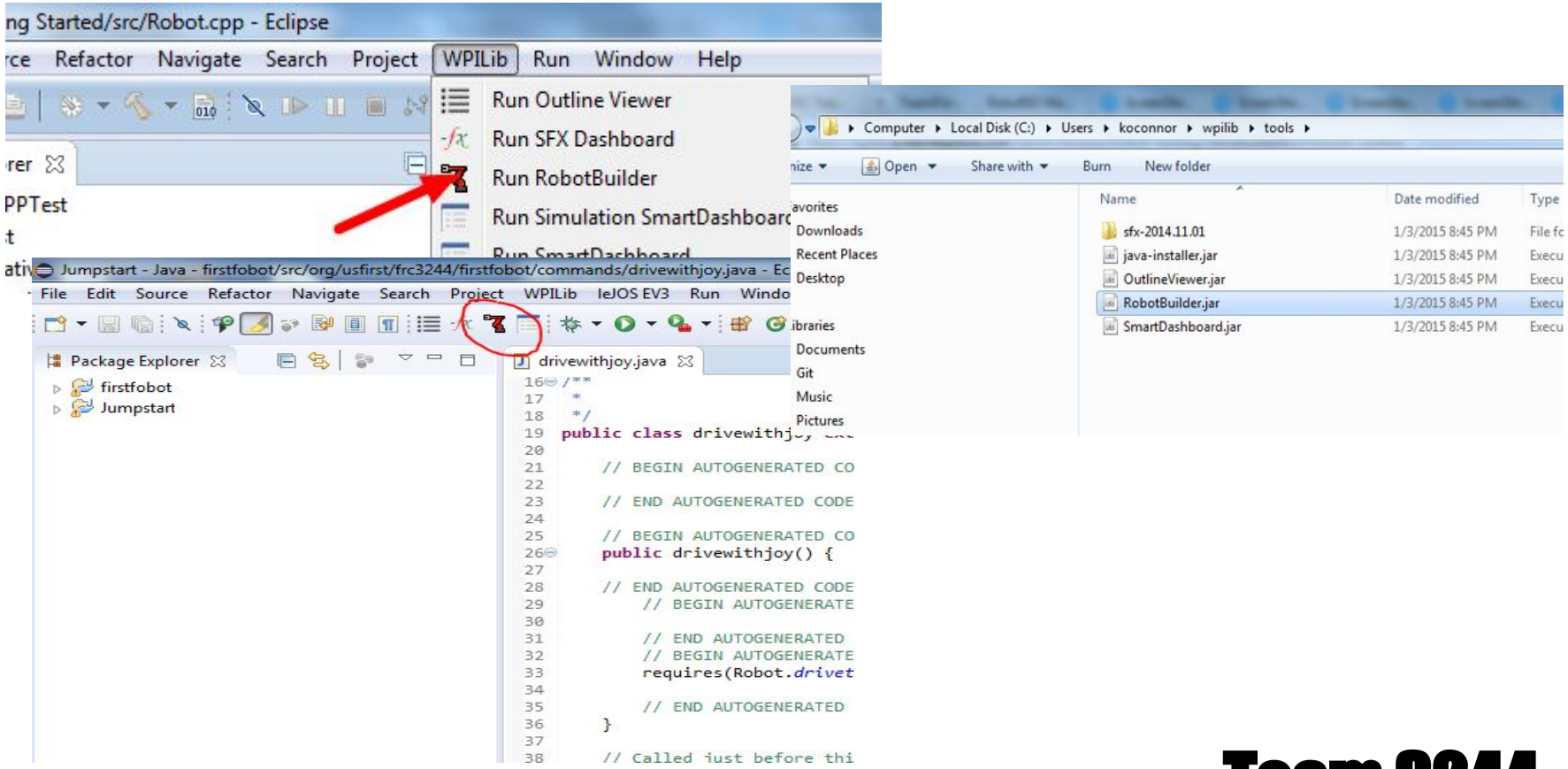


Team 3244

Robotbuilder & Command Based Robot

Live Demo

Start Robotbuilder Tool

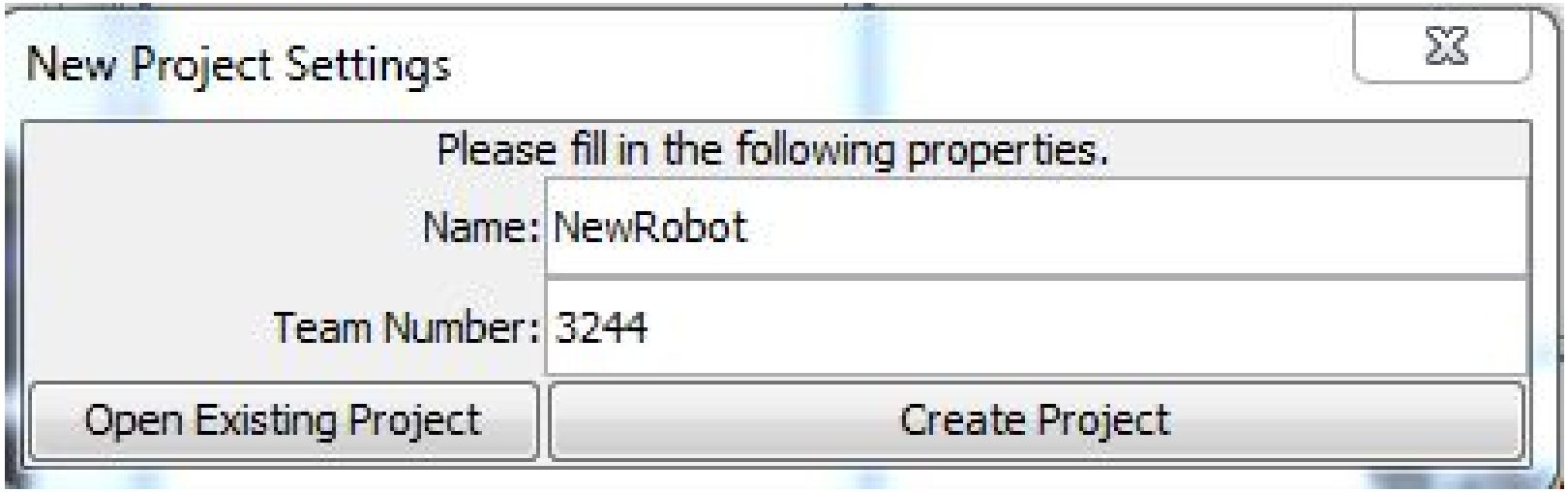


Team 3244

Robotbuilder & Command Based Robot

Live Demo

Start a New Project



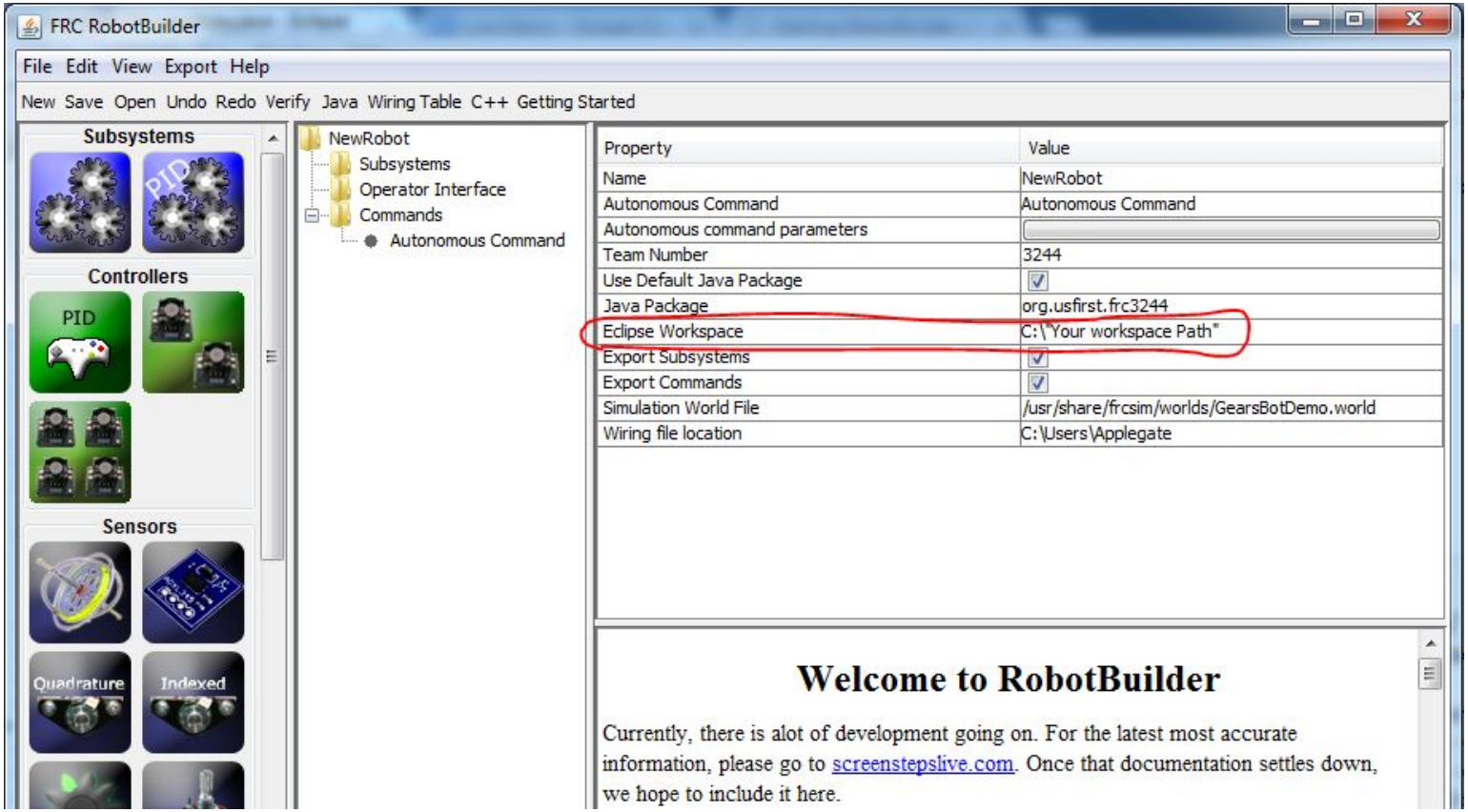
1. Fill in Name and Team Number

Team 3244

Robotbuilder & Command Based Robot

Live Demo

Complete the Eclipse Workspace location



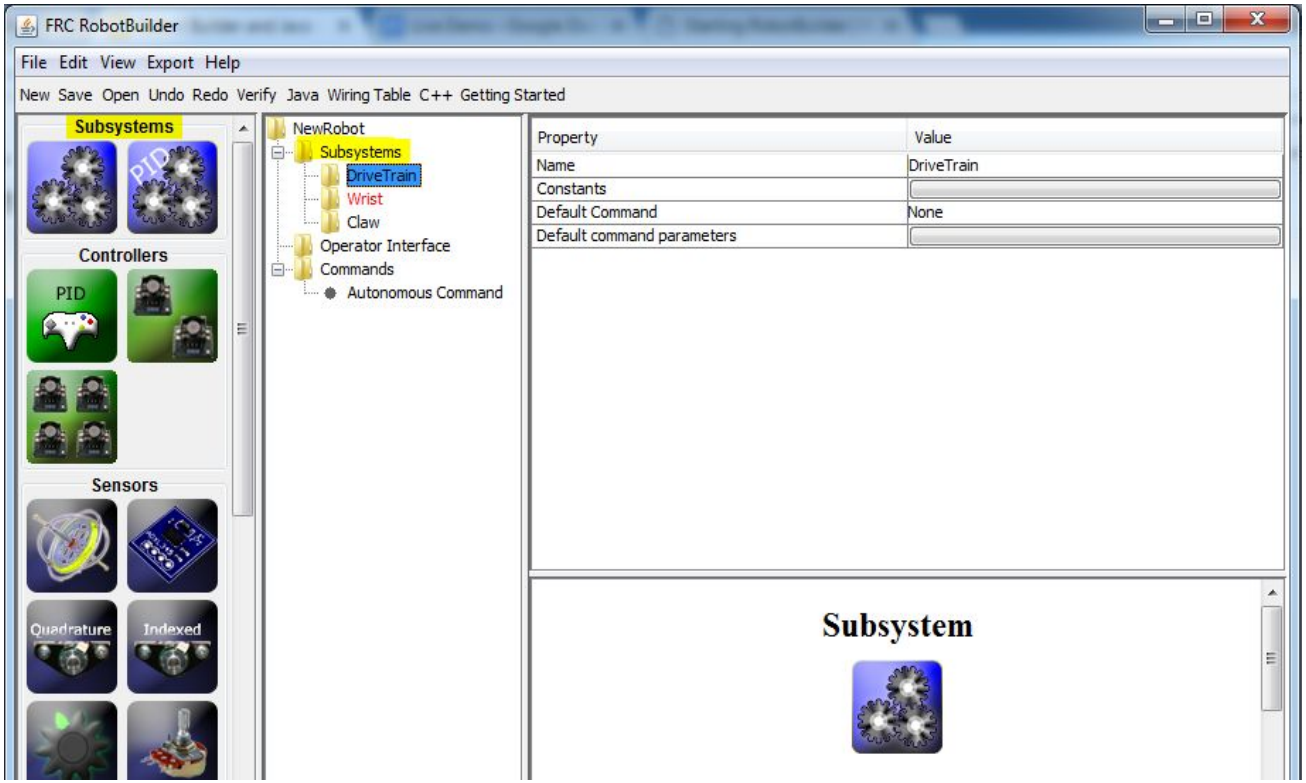
Team 3244

