

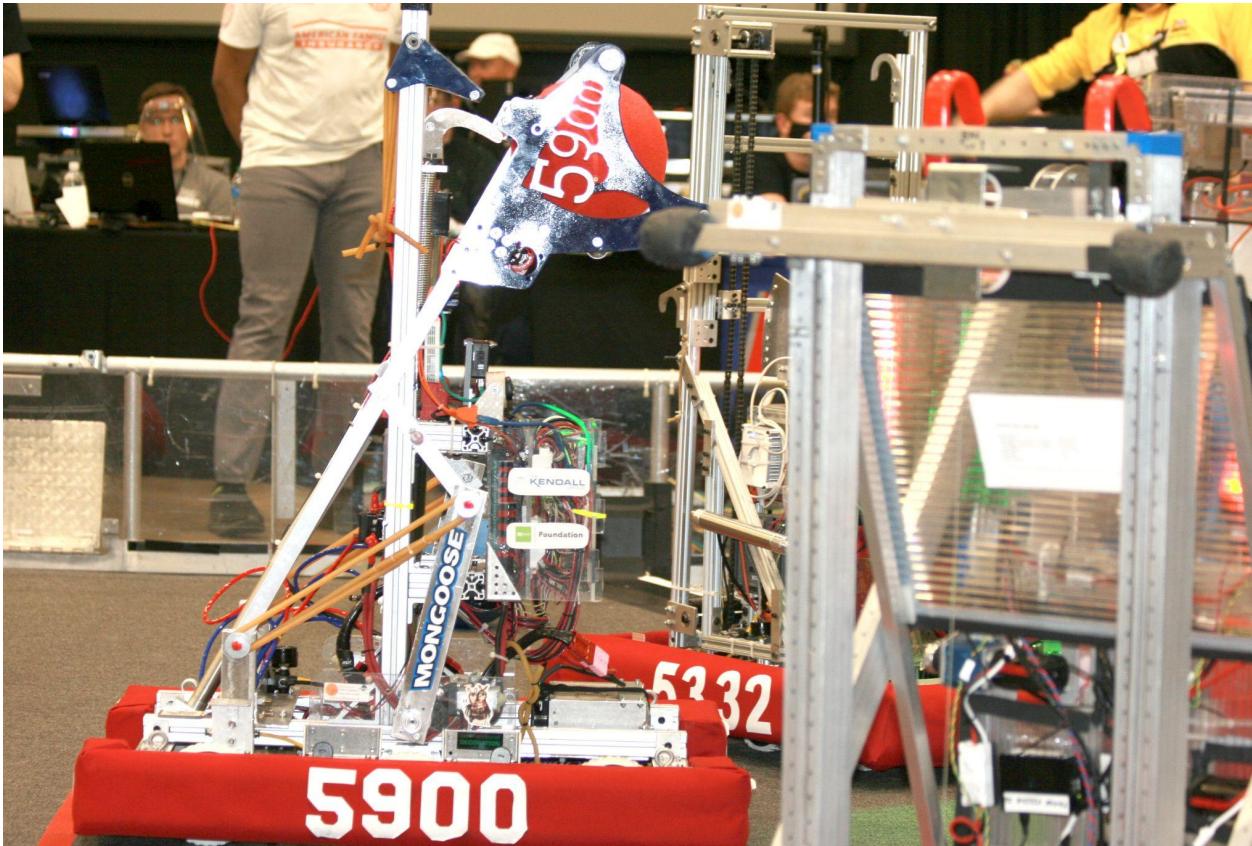
# RAPID REACT 2022

# “MISS-ALIGNED”

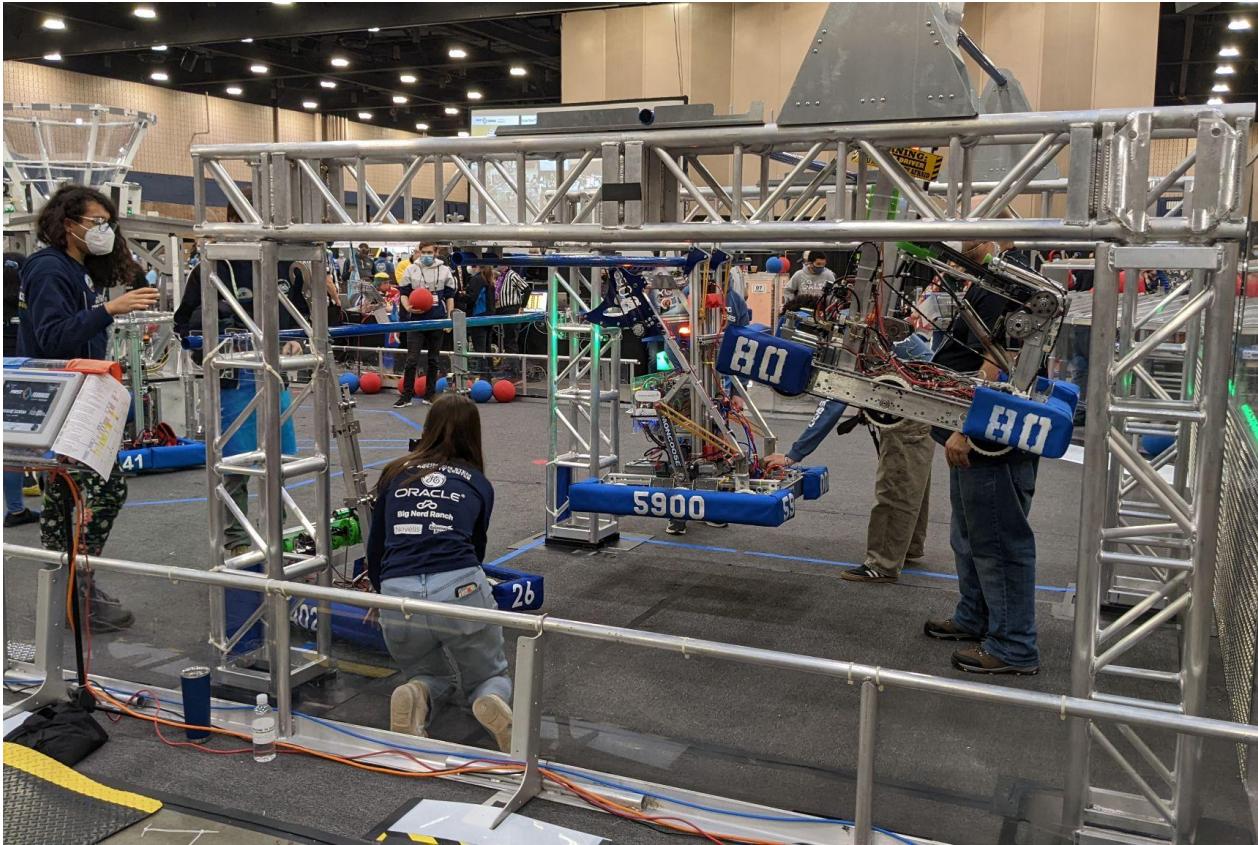


JLAS MANGOSTAS LUCHADORAS  
TEAM 5900 PROGRAMMING MANUAL  
LAST UPDATE - FEBRUARY 27, 2022

# "MISS-ALIGNED"

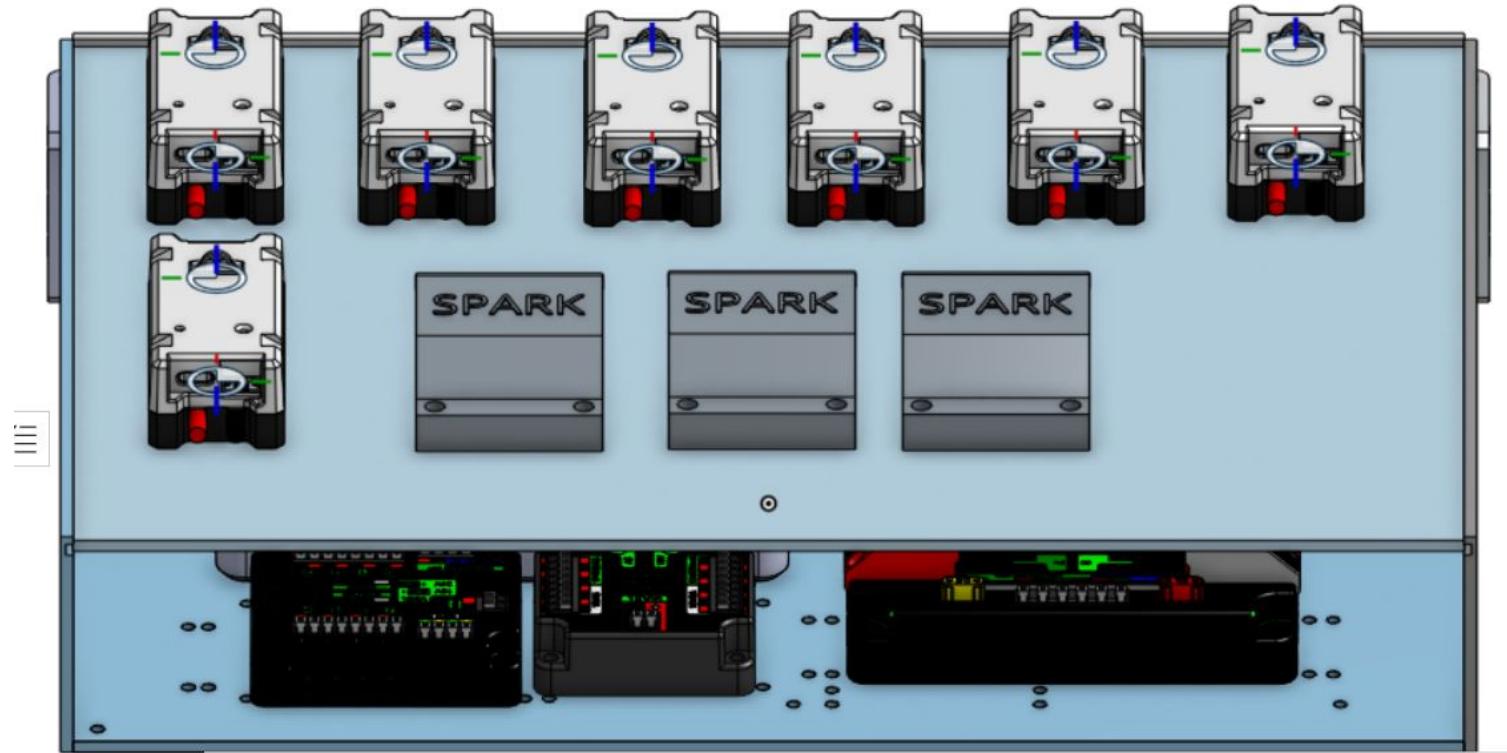


# "MISS-ALIGNED"

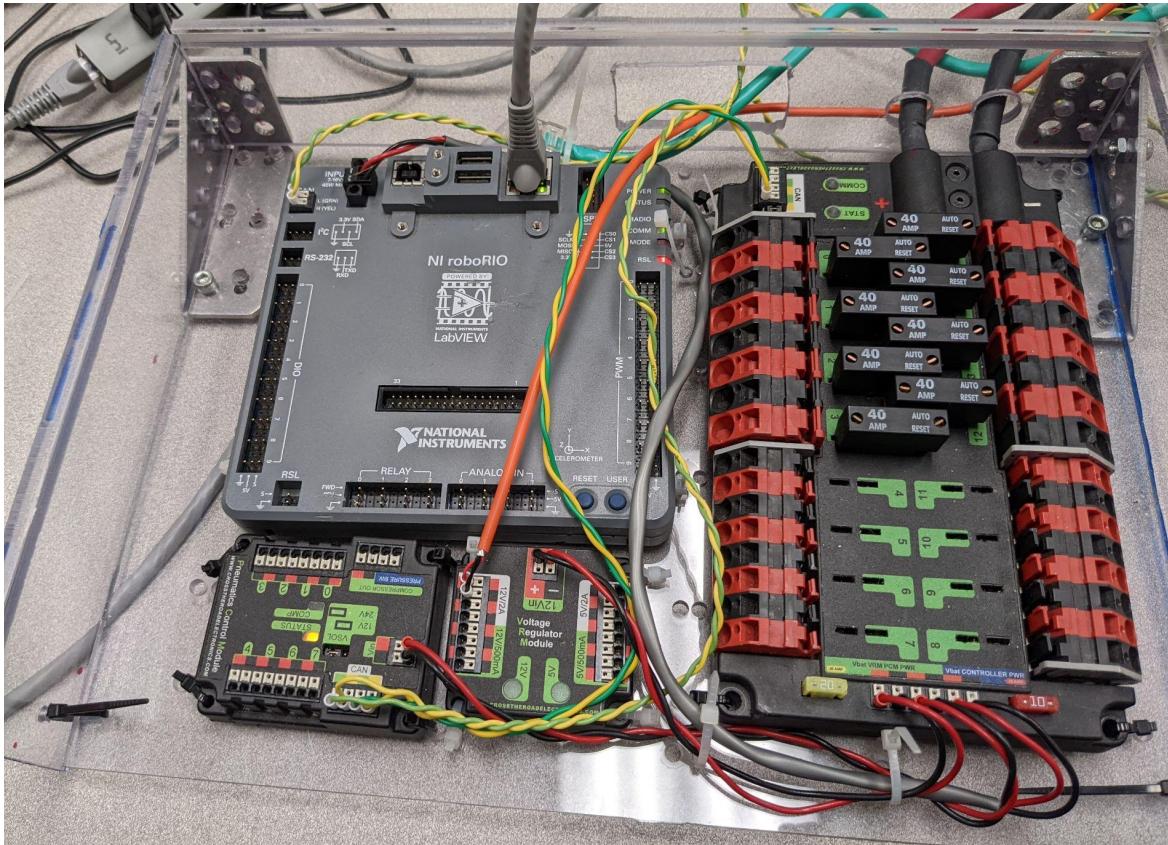


# CONTROLS

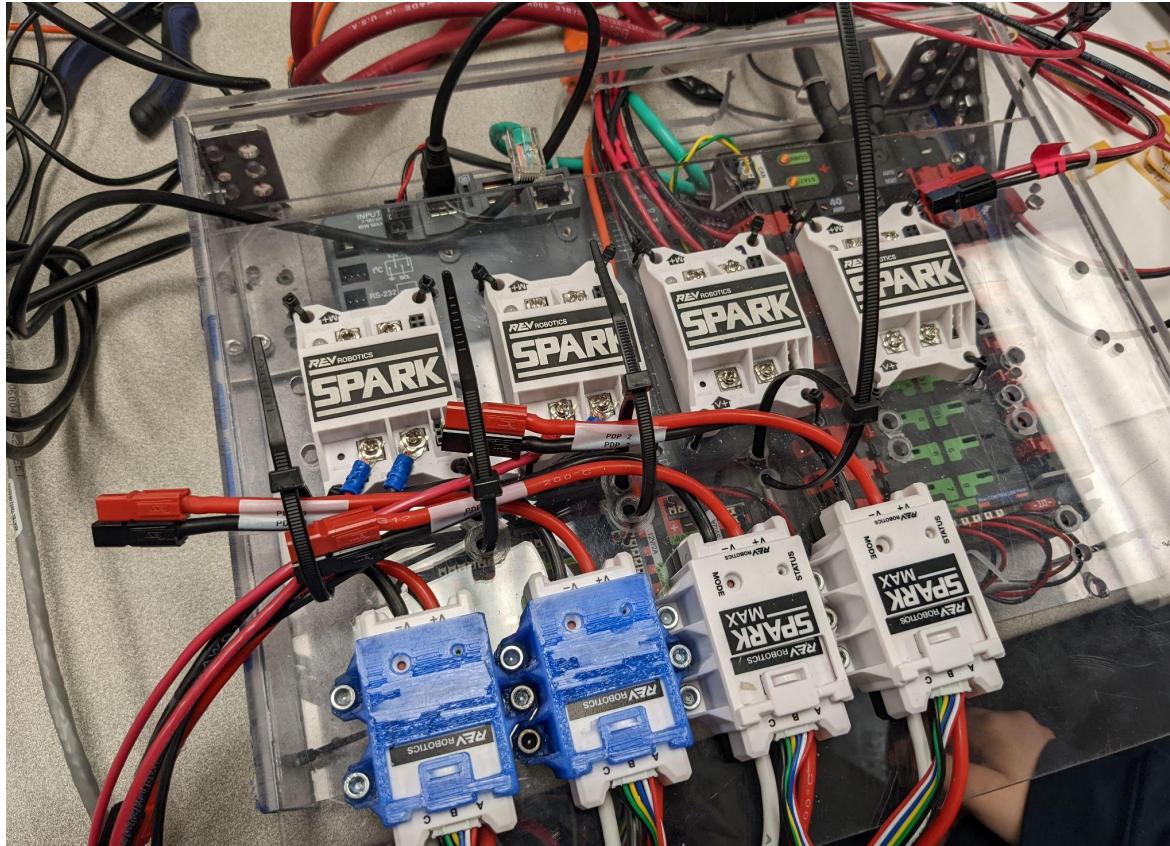
# CONTROLS ENCLOSURE



# CONTROLS ENCLOSURE - ROBORIO / PDP PANEL

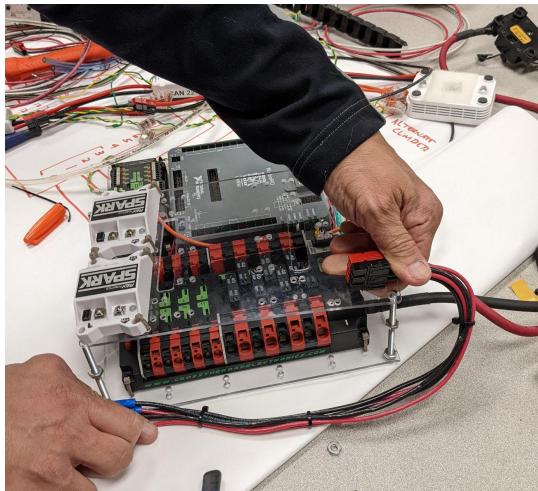


# CONTROLS ENCLOSURE - TOP PANEL FOR SPARKMAX AND SPARK DRIVES

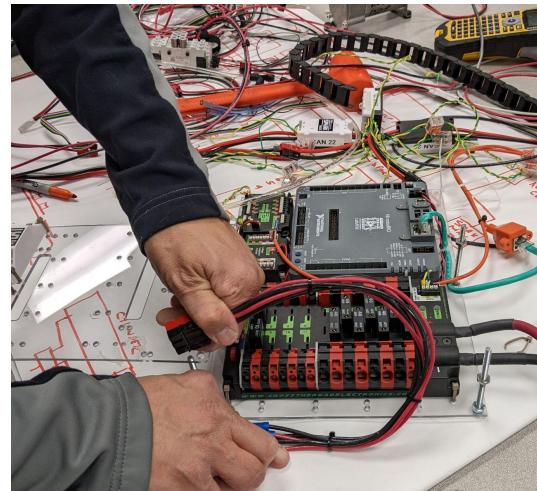


# CONTROLS ENCLOSURE CABLE MOVEMENT

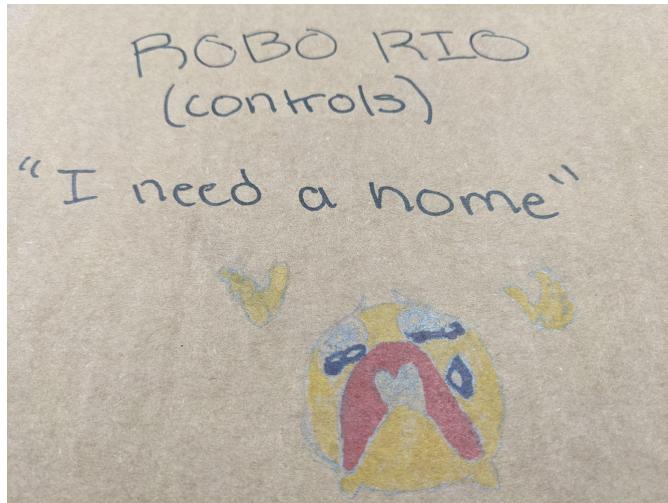
SHELF IN POSITION



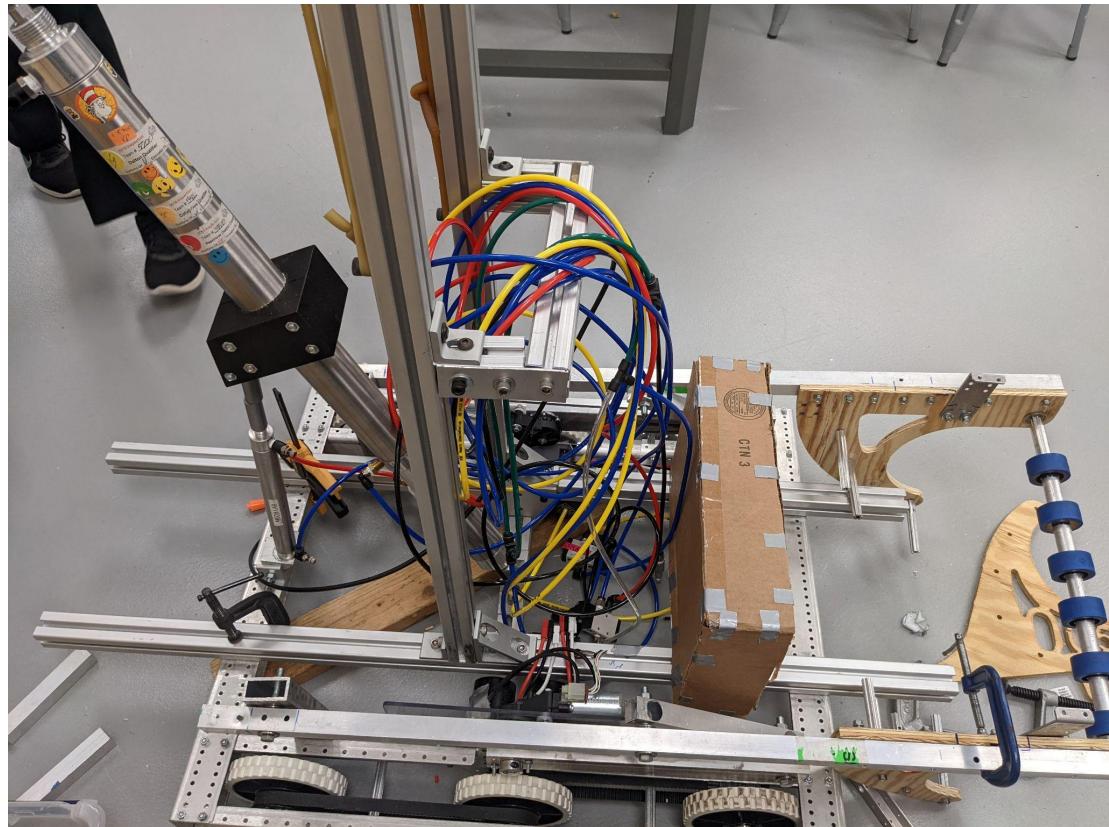
