D.Frederick

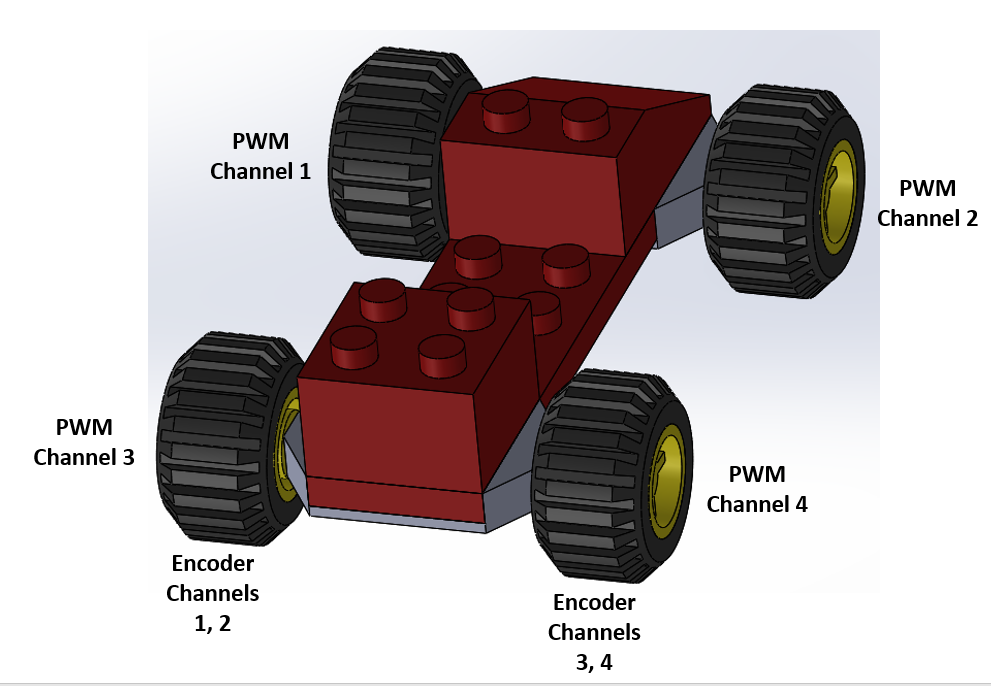
11/30/2016 4:47:00 AM

df20151026\_Robotics\_Software\_Development\_processs\_using\_Eclipse\_and\_RobotBuilder\_v01.docx

**Purpose:**

This document describes how to build a simple robot using RobotBuilder Tool and Eclipse.

Reference:



RobotBuider Instructions:

<https://wpilib.screenstepslive.com/s/4485>

**High Level Goal:**

Create a basic robot that can be driven with a Joystick.

**Assumptions:**

Eclipse is installed and has the WPI Add-in

**Robot Design:**

Four motors controllers:

Where: PWM1=Front Left Wheel PWM2=Front Right Wheel,

PWM3=Rear Left Wheel PWM4=Rear Right Wheel

Encoders: Left side Channels 1 and 2

Right side Channels 3 and 4

Gyro: Analog Channel 0

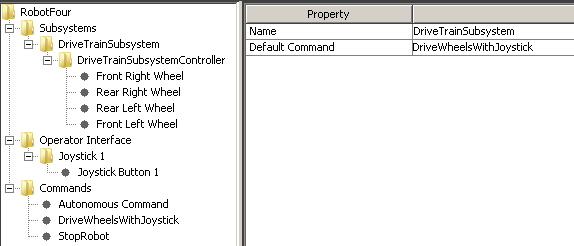
Joystick 1

**Implementation:**

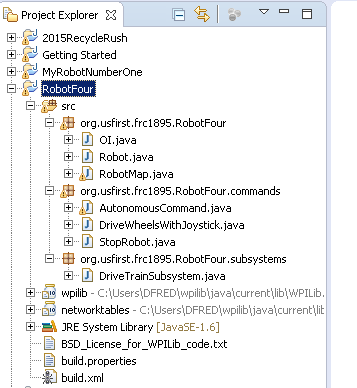
1. Create a folder to store the RobotBuilder project such as “**C:\\_Robotics\2016\Eclipse\_Projects**”.
2. Start Eclipse.
3. Start RobotBuilder using the command sequence **Eclipse** => **WPILib** => **Run RobotBuilder**
4. Select **File** => **New**. On Creation of new Robot Project, set:
   * Project Name: e.g.: **RobotFour**
   * Team Number: e.g.: **1895**