Robotbuilder & Command Based Robot

Live Demo

Divide Robot into Subsystems

1. Subsystem 1
   a. Standard Type "Drivetrain"
2. Subsystem 2
   a. PID Type "Wrist"
3. ….



Team 3244

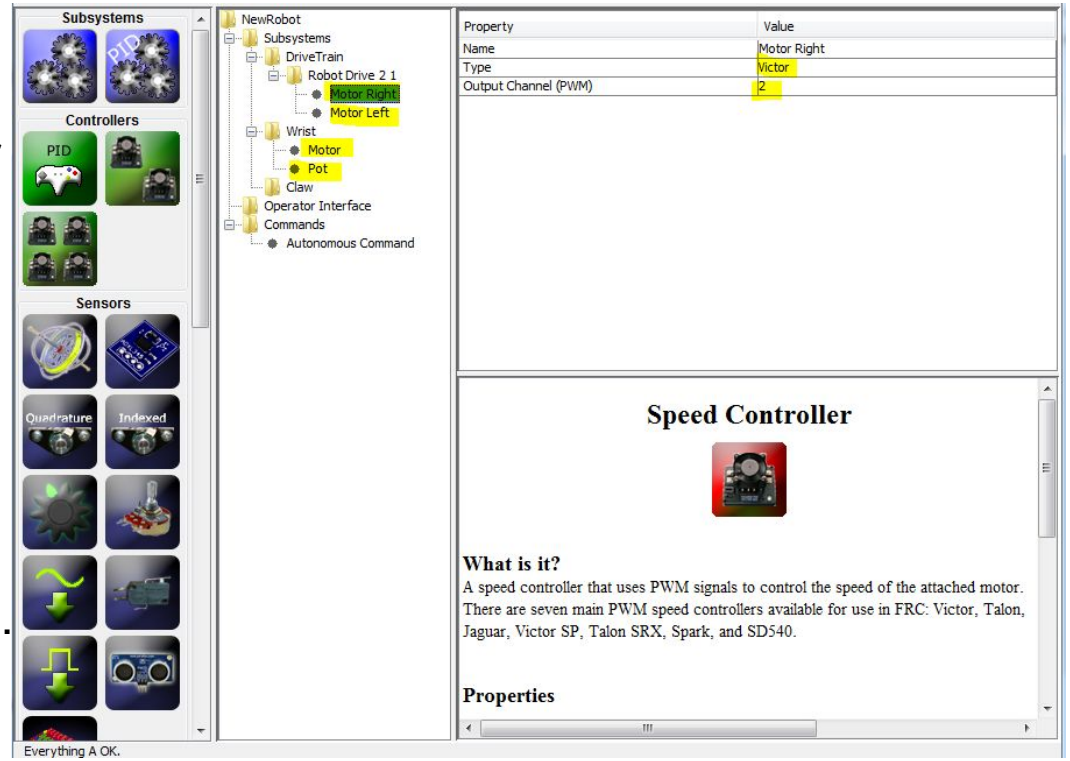
Robotbuilder & Command Based Robot

Live Demo

Add Actuators and Sensors

1. Add Actuators to Subsystems by either dragging into the workspace from the sidebar or right click menu
   a. Speed Controller select Type and PWM Port
   b. CAN Speed Controllers Have their own icon.