SHELF SLIDING OUT



# POSSIBLE HOME FOR THE RIO

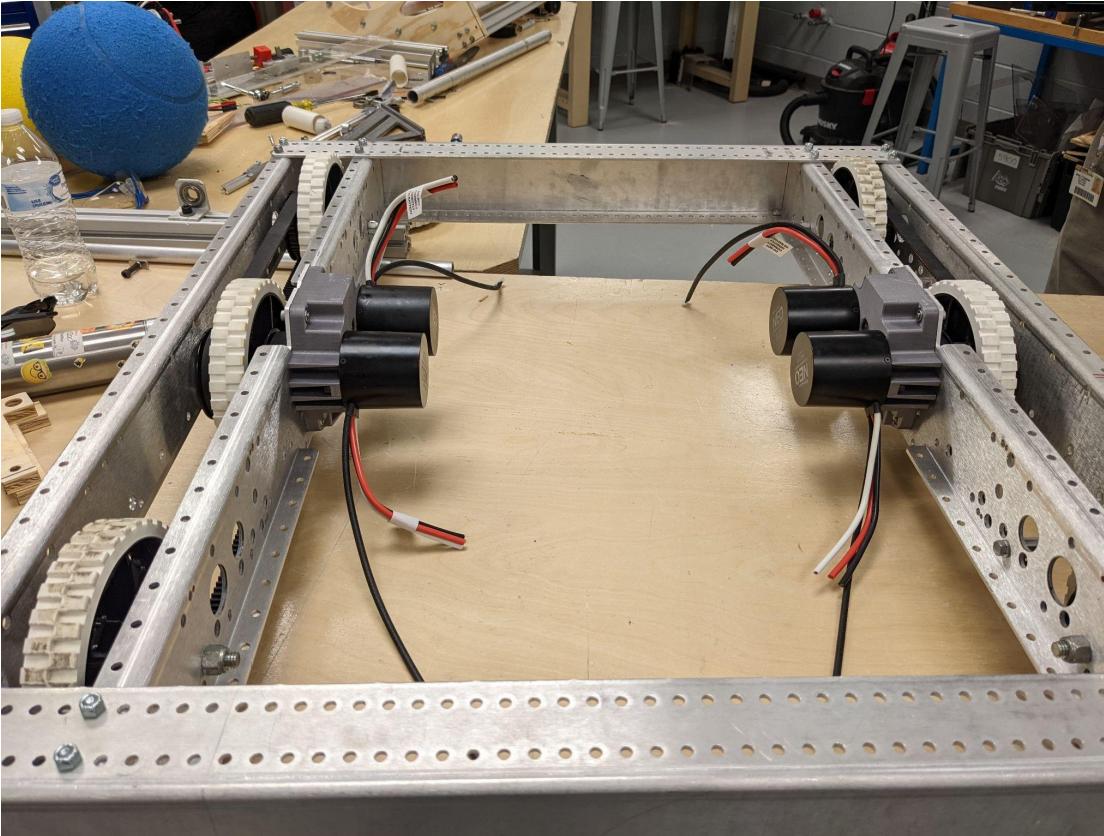


TRY TO KEEP BATTERY AT CENTER OF THE  
ROBOT AS LOW TO GROUND AS POSSIBLE  
FOR STABILITY, BUT EASILY ACCESSIBLE

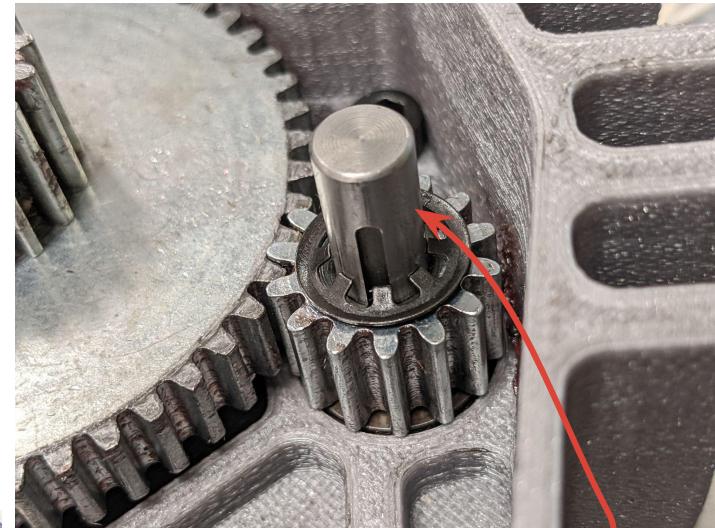
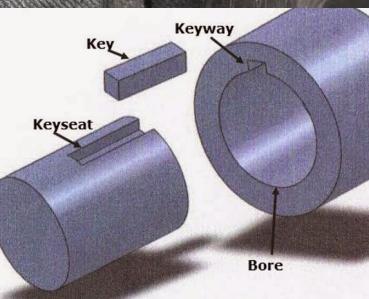
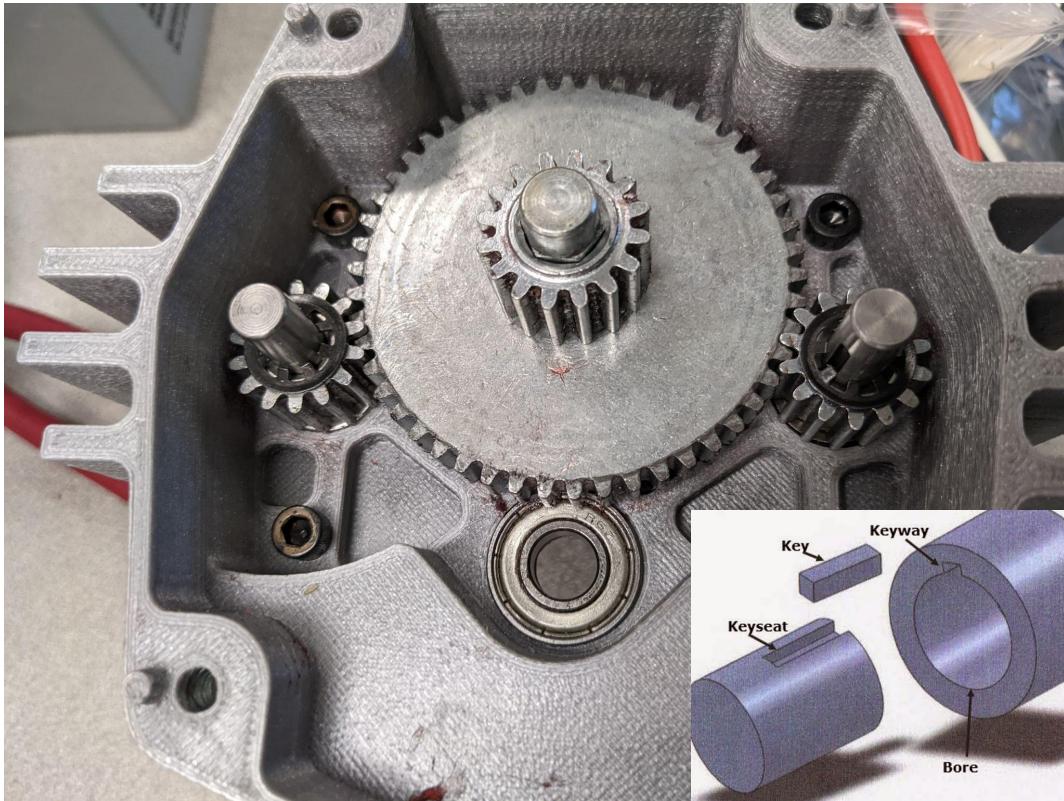


DRIVETRAIN

# DRIVETRAIN - 2 NEO MOTORS FOR LEFT, 2 NEO MOTORS FOR RIGHT



# DRIVETRAIN - THE NEOS CONNECT TO THIS GEARBOX



SHAFT OF NEO MOTOR  
NEEDS KEY TO TURN GEAR

# LOTS OF NEO MOTORS THIS YEAR



PLEASE DO NOT PLUG THESE MOTORS DIRECTLY TO A BATTERY OR IT WILL BLOW UP!  
WATCH OUT FOR OTHER TEAMMATES WHO MAY NOT BE AWARE AND SHARE THIS INFORMATION WITH THEM. THANKS!