Select **Create Project**

1. In Robot Project, set:
   * Eclipse Workspace: e.g.: “**C:\\_Robotics\2016\Eclipse\_Projects**”
   * Wiring File: e.g.: “**C:\\_Robotics\2016\Eclipse\_Projects\RobotFourWires**”
2. Save the RobotBuilder Project file:
   * Select **RobotBuilder** => **Save As** => “**C:\\_Robotics\2016\Eclipse\_Projects\RobotFour**"
3. Create a drivetrain "**Subsystem**", Right click subsystem and select "**Add Subsystem**"
4. On the right side of the page, rename the subsystem to "**DriveTrainSubsystem**"
5. Add a "**Controller**" by right clicking on the subsystem "**DriveTrainSubsystem**" and selecting "**Add Controller**" then select "**Add Robot Drive 4**".
6. Rename "**Drive Train 4 1**" to "**DriveTrainSubsystemController**". *(Ignore the warnings at this time)*
7. Add four motors to the Drive Train by Right clicking on the "**DriveTrainSubsystemController**" and selecting "**Add Speed Controller**". Perform this four times.
8. Rename "**Speed Controller 1**" to "**Front Left Wheel**"
9. Rename "**Speed Controller 2**" to "**Front Right Wheel**"
10. Rename "**Speed Controller 3**" to "**Rear Left Wheel**"
11. Rename "**Speed Controller 4**" to "**Rear Right Wheel**"
12. Change the "**Front Left Wheel**" type to **Jaguar**, Set **Output Channel (PWM)** to the value of **1**.
13. Change the "**Front Right Wheel**" type to **Jaguar**, Set **Output Channel (PWM)** to the value of **2**.
14. Change the "**Rear Left Wheel**" type to **Jaguar**, Set **Output Channel (PWM)** to the value of **3**.
15. Change the "**Rear Right Wheel**" type to **Jaguar**, Set **Output Channel (PWM)** to the value of **4**.
16. Update the "**DriveTrainSubsystemController**" **Left Front Motor** to "**Front Left Wheel**"
17. Update the "**DriveTrainSubsystemController**" **Left Rear Motor** to "**Rear Left Wheel**"
18. Update the "**DriveTrainSubsystemController**" **Right Front Motor** to "**Front Right Wheel**"
19. Update the "**DriveTrainSubsystemController**" **Right Rear Motor** to "**Rear Right Wheel**"
20. Save the RobotBuilder project. Select "**File**" the "**Save**".
21. Add a command to have the Joystick drive the wheels. Right click on "**Commands**" and select "**Add Command**".
22. Rename "**Command 1**" to "**DriveWheelsWithJoystick**"
23. Update the "**Requires**" field to "**DriveTrainSubsystem**".
24. Add a second command to stop the robot. Right click on "**Commands**" and select "**Add Command**".
25. Rename "Command 1" to "**StopRobot**"
26. Update the command "**StopRobot**" "**Requires**" field to "**DriveTrainSubsystem**"
27. Assign the **DriveTrainSubsystem** Default Command to "**DriveWheelsWithJoystick**", Select "**DriveTrainSubsystem**" and the update the field in the right window.
28. Add a Joystick. Right click on the **Operator Interface** and select "**add JoyStick**"
29. Add a Joystick Button. Right click on the newly created "**Joystick 1**" and select " **Add Joystick Button**"   
    *(Ignore the warning)*
30. Assign the Joystick button to stop the robot. Select **"Joystick Button 1**" and change the command to "**StopRobot**"
31. Save the Project.
32. Export the RobotBuilder project to Eclipse. Select **Export** => **Java**.
33. Close the RobotBuilder Application.



1. Import the Java Project into Eclipse
   * Select **Eclipse** => **File** => **Import** => **General** => **Existing Projects into Workspace,** Select **Next**
   * Select **Browse**. Browse to project folder: e.g.: “**C:\\_Robotics\2016\Eclipse\_Projects\RobotFour**”
   * Highlight **RobotFour** and select **OK**.
   * Select **Finish**.
2. Within Eclipse, expand the “src” (Source code) folder and subfolder. Double-click on the **DriveTrainSubsystem** class.