2. Add Sensors the same way
   a. Set DIO/Analog Channels.
   b. Set other special setting per device type.



Team 3244

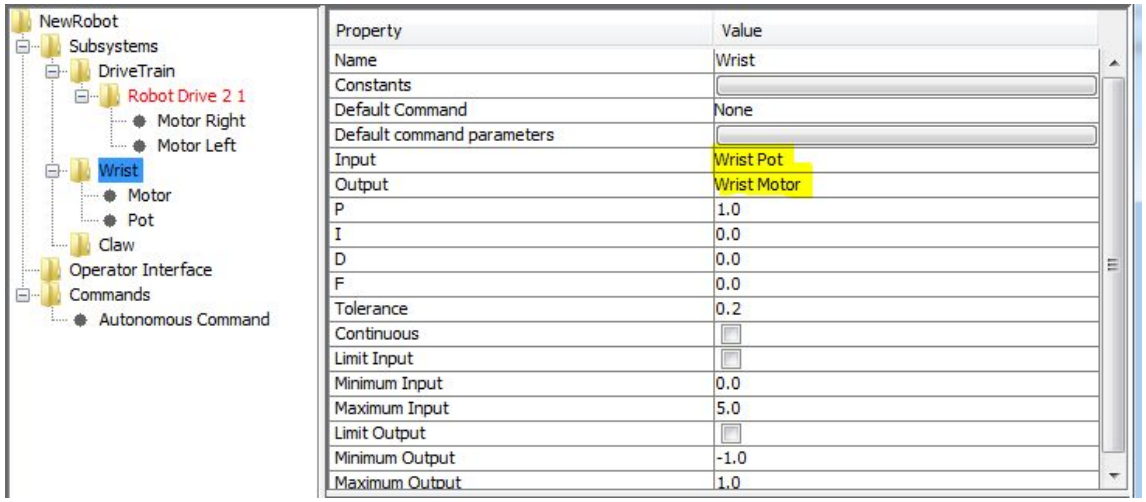
Robotbuilder & Command Based Robot

Live Demo

Correct Red Highlighted Errors

1. Robot Drive will alway Prefill with the first Speed Controller. You will need to Manually select in drop downs

2. PID Type Subsystems require a Speed Controller and an input. Analog or Encoder



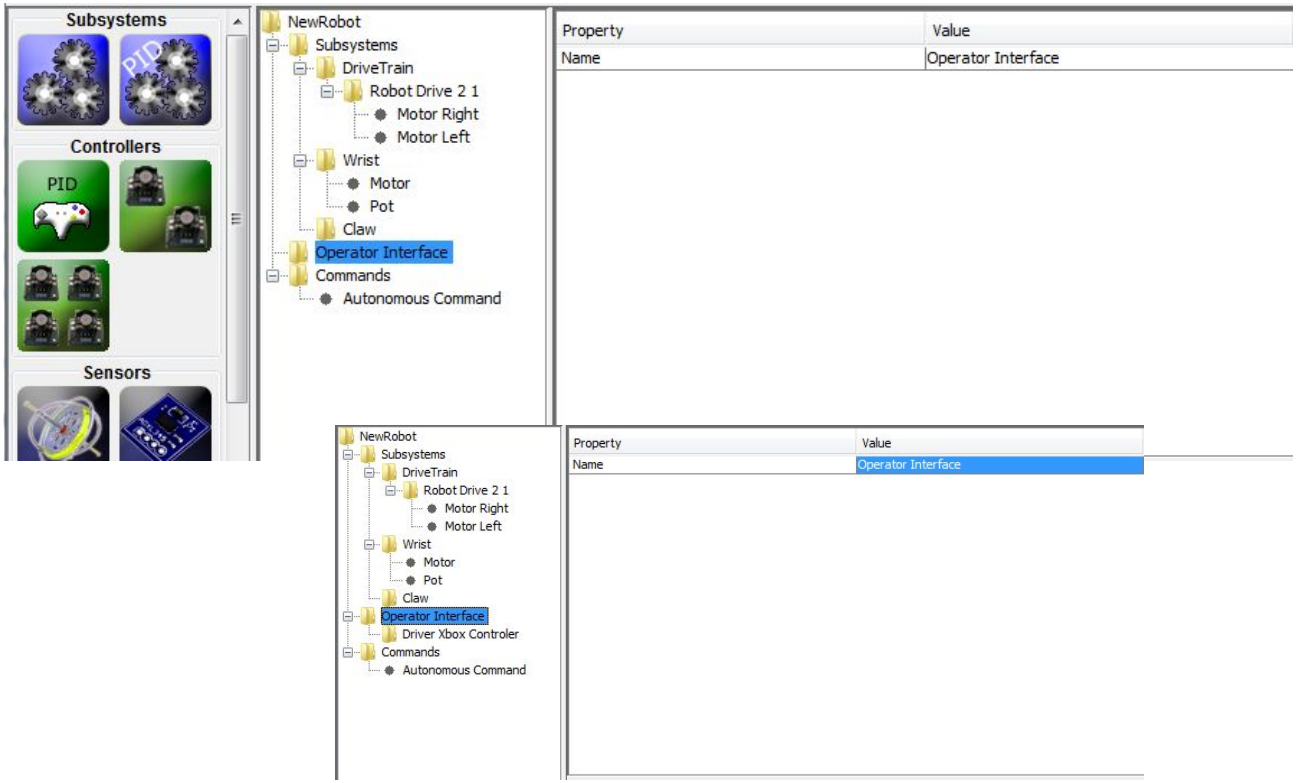
Team 3244

Robotbuilder & Command Based Robot

Live Demo

Add Operator Interface

1. Drag or right click Operator Interface and add Joystick
   a. Name Joystick
   b. Set Device ID



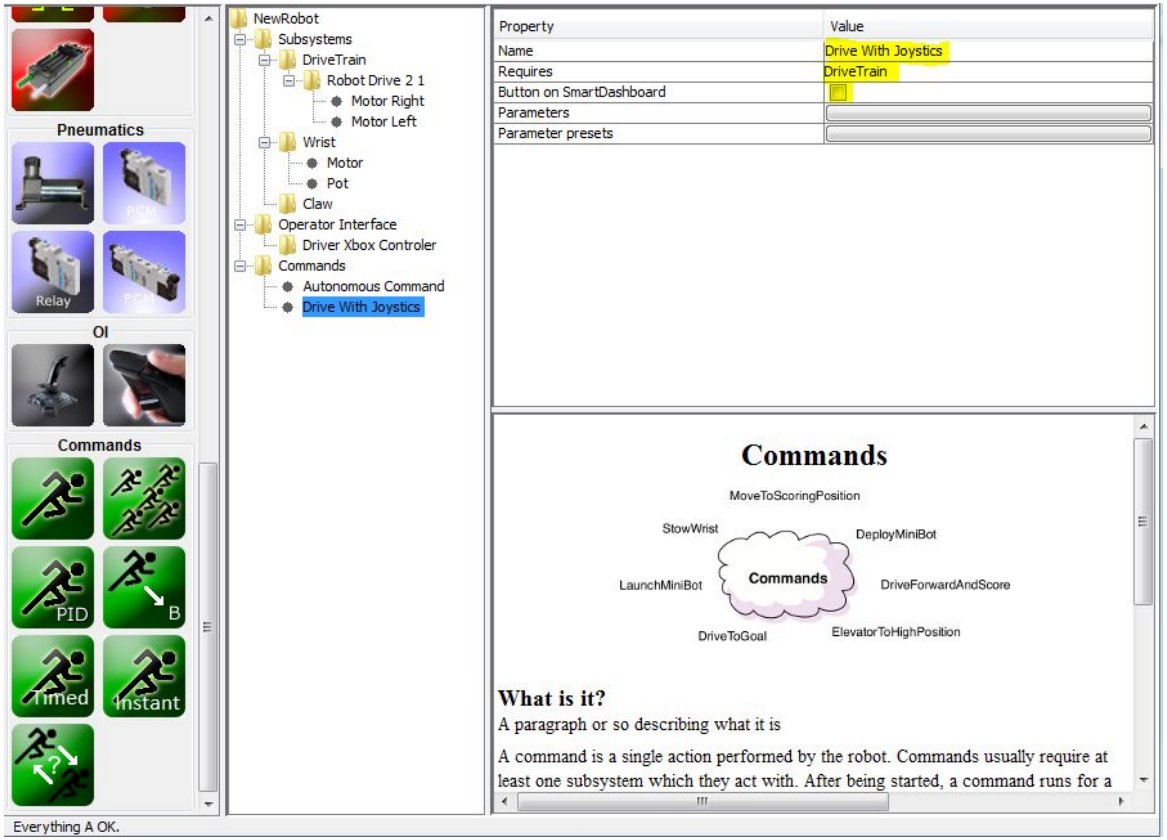
Team 3244

Robotbuilder & Command Based Robot

Live Demo

Create a Command

1. Right Click Commands or drag in from tools a simple command.
   a. Name Command. Use a good Descriptive title to identify its intentions.
      *I like to start the name with the sub system
   b. Set the Requires to the Subsytem this command is going to use.
   c. Uncheck Button on SmartDashboard if you DO NOT wish to see this command on driverstation