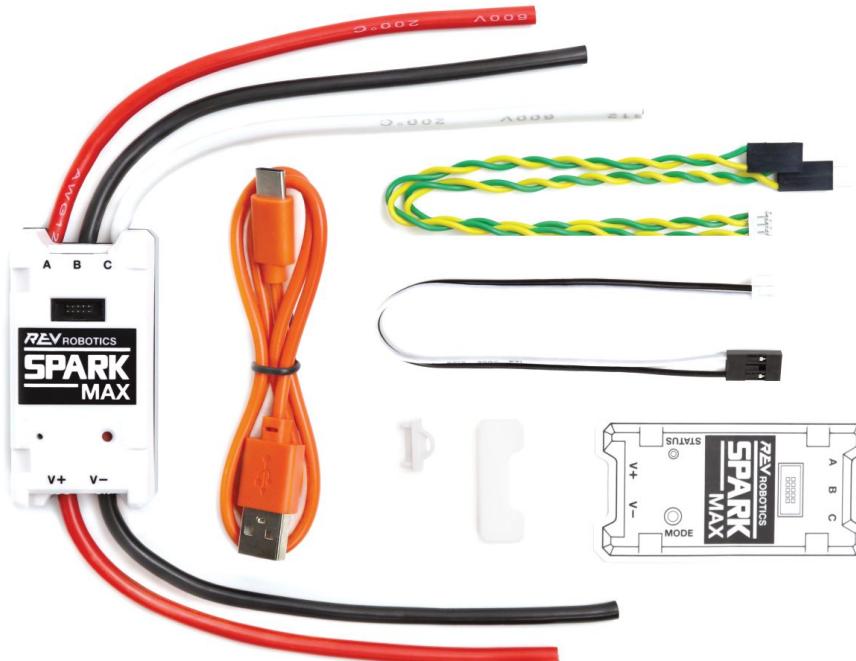
THE NEO MOTORS ARE BRUSHLESS MOTORS. THE NEOS NEED A ELECTRONIC SPEED CONTROLLER (ESC) TO SPIN.

THE CIM MOTORS ARE DC BRUSH MOTORS AND CAN BE CONNECTED TO A BATTERY.

HERE'S HOW BRUSHLESS MOTORS WORK:

[HTTPS://WWW.YOUTUBE.COM/WATCH?V=BCEI0NU0DAC](https://www.youtube.com/watch?v=BCEI0NU0DAC)

# THE NEO IS CONTROLLED BY A SPARK MAX MOTOR CONTROLLER

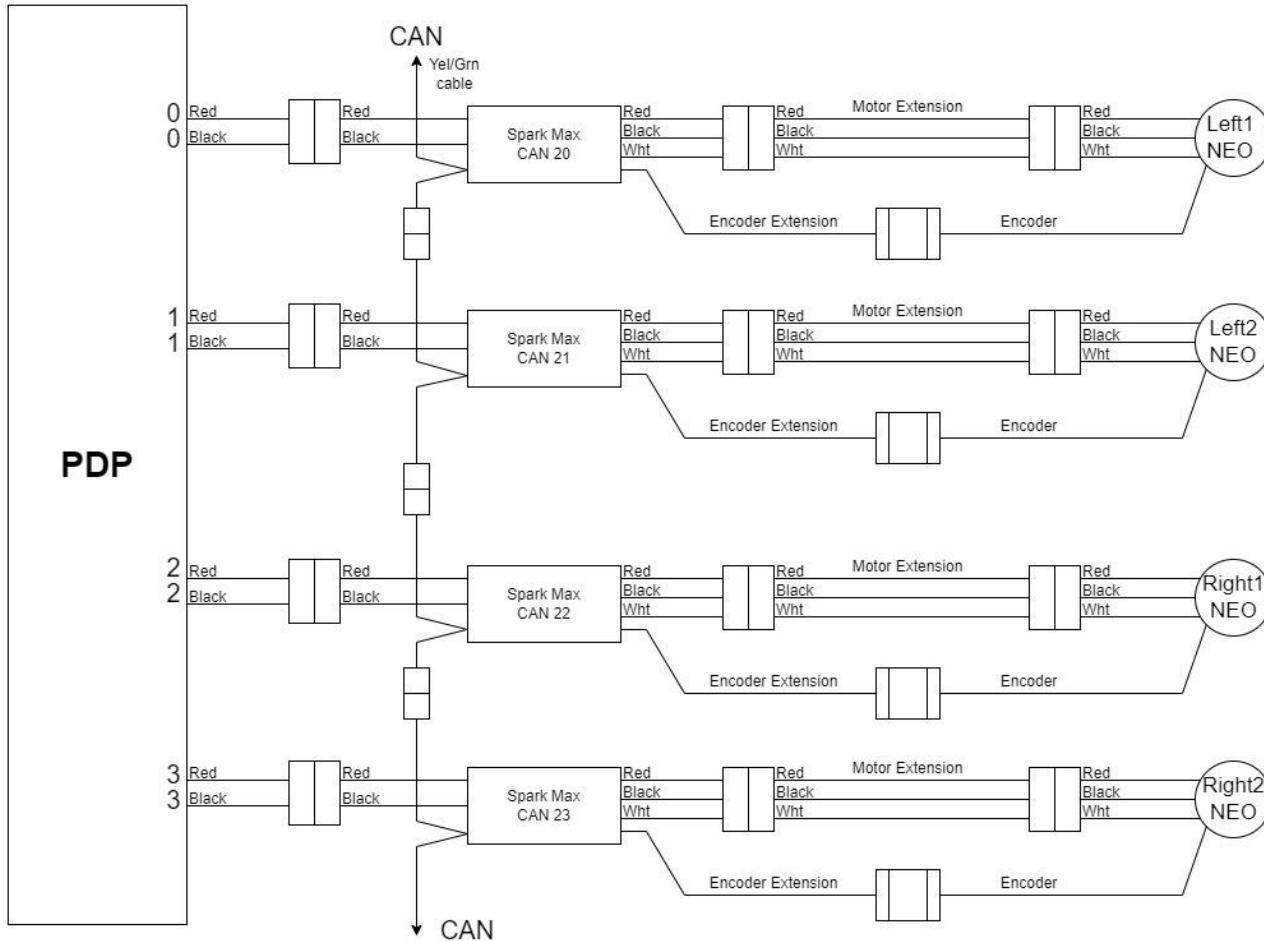


THE ROBORIO TALKS TO THE SPARK MAX USING A NETWORK CALLED CAN. IT'S THE YELLOW/GREEN CABLE THAT YOU SEE.

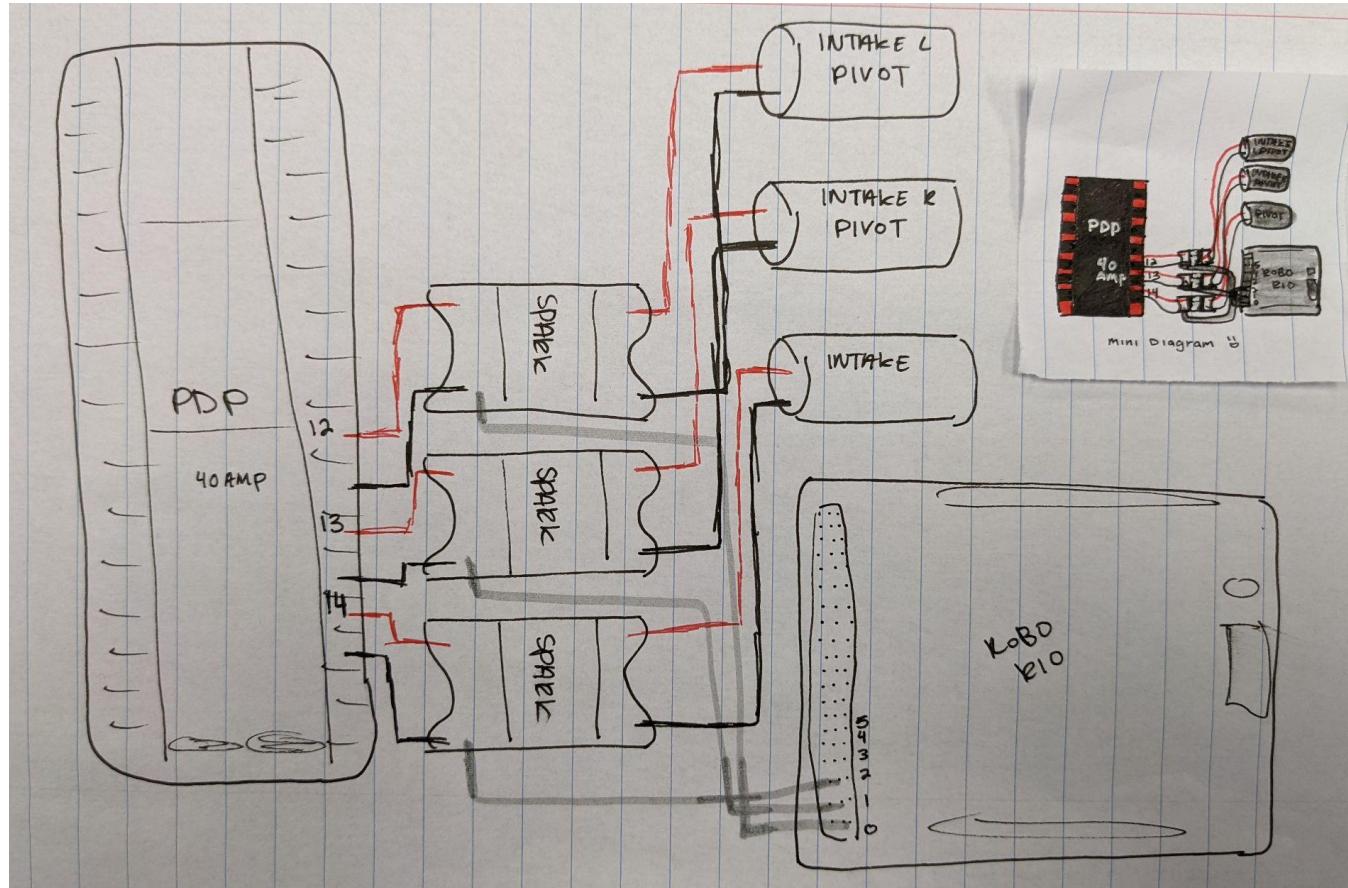
ALL DEVICES ON THE CAN NETWORK HAVE AN ADDRESS / ID NUMBER. ALL THE DEVICES ARE LISTENING TO MESSAGES FROM THE ROBORIO, BUT ONLY RESPOND WHEN THE MESSAGE IS DIRECTED TO ITS ID.