1. Within the "**DriveTrainSubSystem**" class (.java), add methods to control the wheels. Create a "**stop**" method by entering the following code below the *"****// Put methods for controlling this subsystem here. Call these from Commands.****"*:

public void stop (){

driveTrainSubsystemController.arcadeDrive(0, 0);

}

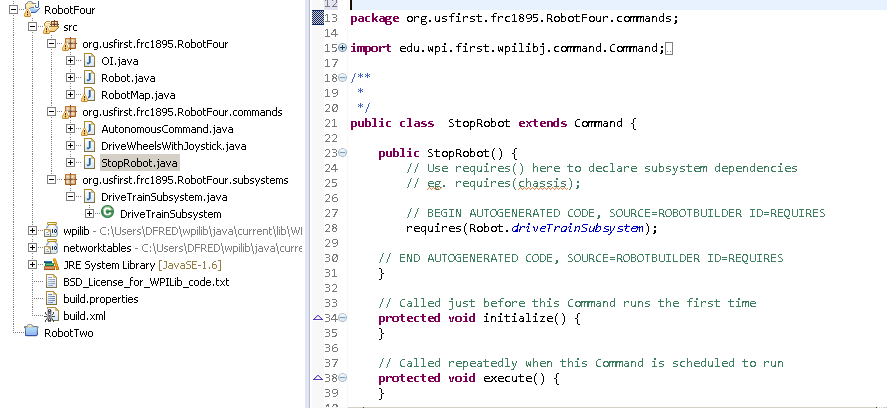
1. Within the "**DriveTrainSubSystem**" class (.java), add methods to control the wheels. Create a "arcade" method by entering the following code:

public void ArcadeDrive (double Yaxis, double Xaxis) {

driveTrainSubsystemController.arcadeDrive(Yaxis, Xaxis);

}

1. Open the "StopRobot" **command** class**.**



1. Link the Joystick button with the stop method in the "DriveTrainSubSystem". Open the "StopRobot" **command** and update the execute methods as follows.

protected void execute() {

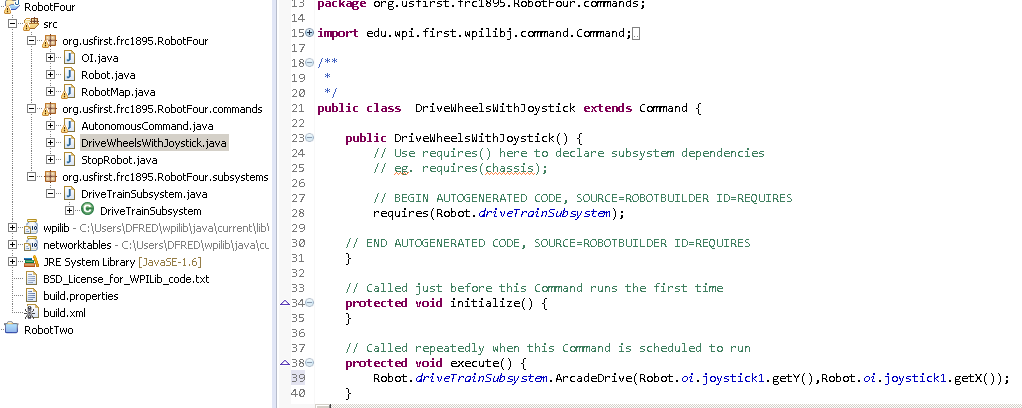
**Robot.driveTrainSubsystem.stop();**

}

1. Open the "DriveWheelsWithJoystick " command.
2. Link the Joystick X and Y motions with the arcadeDrive method in the "DriveTrainSubSystem". Update the execute methods as follows.

protected void execute() { Robot.driveTrainSubsystem.ArcadeDrive(Robot.oi.joystick1.getY(),Robot.oi.joystick1.getX());

}



1. Connect the Workstation to the Robot using a network cable and power on the Robot.   
   **(THINK SAFETY: Be sure everyone is clear of the Robot)**
2. Download the code into the RoboRIO. Select **Run**, then **Run,** then **WPILib JAVA Deploy,** then **OK**
3. Start the FRC Drivers Station.
4. Select **TeleOperated** Mode
5. Select **Enable** to start the Robot.