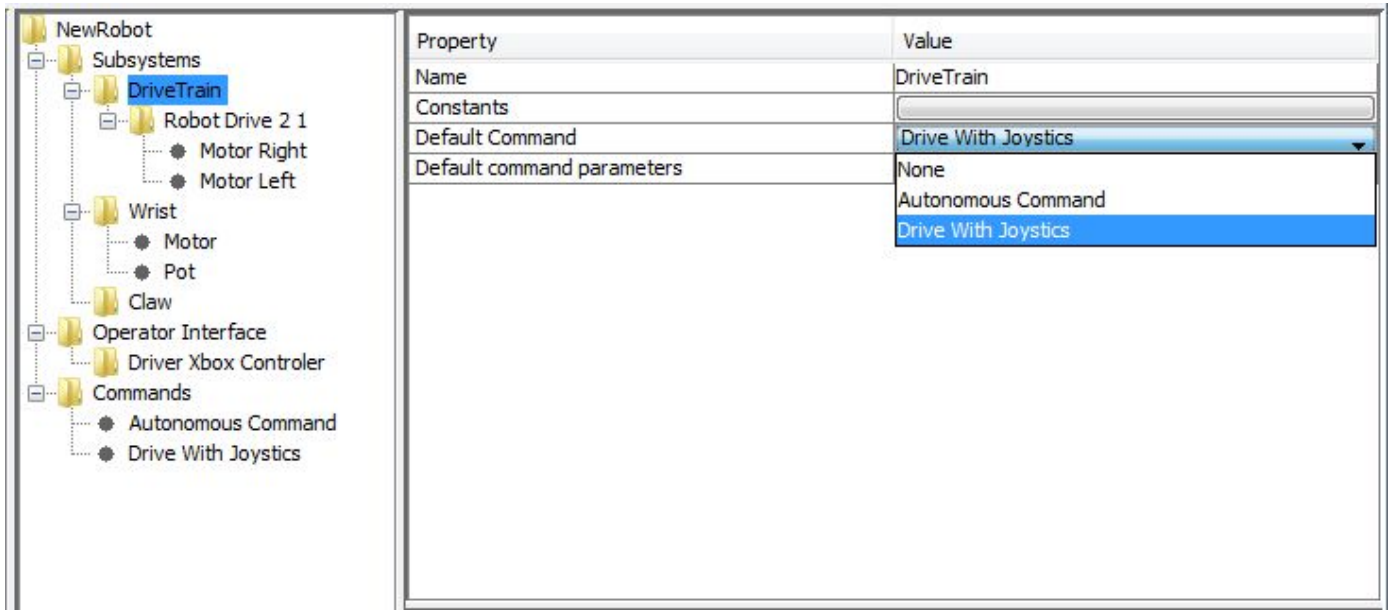
Team 3244

Robotbuilder & Command Based Robot

Live Demo

With Command Created Set Subsystem default Command

1. This will be the command that runs if no other commands is using this subsystem.



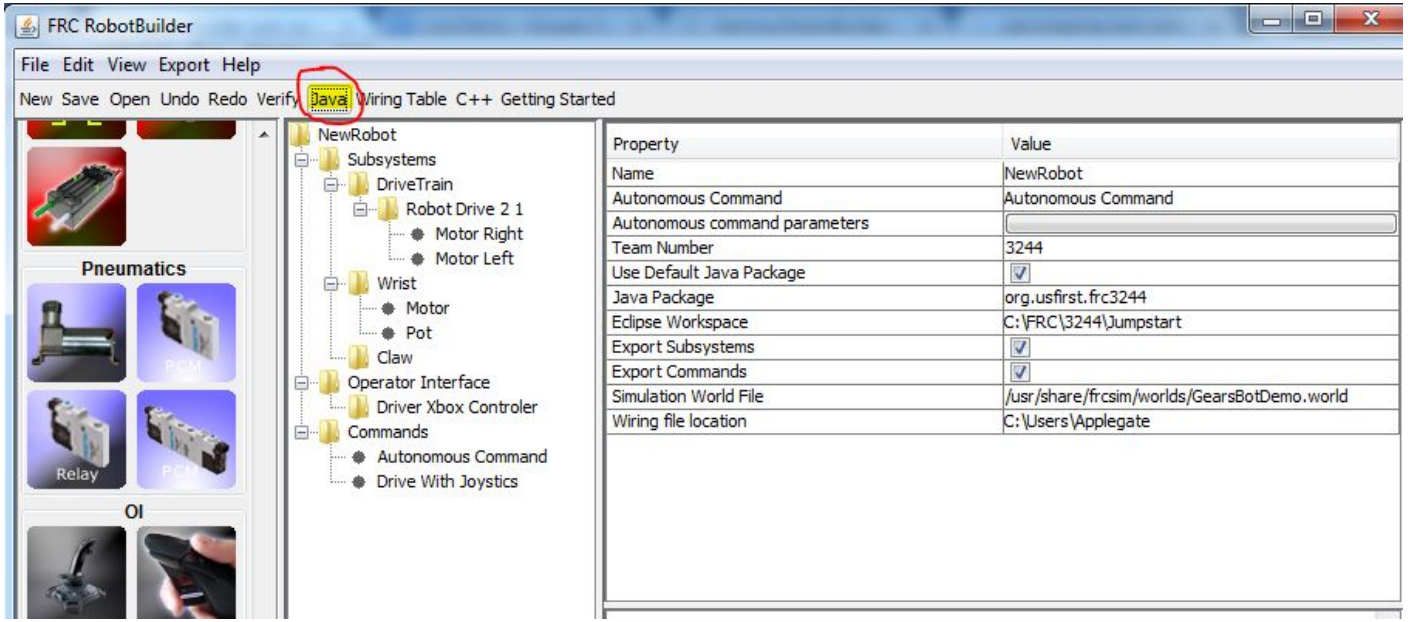
Team 3244

Robotbuilder & Command Based Robot

Live Demo

Save and generate Java Code

1. Save Project using standard windows steps
2. Make sure the Eclipse Workspace is set
3. Click Java Button



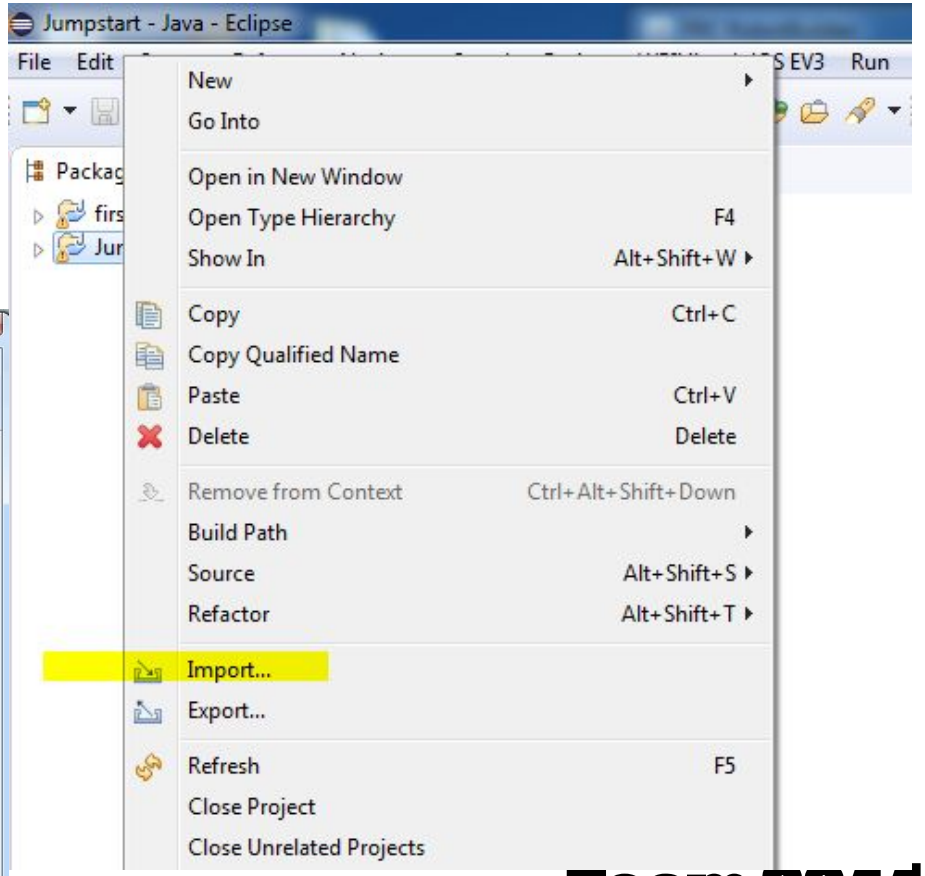
Team 3244

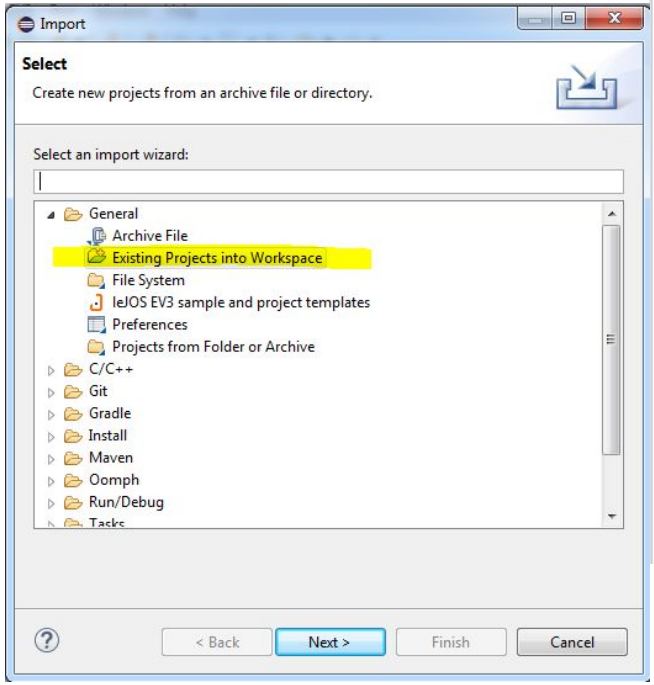
Robotbuilder & Command Based Robot

Live Demo

Import Robotbuilder code into Eclipse

1. Click “File” or right click “Package Explorer” and select Import…
2. Select “Existing Projects into Workspace” then “Next”