TO CONFIGURE THE SPARK MAX ID AND OTHER PARAMETERS, USE THE [REV HARDWARE CLIENT SOFTWARE](#).

# DriveTrain Wiring



# INTAKE MOTORS - 3 SPARK CONTROLLERS USING PWM SIGNAL



# NEXT, YOU NEED TO ADD VENDOR LIBRARY FOR SPARKMAX

## Online Installation

You can use the online method to install REVLib C++/Java if your development machine is connected to the internet:

1. Open your robot project in VSCode.
2. Click on the WPI icon in the corner to open the WPI Command Pallet.
3. Select **Manage Vendor Libraries**.
4. Select **Install new library (online)**.
5. Enter the following installation URL and press ENTER:  
<https://software-metadata.revrobotics.com/REVLib.json>

## Offline Installation

1. Download and unzip the latest REVLib into the `C:\Users\Public\wpilib\2022` directory on Windows and `~/wpilib/2022` directory on Unix-like systems.
2. Follow the [adding an offline-installed library](#)

# NOW SOME SAMPLE CODE FOR MULTI-MOTOR DIFFERENTIAL DRIVE SYSTEMS

```
import com.revrobotics.CANSparkMax;
import com.revrobotics.CANSparkMaxLowLevel.MotorType;
import edu.wpi.first.wpilibj.motorcontrol.MotorControllerGroup;

public class Robot {
    CANSparkMax m_left1 = new CANSparkMax(1, MotorType.kBrushless);
    CANSparkMax m_left2 = new CANSparkMax(2, MotorType.kBrushless);
    MotorControllerGroup m_left = new MotorControllerGroup(m_left1, m_left2);

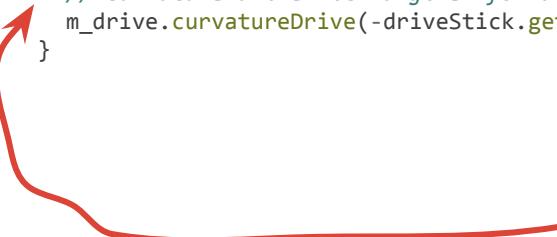
    CANSparkMax m_right1 = new CANSparkMax(3, MotorType.kBrushless);
    CANSparkMax m_right2 = new CANSparkMax(4, MotorType.kBrushless);
    MotorControllerGroup m_left = new MotorControllerGroup(m_right1, m_right2);

    DifferentialDrive m_drive = new DifferentialDrive(m_left, m_right);

    public void robotInit() {
        m_left.setInverted(true); // if you want to invert the entire side
    }
}
```

# DO YOU WANT TO DRIVE A TANK, ARCADE OR CURVATURE?

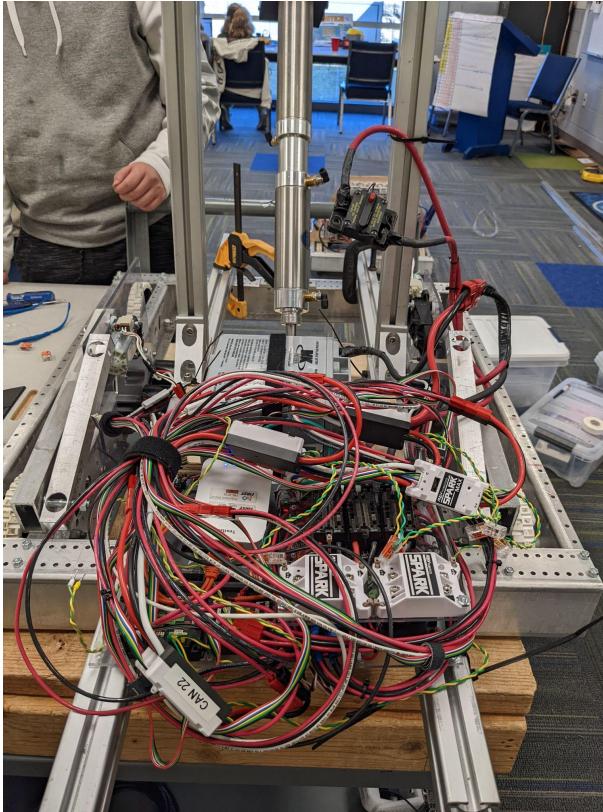
```
public void teleopPeriodic() {  
    // Tank drive with a given left and right rates  
    m_drive.tankDrive(-leftStick.getY(), -rightStick.getY());  
  
    // Arcade drive with a given forward and turn rate  
    m_drive.arcadeDrive(-driveStick.getY(), driveStick.getX());  
  
    // Curvature drive with a given forward and turn rate, as well as a button for turning in-place.  
    m_drive.curvatureDrive(-driveStick.getY(), driveStick.getX(), driveStick.getButton(1));  
}
```



Let's try all three!!!

Curvature might be cool!

# TESTING THE DRIVETRAIN



# HOW DO WE KEEP THE BOT FROM TIPPING OVER?

IF YOU STOP TOO FAST, THE ROBOT WILL TRY TO TIP OVER. IT MIGHT ALSO TIP OVER IF YOU START TOO FAST. ONE WAY TO REDUCE THE CHANCES OF FLIPPING IS TO ACCELERATE AND DECELERATE AT A REASONABLE AMOUNT. HERE'S ONE METHOD FROM CHIEF DELPHI:



timytamy

Jan '15

G gpetilli:

We typically do

$x = joyx * \text{abs}(joyx); y = joyy * \text{abs}(joyy)$

this squares the input giving a parabolic response while maintaining sign.

The wpilib actually has this built in. If you call the arcadeDrive (or tankDrive) method, one of the parameters is a boolean "squaredInputs", which does pretty much what your doing.

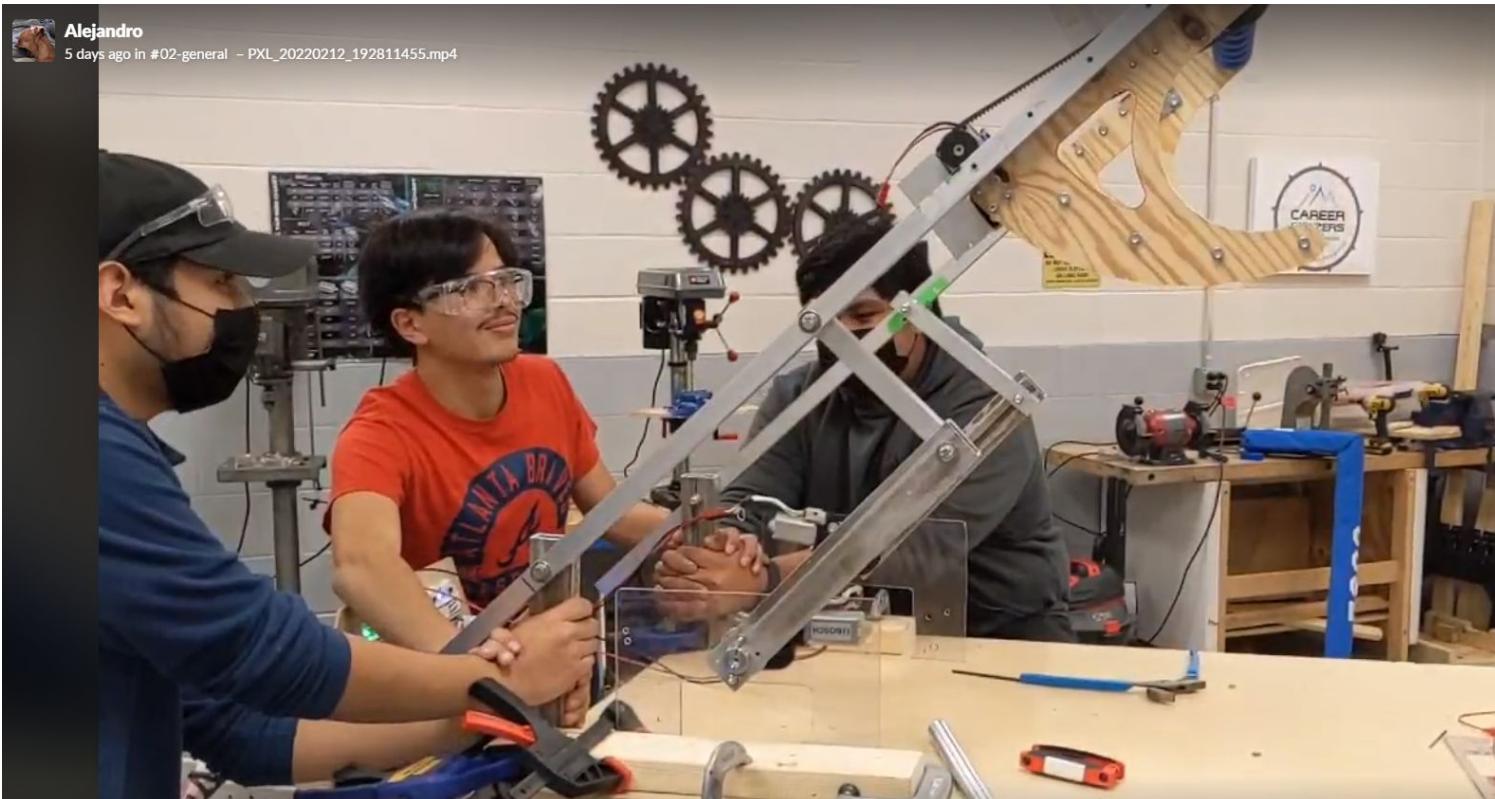
eg:

```
driveSys.arcadeDrive(joyL.getY(), joyR.getX(), true);
```

ANOTHER OPTION TO TRY IS A SLEW RATE LIMITER.

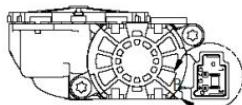
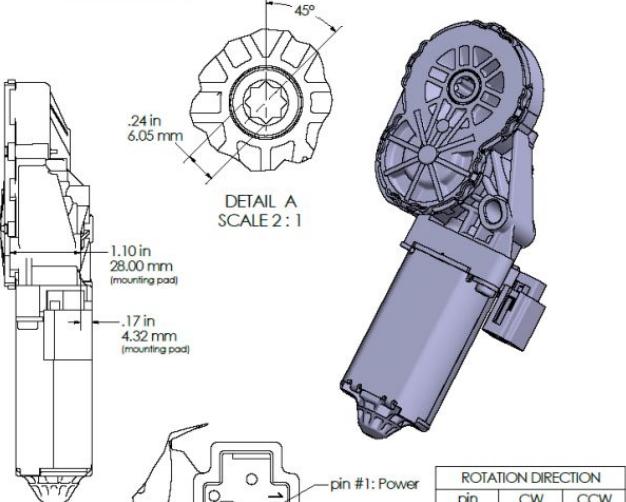
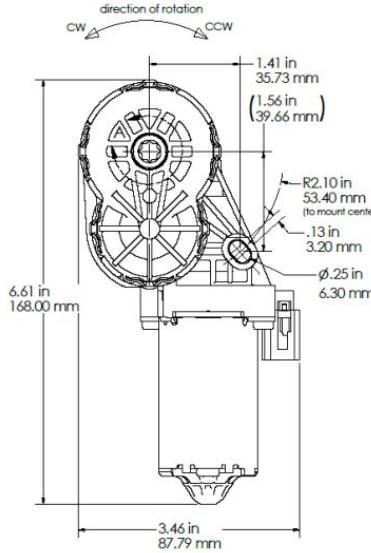
# INTAKE

# INTAKE + INTAKE ACTUATOR



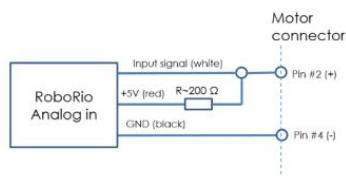
# INTAKE ACTUATOR - LOWERS AND LIFTS INTAKE, 2 BOSCH SEAT MOTORS

[www.bosch.us](http://www.bosch.us)