**(THINK SAFETY: Be sure everyone is clear of the Robot)**

1. Test the code by moving the joysticks.

**Expand the Project: (Add a Gyro and Shaft Encoders)**

1. Open the RobotBuilder Application.
2. Highlight the **DriveTrainSubsystem**, Right-click and select **Add Sensors**.
3. Select **Add Analog Gyro**, Update the name to **gryo**.
4. Verify the **Input Channel (Analog)** is set to **0**.
5. Highlight the DriveTrainSubsystem, Right-click and select **Add Sensors**.
6. Select **Add Quadrature Encoder**.
7. Rename **Quadrature Encoder 1** to **LeftEncoder**.
8. Set **Channel A Channel (Digital)** to a value of **1**.
9. Set **Channel B Channel (Digital)** to a value of **2**.
10. Select **Add Quadrature Encoder**.
11. Rename **Quadrature Encoder 1** to **RightEncoder**.
12. Set **Channel A Channel (Digital)** to a value of **3**.
13. Set **Channel B Channel (Digital)** to a value of **4**.
14. Save the Project.
15. Export the RobotBuilder project to Eclipse. Select **Export** => **Java**.
16. Close the RobotBuilder Application.
17. In Eclipse, refresh the project. In the Package Explorer window, select the class **Robot.java** then select **File => Refresh** or press **F5**. The new subsystem and command should appear in the Eclipse workspace.
18. In the DriveTrainSubSystem Class, add methods to read and print the Gyro and Encoders BEFORE the stop method.

public void printGyro() {

System.out.println ("Gyro: " + gyro.getAngle());

}

public void printEncoders() {

System.out.println ("Encoders: " + leftEncoder.getDistance() + " " + rightEncoder.getDistance());

}

1. Within the ArcadeDrive method, call the gyro and encoder print methods by adding the following lines after the arcadeDrive.

printGyro();

printEncoders();

1. Save the Eclipse files. Select "**File**" then "**Save All**".
2. Download the code into the RoboRIO. Select **Run**, then **Run,** then **WPILib JAVA Deploy,** then **OK**.
3. Bring up the RoboRio Console to see the output of the print statements.   
    ( Windows -> Show View -> Other -> General -> RioLog )
4. Select **TeleOperated** Mode
5. Select **Enable** to start the Robot.

**Resources:**

WPI Library JAVA Docs

http://first.wpi.edu/FRC/roborio/release/docs/java/

**Appendix – A One Time setup of Eclipse:**

* Initial Configurations (One Time)
  + Set Team Number
    - **Eclipse** => **Windows** => **Preferences** => **WPI Lib Preferences**  = **Team Number**
  + Configure Eclipse to sync with RobotBuilder
    - Updates in RobotBuilder are automatically added to the Eclipse Project
    - **Eclipse** => **Windows** => **Preferences** => **General** => **Workspace** = Enable Refresh using Native hooks or Polling
  + Display Console Window
    - **Eclipse** => **Window** => **Show View** => **Other** ... => **General** => **RioLog**
  + Create Workspace in Eclipse to hold Robot Project
    - **Eclipse** => **File** => **New** => **Project**… => WPILib Robot Java Development => Example Robot Java Project  
      => Getting Started with Java => Getting Started => Finish

**Appendix – B FRCSIM**

**Resources:**

https://wpilib.screenstepslive.com/s/4485/m/23353

<http://first.wpi.edu/FRC/roborio/release/simulation/>

**Limitations:**

1. The FRCSIM can take a few tries to start.
2. Joystick methods are not working correctly (getX(), getY() and getZ() )

Use: Robot.oi.joystick1.getRawAxis(0)

// Robot.driveTrainSubSystem.ArcadeDrive(Robot.oi.joystick1.getY(),Robot.oi.joystick1.getX());

Robot.driveTrainSubSystem.ArcadeDrive(Robot.oi.joystick1.getRawAxis(0), Robot.oi.joystick1.getRawAxis(1));

Notes: Joystick

- Axis 0 - Left(-) and Right(+)

- Axis 1 - forward (-) and Back (+)

- Axis 2 - Rotate forward (-) and Back (+)

Seems like getX(), getY() and getZ() are not working - wrong order

1. The Analog Gyro does not support the “setSensitivity” method. Comment out in RoboMap.
2. Needed to rename a few sensors (one Time).

/home/robot/wpilib/simulation/plugins/

cp libencoder.so libgz\_encoder.so

cp libgyro.so libgz\_gyro.so