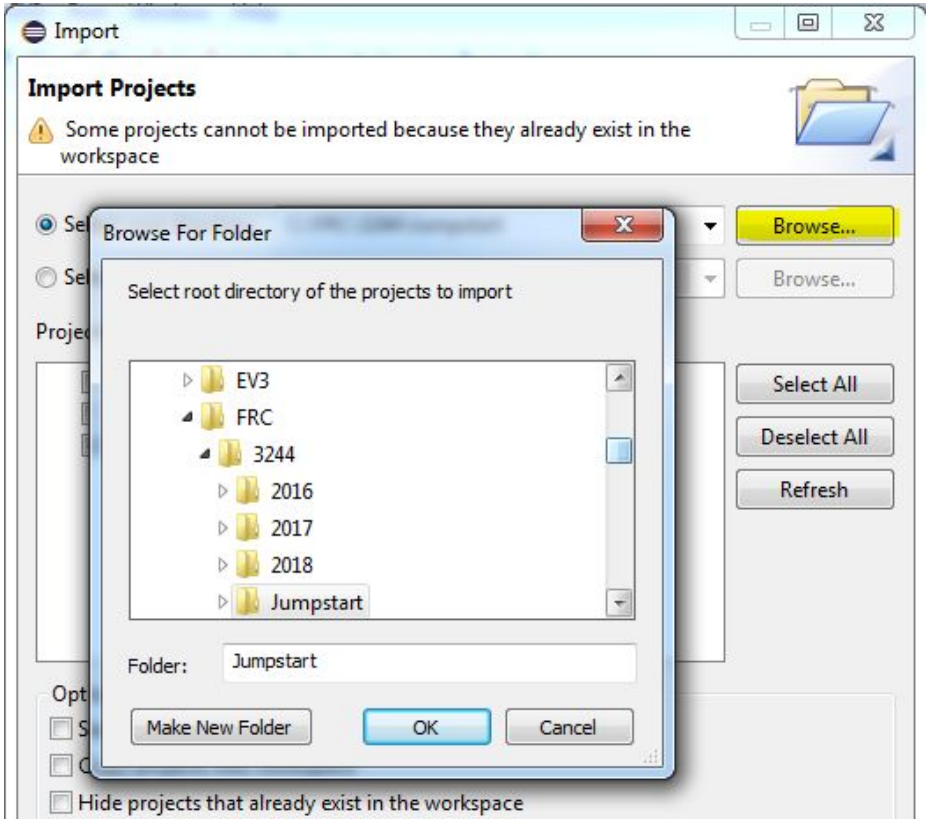
Team 3244

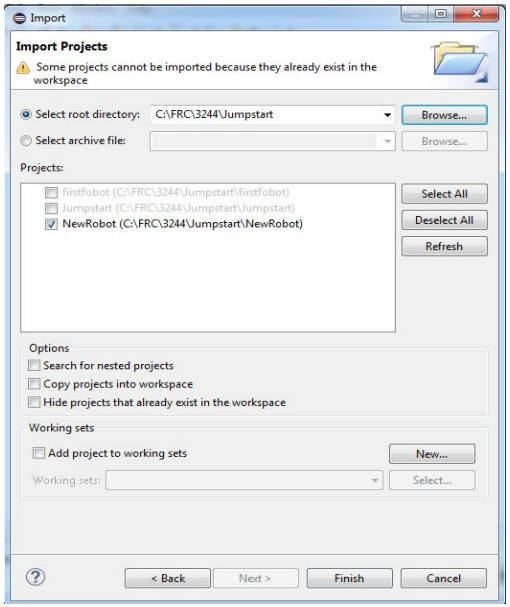
Robotbuilder & Command Based Robot

Live Demo

Import Robotbuilder code into Eclipse

3. Browse for the Eclipse Workspace set in Robotbuilder. This should the current folder and already selected. Click “OK”
4. Check the Project and click “Finish”





Team 3244

Robotbuilder & Command Based Robot

Live Demo

Imported Code

1. Imported Code nicely organized
   a. NewRobot
   b. NewRobot.Commands
   c. NewRobot.Subsystems
2. In the classes there are comments like