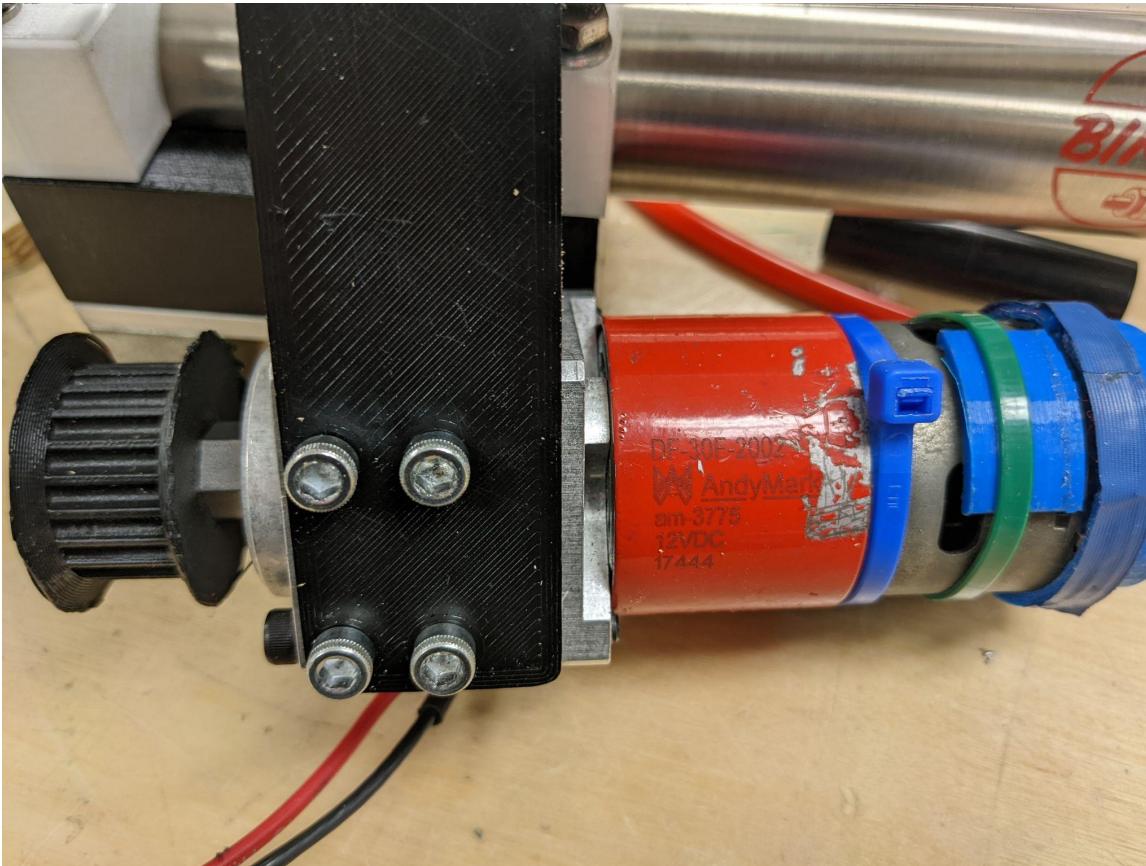


Note: Motor is protected with a thermal switch

Performance values at 13V, 0Ω wire resistance



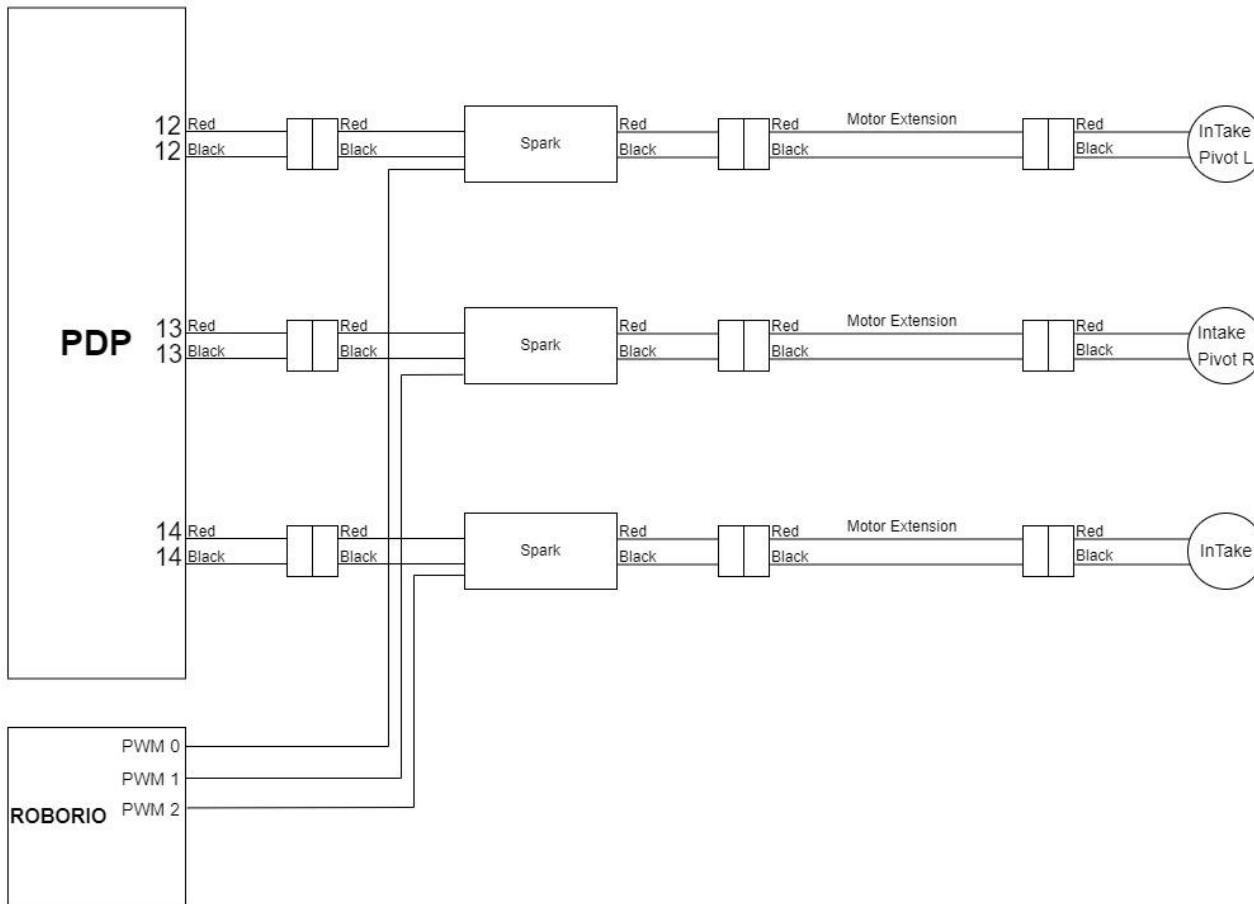
# INTAKE REDLINE MOTOR - RETRIEVE AND DELIVER CARGO



## Specifications

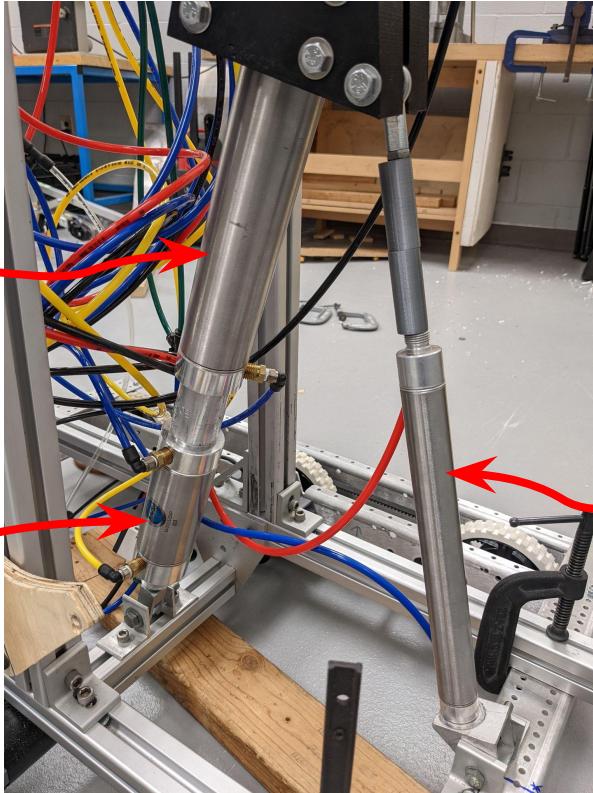
- Bolt Pattern: 2 M4x0.7 Holes on a 29 mm diameter bolt circle
- Diameter: 1.744 in.
- Length: 3.455 in.
- Maximum Power: 380 Watts (+/- 10%)
- No Load Current: 3.8 A
- No Load RPM: 21020 RPM (+/- 10%)
- Output Shaft Diameter: 5 mm
- Output Shaft Length: 0.346 in.
- Stall Current: 130 A
- Stall Torque: 0.7 N·m (6.2 in-lbs) (+/- 10%)
- Voltage: 12 VDC
- Weight: 0.806 lb

# InTake Wiring



CLIMB SYSTEM

# CLIMB CYLINDERS



MAIN CYLINDER

SMALL CYLINDER

ANGLE CYLINDER

# JOYSTICK CONTROLS

# DRIVER CONTROLS



# DRIVER AND PIT CHECKLIST

## Before Match

- Change out Battery (Use Battery Beak check)
- Pre-Charge Air Tanks
- Change bumpers to correct Color
- Joystick connected to correct Port
- Set Robot Preferences to color Color
- Check Robot and make sure nothing falls off
- Have Flag ready
- Send human player to queue with Flag
- **(NO PHONES ALLOWED)**
- Meet with Alliance Teams to discuss strategy

## AFTER MATCH

Review with Programming Crew

Discuss what went wrong and what went right with Bot

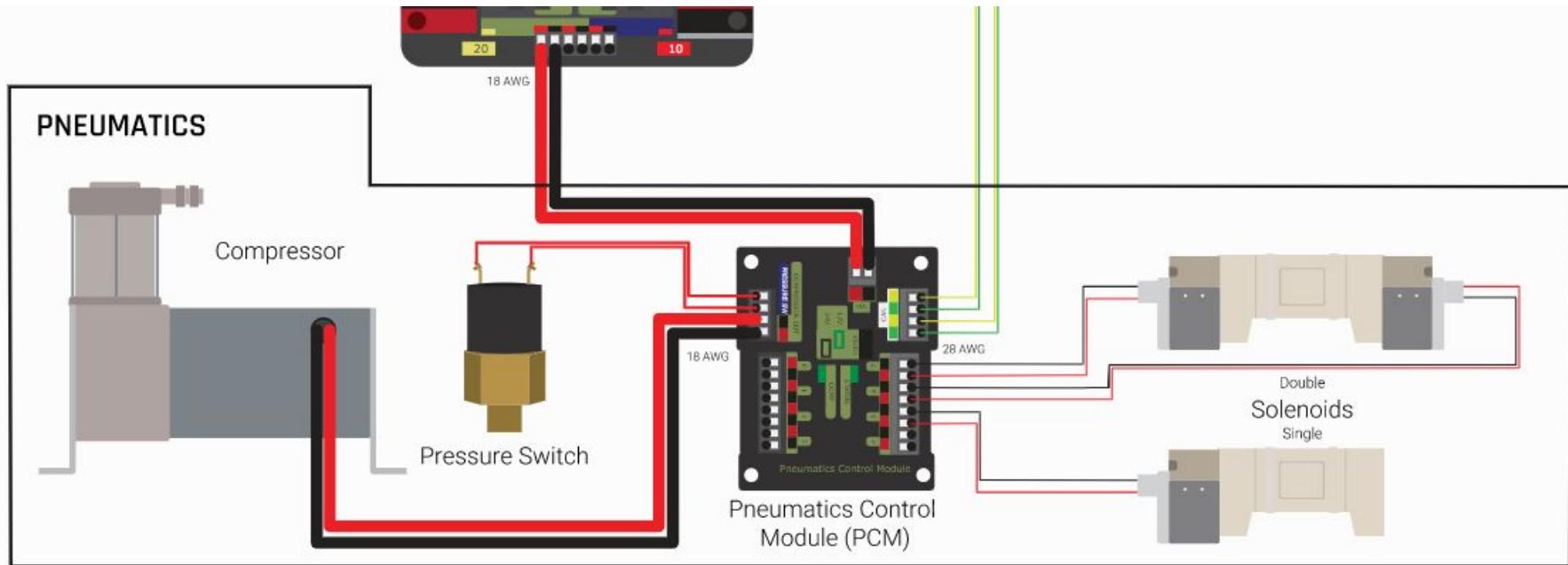
# WHAT'S PNEU, MATICS?

Most of what you need to know to use pneumatics on your bot!

# CYLINDER AND SOLENOID VALVE

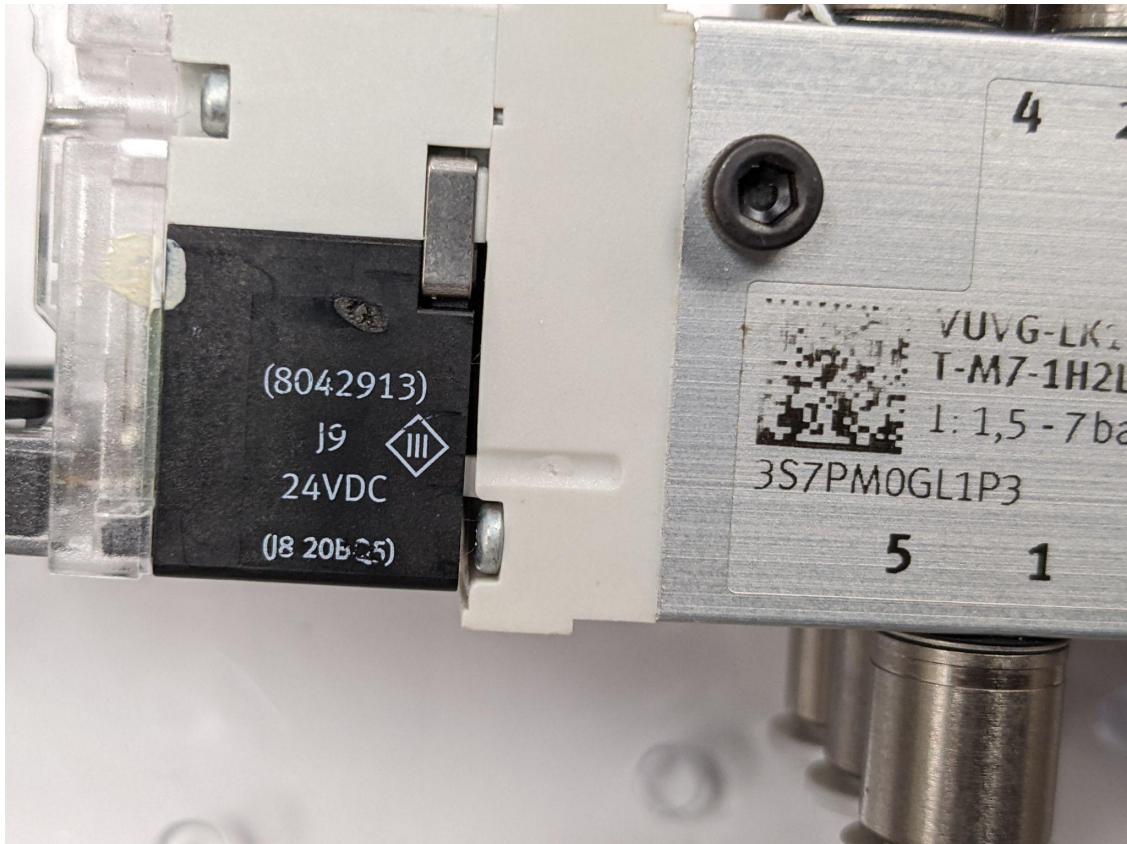
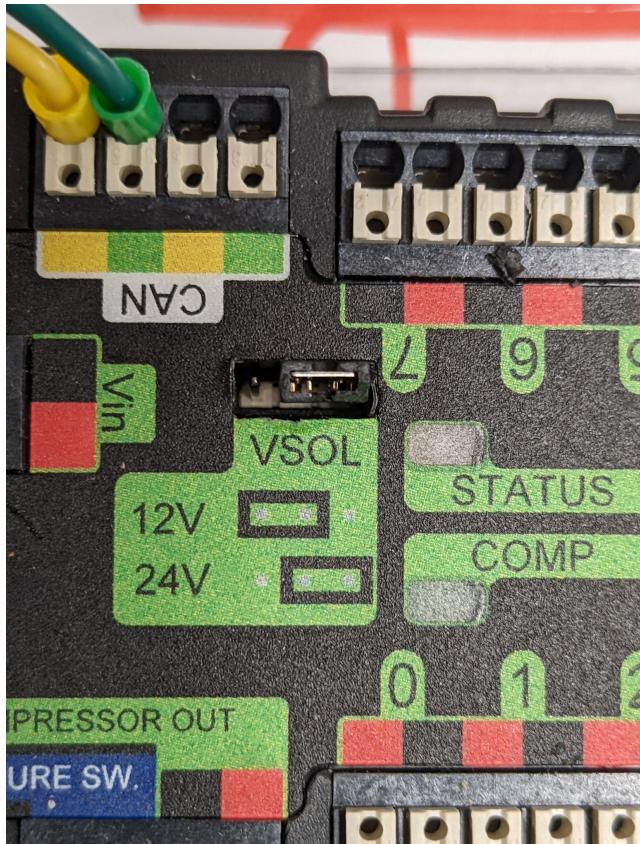


# PNEUMATICS CONTROL MODULE (PCM)



The first PCM on your robot can be wired from the PDP VRM/PCM connectors on the end of the PDP. The PCM is connected to the roboRIO via CAN and can be placed anywhere in the middle of the CAN chain (or on the end with a custom terminator). For more details on wiring a single PCM, see [Pneumatics Control Module Power \(Optional\)](#)

# ARE YOU USING 12V SOLENOIDS OR 24V SOLENOIDS?



LET'S DISSECT SOME  
CODE FOR  
PNEUMATICS  
CONTROL

# FIRST ADD LIBRARIES FOR SOLENOIDS AND PNEUMATIC CONTROLLER

```
import edu.wpi.first.wpilibj.DoubleSolenoid;  
import edu.wpi.first.wpilibj.Joystick;  
import edu.wpi.first.wpilibj.PneumaticsModuleType;  
import edu.wpi.first.wpilibj.Solenoid;  
import edu.wpi.first.wpilibj.TimedRobot;
```

THE DOUBLESOLENIOD LIBRARY IS FOR VALVES THAT REQUIRE TWO SIGNALS. ONE SIGNAL TO EXTEND THE CYLINDER AND ANOTHER SIGNAL TO RETRACT THE CYLINDER.

THE SOLENOID LIBRARY IS FOR VALVES THAT REQUIRE ONLY ONE SIGNAL TO EXTEND AND RETRACT THE CYLINDER. USUALLY, THERE IS A SPRING THAT PHYSICALLY RETURNS THE VALVE TO THE OFF STATE WHEN THE SIGNAL IS OFF.

# CREATING A SOLENOID AND A DOUBLESOLENOID OBJECT

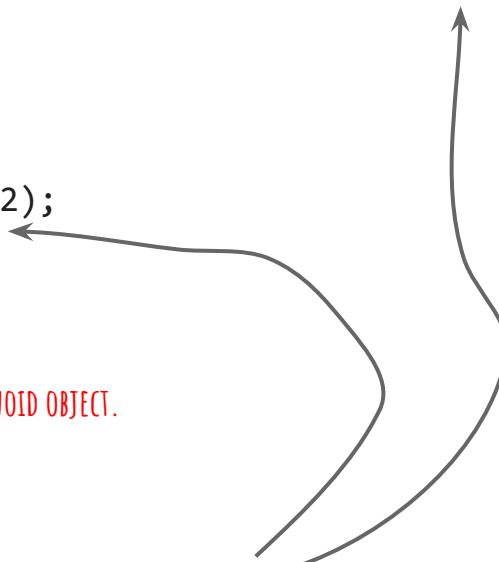
```
// Solenoid corresponds to a single solenoid.  
private final Solenoid m_solenoid = new Solenoid(PneumaticsModuleType.CTREPCM, 0);  
  
// DoubleSolenoid corresponds to a double solenoid.  
private final DoubleSolenoid m_doubleSolenoid =  
    new DoubleSolenoid(PneumaticsModuleType.CTREPCM, 1, 2);
```

THE WORD NEW SAYS THAT WE ARE CREATING AN OBJECT. IN THIS CASE IT IS A SOLENOID OBJECT AND A DOUBLESOLENOID OBJECT.

WE NAME THE SOLENOID OBJECT M\_SOLENOID.

WE NAME THE DOUBLESOLENOID OBJECT M\_DOUBLESOLENOID

THESE NUMBERS ARE FOR THE CHANNELS THAT ARE USED TO CONTROL THESE SOLENOIDS. NOTICE THAT THE DOUBLESOLENOID HAS TWO SIGNALS.



# CONTROLLING THE VALVES IN TELEOP MODE

```
@Override  
  
public void teleopPeriodic() {  
  
    /*  
     * The output of GetRawButton is true/false depending on whether  
     * the button is pressed; Set takes a boolean for whether  
     * to use the default (false) channel or the other (true).  
     */  
  
    m_solenoid.set(m_stick.getRawButton(kSolenoidButton));  
  
    /*  
     * In order to set the double solenoid, if just one button  
     * is pressed, set the solenoid to correspond to that button.  
     * If both are pressed, set the solenoid will be set to Forwards.  
     */  
  
    if (m_stick.getRawButton(kDoubleSolenoidForward)) {  
        m_doubleSolenoid.set(DoubleSolenoid.Value.kForward);  
    } else if (m_stick.getRawButton(kDoubleSolenoidReverse)) {  
        m_doubleSolenoid.set(DoubleSolenoid.Value.kReverse);  
    }  
}
```

THIS ACTIVATES THE SOLENOID VALVE IF BUTTON IS PRESSED.

ONE BUTTON EXTENDS ANOTHER BUTTON RETRACTS THE DOUBLESOLENOID.

# RESOURCES ON USING PNEUMATICS

GREAT VIDEO ON PNEUMATICS

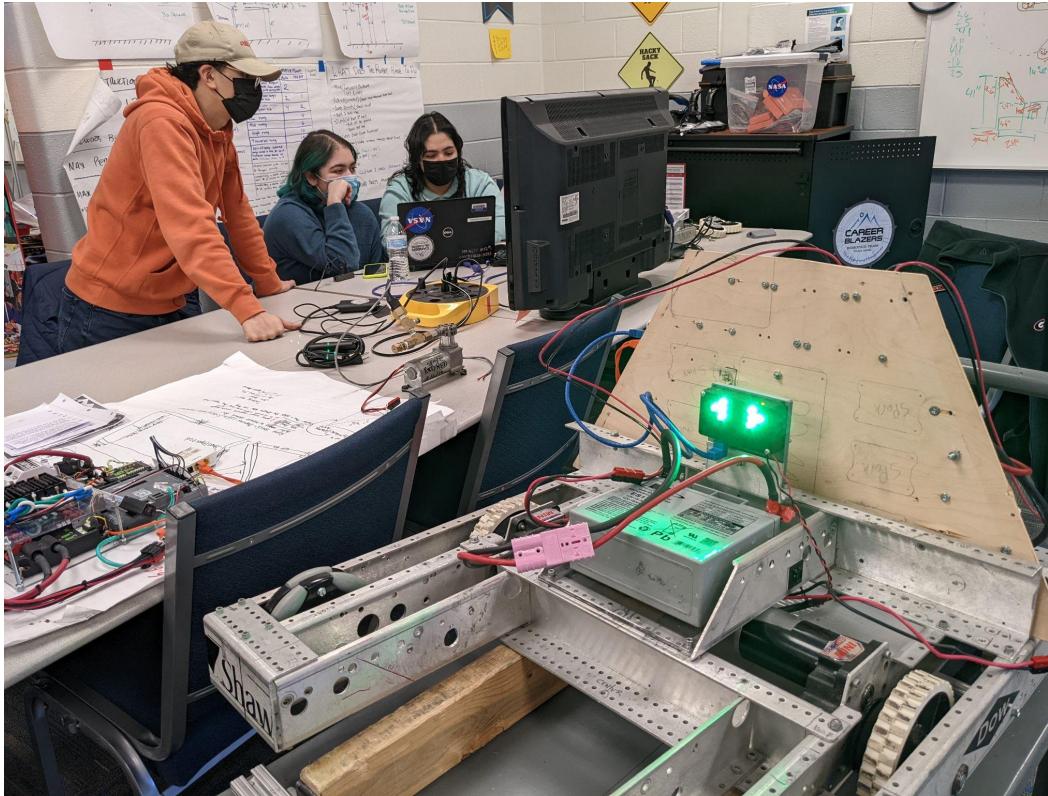
WPILIB SECTION ON PNEUMATICS

CAMERA

WANT SOME LIME WIT DAT?

STUFF YOU MIGHT WANT TO KNOW ABOUT THE LIMELIGHT

# LIMELIGHT CAMERA



HOW DO I SELECT A  
DIFFERENT PIPELINE  
FOR THE LIMELIGHT?

# FIRST, WHAT'S A PIPELINE AND WHAT CAN I DO WITH IT?

THE LIMELIGHT CAN HAVE UP TO 10 PIPELINES ( 0 TO 9 )

THINK OF A PIPELINE AS A PROGRAM OR CONFIGURATION OF THE CAMERA TO LOOK FOR A CERTAIN TARGET.

FOR RAPID-REACT, WE COULD HAVE A PIPELINE FOR DETECTING A RED CARGO, A PIPELINE FOR BLUE CARGO, A PIPELINE TO LOOK FOR UPPER HUB GOAL, ETC.

HOW DO WE SELECT WHICH PIPELINE TO RUN ON THE CAMERA?

# HOW DO WE SELECT WHICH PIPELINE TO RUN ON THE LIMELIGHT?

The screenshot shows a web browser window with the URL [https://docs.limelightvision.io/en/latest/networktables\\_api.html](https://docs.limelightvision.io/en/latest/networktables_api.html). The page displays the NetworkTables API documentation for Limelight. The left sidebar contains navigation links for Camera Controls, Python, Corners, Advanced Usage with Raw Contours, Case Studies, Python Scripting, GRIP Support, and MISC. The main content area shows code snippets for Java, LabView, C++, and Python. The Python tab is selected, showing the following code snippet:

```
NetworkTableInstance.getDefault().getTable("limelight").getEntry("<variablename>").setNumber(<value>)
```

Below the code, there is a note: "to set this data:" followed by several tables describing parameters:

ledMode	Sets limelight's LED state
0	use the LED Mode set in the current pipeline
1	force off
2	force blink
3	force on

camMode	Sets limelight's operation mode
0	Vision processor
1	Driver Camera (Increases exposure, disables vision processing)

pipeline	Sets limelight's current pipeline
0 .. 9	Select pipeline 0..9

stream	Sets limelight's streaming mode
0	Standard - Side-by-side streams if a webcam is attached to Limelight

At the bottom of the browser window, the address bar shows "v: latest". The taskbar at the bottom includes a search bar, the Start button, and various pinned application icons. The system tray shows the date (2/6/2022), time (5:31 PM), weather (48°F Sunny), and battery status.