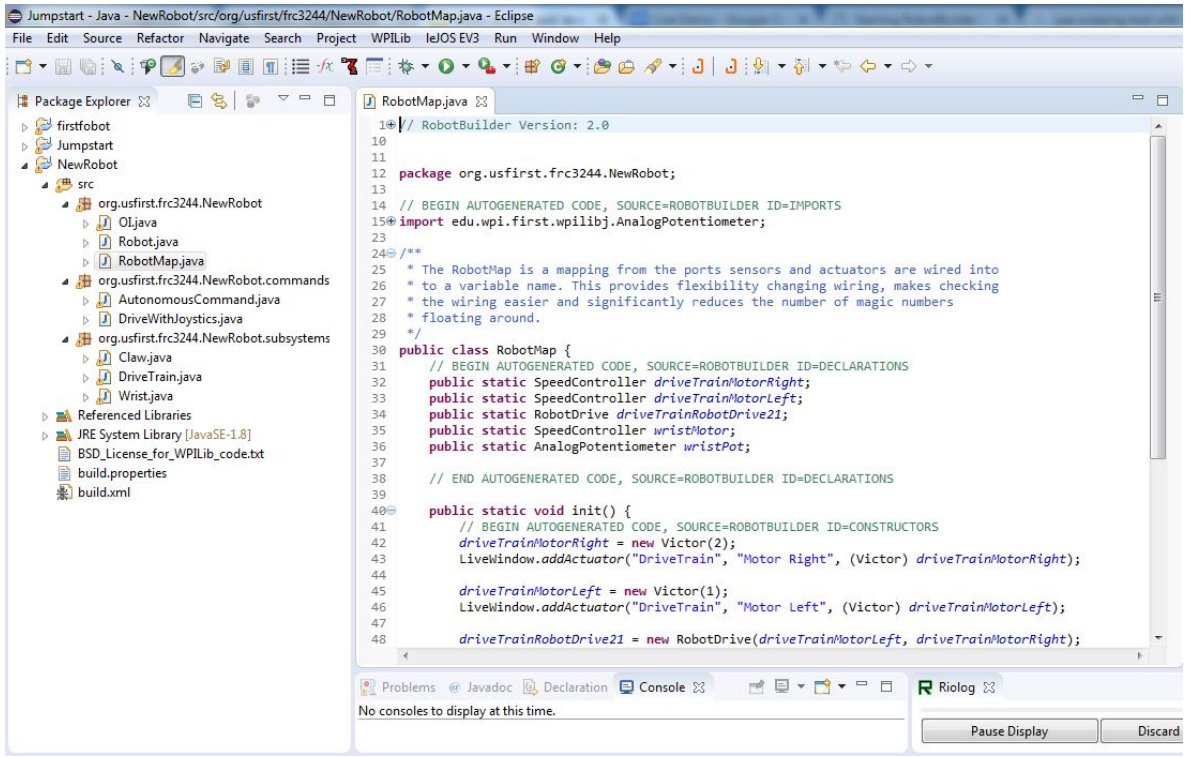
   // BEGIN AUTOGENERATED CODE

   …

   …

   // END AUTOGENERATED CODE

   These areas are code generated by Robotbuilder and will be overwritten each time we make changes in Robotbuilder.



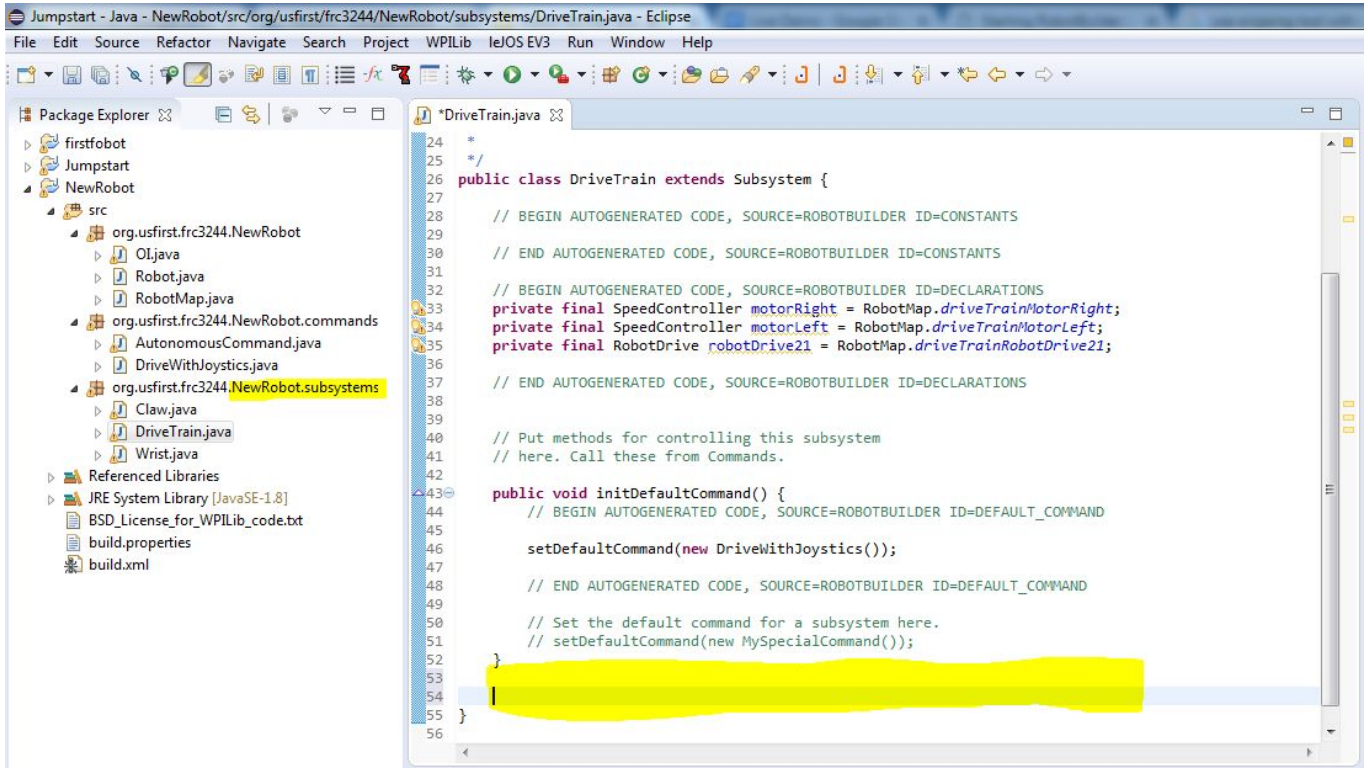
Team 3244

Robotbuilder & Command Based Robot

Live Demo

Subsystems

1. At the top are the objects available for this subsystem to control.
2. A Method created for us to set the default command
3. Highlighted area will be where we write our methods this subsystem can do, get, and set methods for Data.



Team 3244

Robotbuilder & Command Based Robot

Live Demo

Commands

This will tell our Subsytems what to do.

1. Command Constructor Sets the Requires

2. initialize() Called just before this Command runs the first time

3. execute()Called repeatedly when this Command is scheduled to run
   a. Most of our code will go here

4. isFinished()Make this return true when this Command no longer needs to run execute()

5. end()Called once after isFinished returns true

6. interrupted()Called when another command which requires one or more of the sam subsystems is scheduled to run

*DriveTrain.java | *DriveWithJoystics.java

```java
public class DriveWithJoystics extends Command {

    public DriveWithJoystics() {

    // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=CONSTRUCTOR
        // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=VARIABLE_SETTING
        // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=VARIABLE_SETTING

        // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=REQUIRES
        requires(Robot.driveTrain);
        // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=REQUIRES
    }
    // Called just before this Command runs the first time
    protected void initialize() {
    }

    // Called repeatedly when this Command is scheduled to run
    protected void execute() {
    }

    // Make this return true when this Command no longer needs to run execute()
    protected boolean isFinished() {
        return false;
    }

    // Called once after isFinished returns true
    protected void end() {
    }

    // Called when another command which requires one or more of the same
    // subsystems is scheduled to run
    protected void interrupted() {
    }
}
```

Team 3244

Robotbuilder & Command Based Robot

Live Demo

Robot Map

1. Container that maps the port numbers all PWM, DIO, Relay, Analog, and other components connected to the robot.

*DriveTrain.java | *DriveWithJoystics.java | RobotMap.java

```java
public class RobotMap {
    // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=DECLARATIONS
    public static SpeedController driveTrainMotorRight;
    public static SpeedController driveTrainMotorLeft;
    public static RobotDrive driveTrainRobotDrive21;
    public static SpeedController wristMotor;
    public static AnalogPotentiometer wristPot;

    // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=DECLARATIONS

    public static void init() {
        // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=CONSTRUCTORS
        driveTrainMotorRight = new Victor(2);
        LiveWindow.addActuator("DriveTrain", "Motor Right", (Victor) driveTrainMotorRight);

        driveTrainMotorLeft = new Victor(1);
        LiveWindow.addActuator("DriveTrain", "Motor Left", (Victor) driveTrainMotorLeft);

        driveTrainRobotDrive21 = new RobotDrive(driveTrainMotorLeft, driveTrainMotorRight);

        driveTrainRobotDrive21.setSafetyEnabled(true);
        driveTrainRobotDrive21.setExpiration(0.1);
        driveTrainRobotDrive21.setSensitivity(0.5);
        driveTrainRobotDrive21.setMaxOutput(1.0);

        wristMotor = new VictorSP(0);
        LiveWindow.addActuator("Wrist", "Motor", (VictorSP) wristMotor);

        wristPot = new AnalogPotentiometer(0, 1.0, 0.0);
        LiveWindow.addSensor("Wrist", "Pot", wristPot);


        // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=CONSTRUCTORS
    }
```

Team 3244

Robotbuilder & Command Based Robot

Live Demo

OI

1. Container that controls all the Operator Interface devices like Joysticks, Buttons, and SmartDashboard Buttons.

*DriveTrain.java | *DriveWithJoystics.java | RobotMap.java | OI.java

```
47
48      // Start the command when the button is released  and let it run the command
49      // until it is finished as determined by it's isFinished method.
50      // button.whenReleased(new ExampleCommand());
51
52
53      // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=DECLARATIONS
54      public Joystick driverXboxControler;
55
56      // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=DECLARATIONS
57
58      public OI() {
59          // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=CONSTRUCTORS
60
61          driverXboxControler = new Joystick(0);
62
63
64
65          // SmartDashboard Buttons
66          SmartDashboard.putData("Autonomous Command", new AutonomousCommand());
67
68          // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=CONSTRUCTORS
69      }
70
71      // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=FUNCTIONS
72      public Joystick getDriverXboxControler() {
73          return driverXboxControler;
74      }
75
76
77      // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=FUNCTIONS
78  }
79
80
```

Team 3244

Robotbuilder & Command Based Robot

Live Demo

1. Container with everything related to the robot and the modes the robot will transition through when enabled.
   **Init runs once before Periodic runs*
2. robotInit() initialization code
3. disabledInit(), disabledPeriodic() code run when robot disabled
4. autonomousInit(), autonomousPeriodic() code run when robot is autonomous
5. teleopInit(), teleopPeriodic() code run when robot is teleop
6. testPeriodic() code run when robot in testMode

Robot

*DriveTrain.java | *DriveWithJoystics.java | RobotMap.java | OI.java | Robot.java

```java
public class Robot extends IterativeRobot {

    Command autonomousCommand;

    public static OI oi;
    // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=DECLARATIONS
    public static DriveTrain driveTrain;
    public static Wrist wrist;
    public static Claw claw;

    // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=DECLARATIONS

    /**
     * This function is run when the robot is first started up and should be
     * used for any initialization code.
     */
    public void robotInit() {
    RobotMap.init();
        // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=CONSTRUCTORS
        driveTrain = new DriveTrain();
        wrist = new Wrist();
        claw = new Claw();

        // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=CONSTRUCTORS
        // OI must be constructed after subsystems. If the OI creates Commands
        //(which it very likely will), subsystems are not guaranteed to be
        // constructed yet. Thus, their requires() statements may grab null
        // pointers. Bad news. Don't move it.
        oi = new OI();

        // instantiate the command used for the autonomous period
        // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=AUTONOMOUS

        autonomousCommand = new AutonomousCommand();
```

Team 3244

Robotbuilder & Command Based Robot

Live Demo

Download Code to Robot *Test Mode

1. ToDo Test Mode

Robotbuilder & Command Based Robot

Live Demo

Code DriveTrain and Joysticks

Subsystem Code

1. Added Method “myArcadeDrive” This is the action this subsystem can do.
2. The method accepts two parameters to control the speed and direction of the robot
   a. moveValue
   b. rotateValue

DriveTrain.java

```java
 *
 */
public class DriveTrain extends Subsystem {

    // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=CONSTANTS

    // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=CONSTANTS

    // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=DECLARATIONS
    private final SpeedController motorRight = RobotMap.driveTrainMotorRight;
    private final SpeedController motorLeft = RobotMap.driveTrainMotorLeft;
    private final RobotDrive robotDrive21 = RobotMap.driveTrainRobotDrive21;

    // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=DECLARATIONS


    // Put methods for controlling this subsystem
    // here. Call these from Commands.

    public void initDefaultCommand() {
        // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=DEFAULT_COMMAND

        setDefaultCommand(new DriveWithJoystics());

        // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=DEFAULT_COMMAND

        // Set the default command for a subsystem here.
        // setDefaultCommand(new MySpecialCommand());
    }

    public void myArcadeDrive(double moveValue, double rotateValue){
        robotDrive21.arcadeDrive(moveValue, rotateValue);
    }
}
```

Team 3244

Robotbuilder & Command Based Robot

Live Demo

Code DriveTrain and Joysticks

Command Code

1. execute() Called repeatedly
   a. two variables to store Joystick values from the Robot.oi
   b. Send Joystick values to the Robot.driveTrain.myArcade Drive method
2. isFinished() never complete
3. end()
   a. Turn off motors if this command ends
4. interrupted()
   a. Call end() if a new command requires driveTrain