# USING THE NETWORKTABLEINSTANCE FUNCTION

```
NetworkTableInstance.getDefault().getTable("limelight").getEntry("<variablename>").setNumber(<value>);  
  
NetworkTableInstance.getDefault().getTable("limelight").getEntry("pipeline").setNumber(0); //select pipeline 0  
  
NetworkTableInstance.getDefault().getTable("limelight").getEntry("pipeline").setNumber(1); //select pipeline 1  
  
NetworkTableInstance.getDefault().getTable("limelight").getEntry("ledMode").setNumber(1); //turn off LEDs  
  
NetworkTableInstance.getDefault().getTable("limelight").getEntry("ledMode").setNumber(3); //turn on LEDs
```

SEE THE PATTERN?

WHAT VALUES CAN WE  
GET FROM THE  
LIMELIGHT?

# LIMELIGHT DATA

Screenshot of a web browser displaying the Limelight NetworkTables API documentation.

The page title is "Complete NetworkTables API".

The URL is [https://docs.limelightvision.io/en/latest/networktables\\_api.html](https://docs.limelightvision.io/en/latest/networktables_api.html).

The left sidebar contains a navigation menu:

- Camera Controls
- Python
- Corners
- Advanced Usage with Raw Contours
- Case Study: Estimating Distance
- Case Study: Aiming Using Vision
- Case Study: Seeking
- Case Study: Getting in Range
- Case Study: Aiming and Range at the same time.
- Case Study: 2017 Fuel Robot
- Case Study: DEEP SPACE 2019 Examples
- PYTHON SCRIPTING**
- Using Python to create Custom OpenCV Vision Pipelines
- Python Examples
- GRIP SUPPORT**
- Using Grip to create Custom OpenCV Vision Pipelines
- Running GRIP Pipelines on Limelight
- MISC**

The main content area shows tabs for Java, LabView, C++, and Python. The Python tab is selected, showing the following code example:

```
NetworkTableInstance.getDefault().getTable("limelight").getEntry("<variablename>").getDouble(0);
```

Below the code example, it says "to retrieve this data:" followed by a table of variables:

tv	Whether the limelight has any valid targets (0 or 1)
tx	Horizontal Offset From Crosshair To Target (LL1: -27 degrees to 27 degrees   LL2: -29.8 to 29.8
ty	Vertical Offset From Crosshair To Target (LL1: -20.5 degrees to 20.5 degrees   LL2: -24.85 to 24.85
ta	Target Area (0% of image to 100% of image)
ts	Skew or rotation (-90 degrees to 0 degrees)
tl	The pipeline's latency contribution (ms) Add at least 11ms for image capture latency.
tshort	Sidelength of shortest side of the fitted bounding box (pixels)
tlong	Sidelength of longest side of the fitted bounding box (pixels)
thor	Horizontal sidelength of the rough bounding box (0 - 320 pixels)
tvert	Vertical sidelength of the rough bounding box (0 - 320 pixels)
getpipe	True active pipeline index of the camera (0 .. 9)
camtran	Results of a 3D position solution, NumberArray: Translation (x,y,z) Rotation(pitch,yaw,roll)

At the bottom, there is a search bar with the placeholder "Type here to search" and a taskbar with various pinned icons.

NOW, WRITE DOWN HOW YOU THINK IT SHOULD WORK. THEN, START SPERIMENTING!!!

autonomousPeriodic

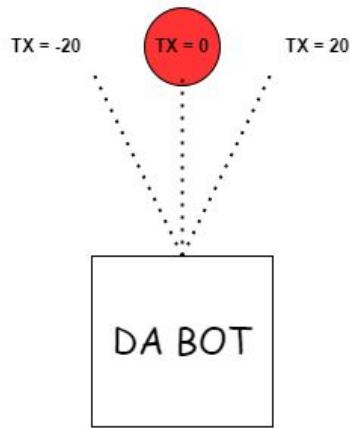
```
if (m_stick.getRawButton(3)) {
    if (tx != 1) {
        Limelight HasValid Target = FALSE
        [ same as before ]
    } else {
        Limelight HasValid Target = TRUE
        while (tx < -0.2) or (0.2 < tx) {
            Limelight Steer Command = tx * STEER-K
            [ Move drivetrain (driveSpeed), Limelight Steer command ]
        }
    }
}
```

- would DRIVE-K being not 0 effect the speed of turning?

Might need to be turned up

whatever # we decide is too far from 0

# LIMELIGHT TX DATA



If TX is zero-ish, do not move DA BOT!

What is zero-ish? whatever you like it to be!

We can start with  $\pm 1$ . TX has to be greater than 1 or less than -1 for motor to steer to cargo

What's the minimum voltage needed to move the motors?

I think this is around 0.35 Volts. So we need the speed voltage to be greater than 0.35 Volts or less than -0.35 Volts. Remember the speed voltage can be from -1.0 to +1.0

If TX is maxed out at -20 or +20, how fast do we want the motor to run? Let's start with 0.5 Volts

$$V_{min} = 0.35$$

$$V_{max} = 0.50$$

$$TX_{max} = 20$$

$$TX_{min} = 1$$

$$V_{delta} = (V_{max} - V_{min}) / (TX_{max} - TX_{min}) \\ = 0.008$$

Check the math:  
When TX = 1.0,  
SteerCommand = 0.358

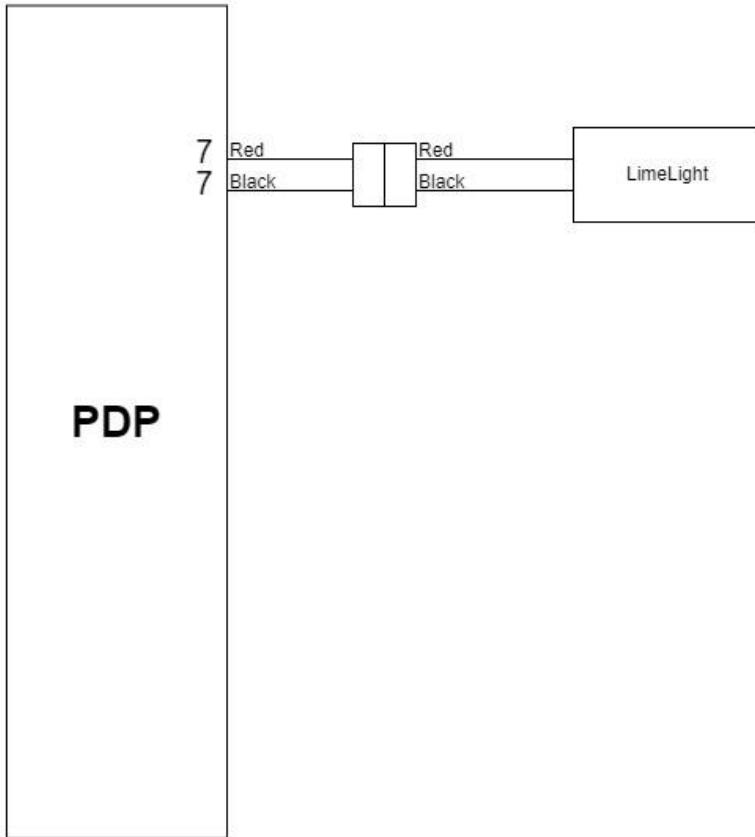
If TX  $\geq 1.0$   
SteerCommand =  $0.35 + TX * V_{delta}$   
ELSE IF TX  $\leq -1.0$   
SteerCommand =  $-0.35 + TX * V_{delta}$   
ELSE  
SteerCommand = 0

When TX = 20.0  
SteerCommand = 0.51

When TX = -1.0,  
SteerCommand = -0.358

When TX = -20.0  
SteerCommand = -0.51

# LimeLight Wiring



# RESOURCES ON USING THE LIMELIGHT

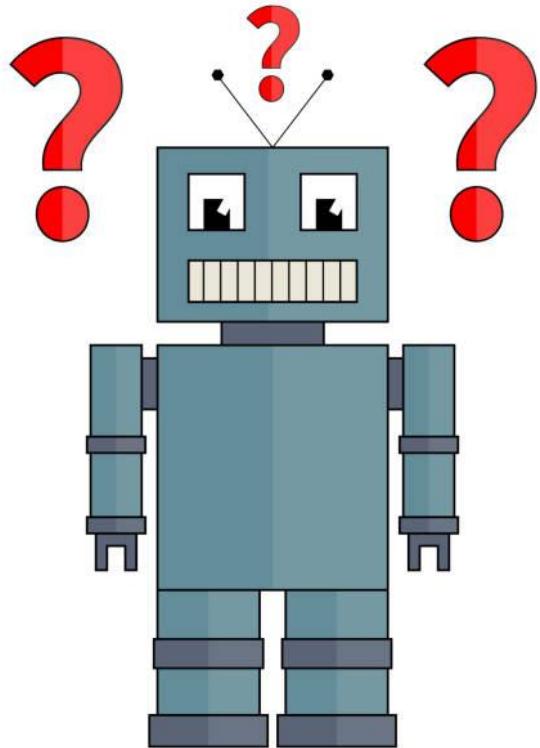
LIMELIGHT FUNCTIONS

LIMELIGHT VIDEO FROM TEAM 3244

# WHAT IS YOUR PREFERENCE?

LEARN ABOUT ROBOT PREFERENCES

# TWEAK SETTINGS ON THE ROBOT WITHOUT REWRITING CODE



# WHY USE ROBOT PREFERENCES?

SET MAXIMUM SPEED FOR DRIVETRAIN

TELL THE ROBOT WHERE IT IS PLACED AT THE START OF THE MATCH - RUN AUTONOMOUS SEQUENCE BASED ON START LOCATION

TELL THE ROBOT WHAT COLOR ALLIANCE IT IS FOR THE MATCH - USE TO SELECT PIPELINE FOR LIMELIGHT

INSTEAD OF HAVING TO RECOMPILE / DEPLOY NEW CODE EVERYTIME YOU CHANGE A SETTING, YOU CAN STORE IN ROBOT PREFERENCES

RETAINS PREFERENCES WHEN TURNING OFF POWER TO THE ROBOT - YOU DON'T HAVE TO RE-ENTER THE SETTINGS.

# USE SMARTDASHBOARD TO CREATE ROBOT PREFERENCES

The screenshot shows a web browser window displaying a guide on how to use SmartDashboard to create robot preferences. The browser tabs include "Robot Preferences - Google Slides", "Setting Robot Preferences — FIRST", and "Setting Robot Preferences — FIRST". The main content area has a title "Displaying Preferences in SmartDashboard" and a sub-section "Viewing and Editing Preferences".

**Displaying Preferences in SmartDashboard**

In the SmartDashboard, the Preferences display can be added to the display revealing the contents of the preferences file stored in the roboRIO flash memory.

**Viewing and Editing Preferences**

SmartDashboard – 190

Key	Value
ArmUpPosition	1.0
ArmDownPosition	4.0

File View

- Editable ^E
- ✓ Edit Subsystems ^⇧E
- Reset LiveWindow ^R
- Add... ▾
- Reveal... ▾
- Remove Unused
- Robot Preferences

Robot Preferences

robot\_pref.jpg

Type here to search

42°F 12:03 PM 2/7/2022 25

# READ ROBOT PREFERENCES

The Robot Preferences (Java, C++) class is used to store values in the flash memory on the roboRIO. The values might be for remembering preferences on the robot such as calibration settings for potentiometers, PID values, etc. that you would like to change without having to rebuild the program. The values can be viewed on the SmartDashboard and read and written by the robot program.

## Reading and Writing Preferences

Often potentiometers are used to measure the angle of an arm or the position of some other shaft. In this case, the arm has two positions, `ArmUpPosition` and `ArmDownPosition`. Usually programmers create constants in the program that are the two pot values that correspond to the positions of the arm. When the potentiometer needs to be replaced or adjusted then the program needs

Java    C++

```
public class Robot extends TimedRobot {  
    double armUpPosition;  
    double armDownPosition;  
  
    public void robotInit() {  
        armUpPosition = Preferences.getDouble("ArmUpPosition", 1.0);  
        armDownPosition = Preferences.getDouble("ArmDownPosition", 4.);  
    }  
}
```

Read the Docs    v: stable

robot\_pref.jpg

Type here to search

42°F 12:09 PM 2/7/2022 25

DEFAULT VALUE IF  
PREFERENCE IS NOT  
FOUND

# RESOURCES ON USING ROBOT PREFERENCES

SETTING ROBOT PREFERENCES

# A-TON-OF-MOOSE

Learn how to use enum and switch in Java

# ENUM - PROVIDES A DESCRIPTIVE NAME TO CONSTANT VALUES

```
enum AutoStates  
{  
    wait_for_activation,  
    initialize,  
    wait_for_turn,  
    wait_for_elapsetime  
};
```

So what?

Instead of Autostates equalling 0, 1, 2, 3

The AutoStates values can be given a descriptive name.

In this example,

AutoState.wait\_for\_activation has a value of 0.

AutoState.initialize has a value of 1.

AutoState.wait\_for\_turn has a value of 2..

AutoState.wait\_for\_elapsetime has a value of 3.