DriveTrain.java | *DriveWithJoystics.java

```java
        // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=VARIABLE_SETTING
        // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=VARIABLE_SETTING

        // BEGIN AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=REQUIRES
        requires(Robot.driveTrain);
        // END AUTOGENERATED CODE, SOURCE=ROBOTBUILDER ID=REQUIRES
    }
    // Called just before this Command runs the first time
    protected void initialize() {
    }

    // Called repeatedly when this Command is scheduled to run
    protected void execute() {
        double moveValue = Robot.oi.driverXboxControler.getRawAxis(0);
        double rotateValue = Robot.oi.driverXboxControler.getRawAxis(3);
        Robot.driveTrain.myArcadeDrive(moveValue, rotateValue);
    }

    // Make this return true when this Command no longer needs to run execute()
    protected boolean isFinished() {
        return false;
    }

    // Called once after isFinished returns true
    protected void end() {
        Robot.driveTrain.myArcadeDrive(0, 0);
    }

    // Called when another command which requires one or more of the same
    // subsystems is scheduled to run
    protected void interrupted() {
        end();
    }
}
```

Team 3244

Robotbuilder & Command Based Robot

Live Demo

Tune PID Using Test Mode

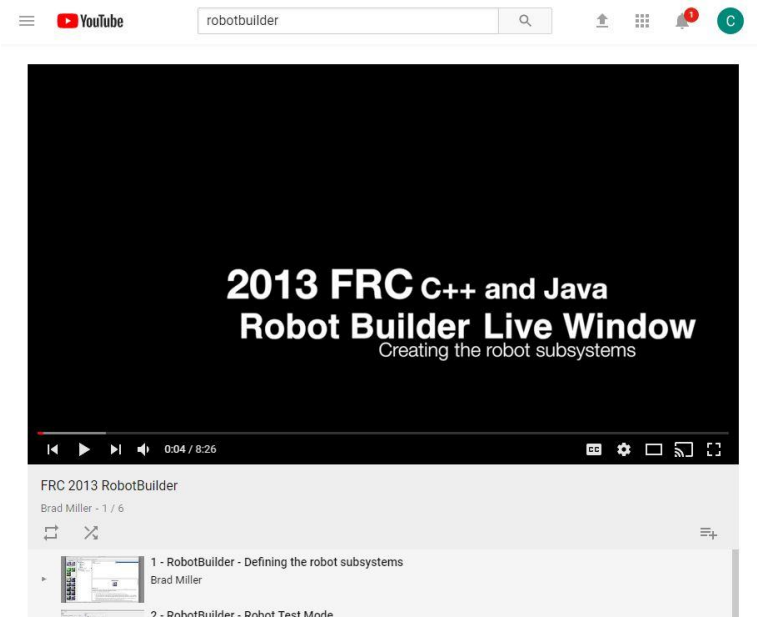
1. ToDo

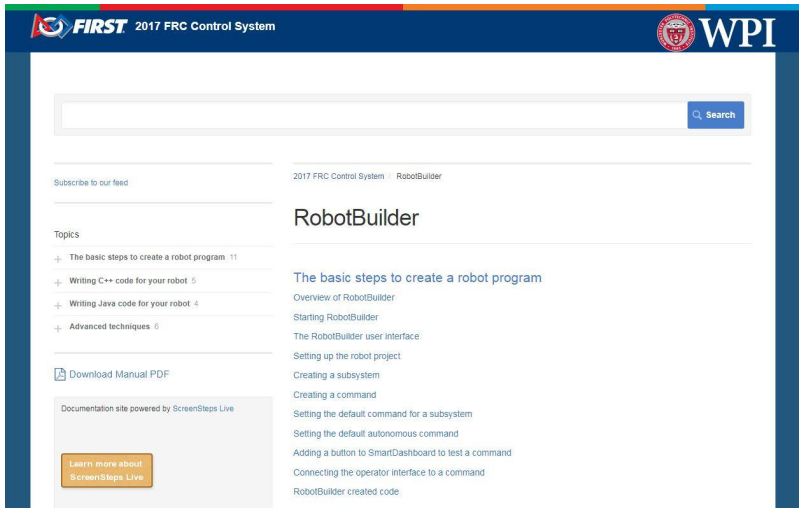
Team 3244

Robotbuilder & Command Based Robot

Find More Information on Robotbuilder

YouTube Brad Miller from WPI created 6 videos in 2013 that are still a great video series.
Search [FRC 2013 Robotbuilder]



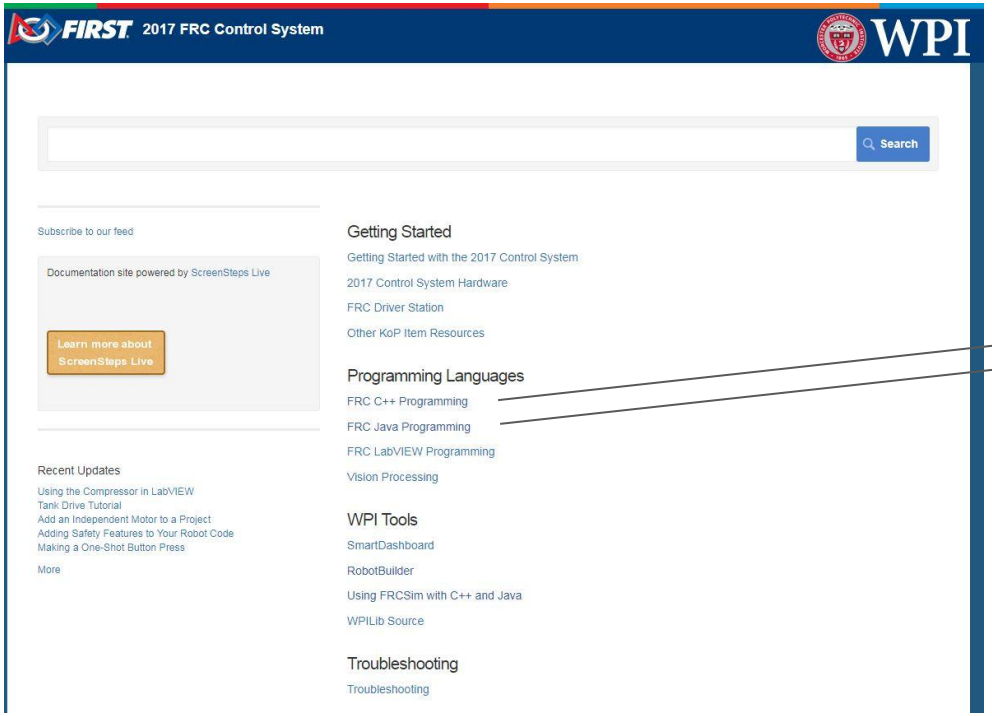


Everyone should be familiar with
[wpilib.screenstepslive.com]

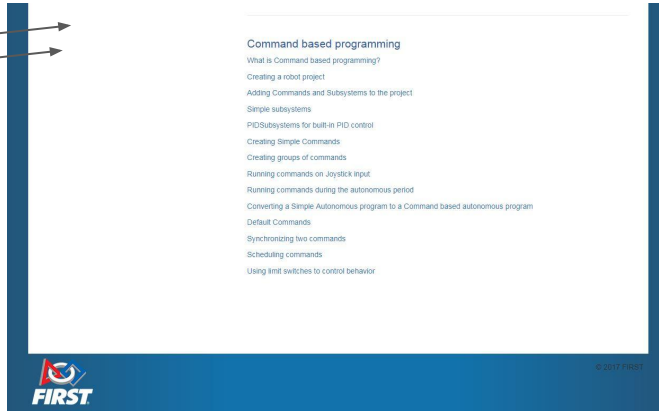
Team 3244

Robotbuilder & Command Based Robot

Find More Information On Command Based Programing



At the bottom of both the C++ and Java Programing Languages will be a Section for Command Based Programing



Team 3244

Questions

Team 3244