Let's see how this makes ATonOfMoose code more readable!

# CREATE A VARIABLE FOR THE STATE OF THE AUTONOMOUS CODE.

```
private AutoStates autonomous_state;
```

The type of data  
that the variable  
we will be storing

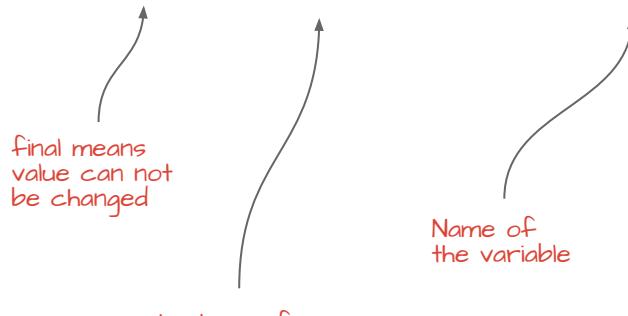
Name of  
the variable

The variable `autonomous_state` is of the type `enum AutoStates` that we created in the previous slide.

What is a variable? A variable is a placeholder. A location in the roboRIO's memory where we want to store a value that we will need to read or change later in the program.

# CREATE A CONSTANT VALUE FOR TIMING A DRIVETRAIN MOVE

```
final double MoveCountDown = 2.0;
```



final means  
value can not  
be changed

The type of  
data that the  
variable will be  
storing is a  
decimal number

Name of  
the variable

The variable called `MoveCountDown` is of type `double` which is a decimal number.

What is `final` mean? It says that this value cannot be changed.

We will use this value to tell us how long to move the robot.

In this example, the `2.0` is the number of seconds for our move.

# SET THE ATONOMOUS\_STATE VARIABLE FOR THE START OF THE MATCH

```
public void StartGameClock()
{
    atonomous_state = AutoStates.initialize;
}
```

When the game starts, the robot will be placed in ATonOfMoose mode for 15 seconds.

The variable atonomous\_state is set to the initialize state.

# DURING AUTONOMOUS - THIS ROUTINE WILL BE CALLED EVERY 20MILLISECONDS.

```
public void autonomousPeriodic()
{
    switch( atonomous_state )
    {
        case wait_for_activation:
            break;

        case initialize:
            aclock.reset();
            aclock.start();
            Atonomous_state = AutoStates.wait_for_turn;
            break;

        case wait_for_turn:
            if( aclock.get() > 0.8)
                MoveDriveTrain( 0, 0);
            autonomous_stat = AutoStates.wait_for_elapsetime;
            else
                MoveDriveTrain( 0, -0.5);
            break;

        case wait_for_elapsetime:
            if( aclock.get() > MoveCountDown )
            {
                MotorsOff();
                Atonomous_state = AutoStates.wait_for_activation;
            }
            else
                MoveDriveTrain( -0.5, 0 );
            break;

        default:
            atonomous_state = AutoStates.wait_for_activation;
            break;
    }

    // reset timer
    // start timing

    // wait 0.8 seconds
    // turn off both motors

    // CCW left motor off and right motor at half speed

    // wait till time is greater than MoveCountDown

    // Turn off the drivetrain motors
    // We are done, just wait now for teleOp mode

    // CW left motor at half speed and right motor off

    // if atonomous_state is unknown, set to wait_for_activation
}
```

SHOW A LITTLE CLASS!

# BASIC CLASS DEFINITION - A CLASS IS A LIKE A BLUEPRINT DESCRIBING AN OBJECT

```
// Name of the package must be same as the directory as our project folder

package frc.robot; //We need this to say this class is part of our frc robot code

public class MyClass
{
    MyClass(){ //This is a constructor - when you use new keyword to create object

    }

    public void func_one(String s)
    {
    }

    public void another_func()
    {
    }
}
```

[MORE INFO ON JAVA CLASS HERE!!!](#)

# TODO LIST

# TO DO LIST

- TEST DRIVETRAIN
- TEST CLIMB CODE
- WIRE UP SOLENOID VALVES, COMPRESSOR, PRESSURE SENSOR
- FINISH AUTONOMOUS CODE AND TEST
- BUILD CONTROLS ENCLOSURE
- FIND HOME FOR CONTROLS BOX ENCLOSURE
- FIND HOME FOR LIMELIGHT, COMPRESSOR AND BATTERY
- WORK WITH DRIVERS ON HOW THEY WANT TO CONTROL DA BOT
- SETUP ROBOT PREFERENCES FOR ALLIANCE COLOR
- SEPARATE CODE FOR DIFFERENT PARTS OF THE BOT - IN JAVA WE CALL THESE CLASSES

DA PROGRAMMING  
AND  
ELECTRICAL CREW

# RAPID REACT PROGRAMMING / ELECTRICAL CREW

ALEJANDRO MIRANDA JR

ISAAC JONES

VANNESA GARCIA

AUSTEN MELENDEZ

CRYSTAL MELENDEZ

GISELLE RODRIGUEZ

ZERO HARDY

WESLEY SANTANA

MIA MENDOZA

JAZMINE AYABAR

IVAN FRANCO

HUGO LOZANO

WE



KEEP EXPERIMENTING, KEEP LEARNING, AND  
MOST OF ALL, HAVE FUN!!!!

# ROBOTICS ARCHIVES

# PROGRAMMERS FOR 2021 INFINITE RECHARGE 2

Isaac Jones

Alejandro Miranda

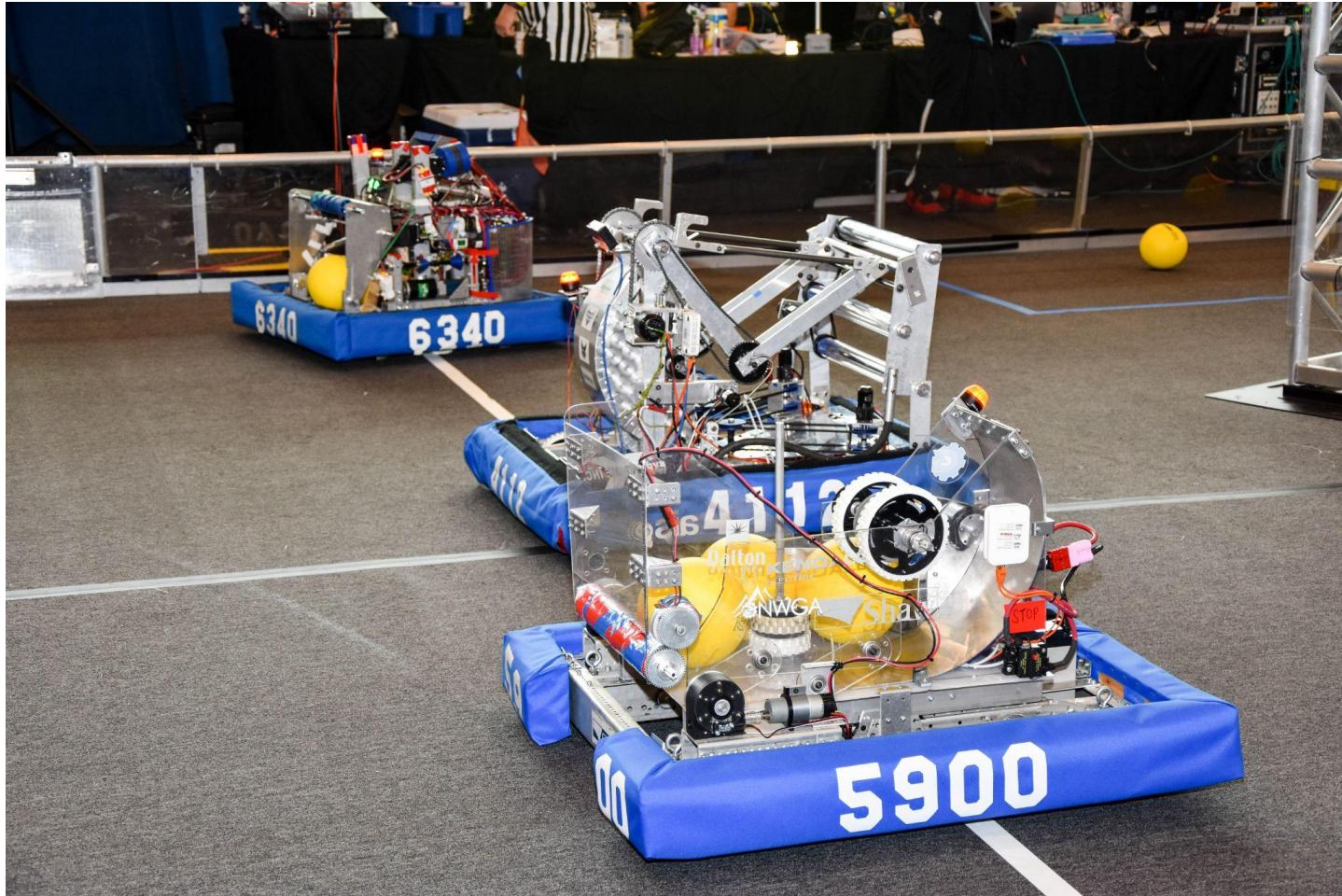
Dylan McClure

Caleb Henricks

(Java)

Bot: Chonk 2.0 and  
Wyoming





# PROGRAMMERS FOR 2020 INFINITE RECHARGE

Troy Johnson

Alejandro Miranda

Jazmine Colindres

Trevor Silvers

Gage Godfrey

(Java)

Bot: No Quema Cuh





# PROGRAMMERS FOR 2019 DEEP SPACE

Gage Godfrey

Conner Henderson

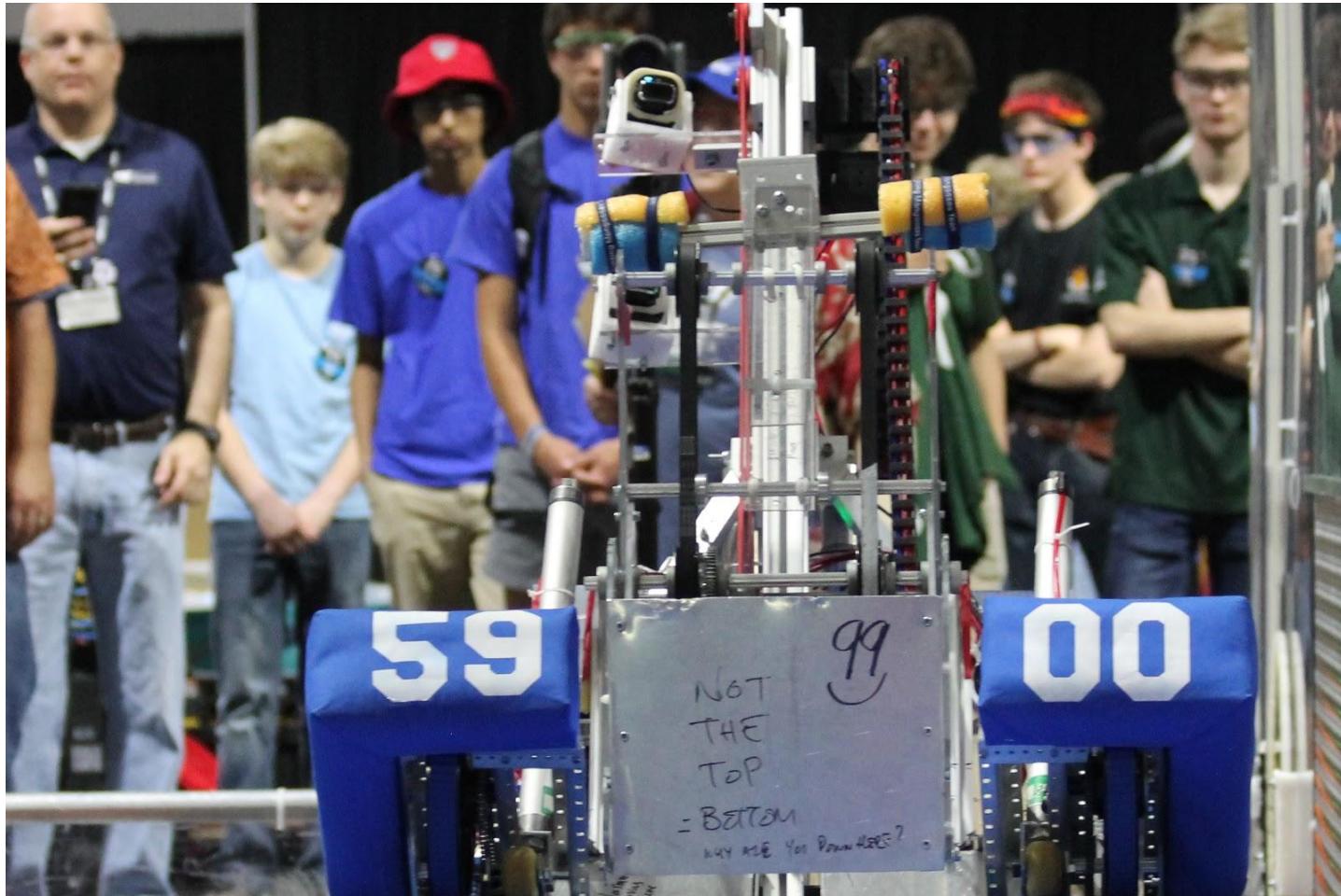
Trevor Silvers

Ian Scaife

(Java)

Bot: Rocket





# PROGRAMMER FOR 2018 POWER UP

Morrison Outlaw  
(Java)

Bot: ZotsBot





# PROGRAMMERS FOR 2017 STEAMWORKS

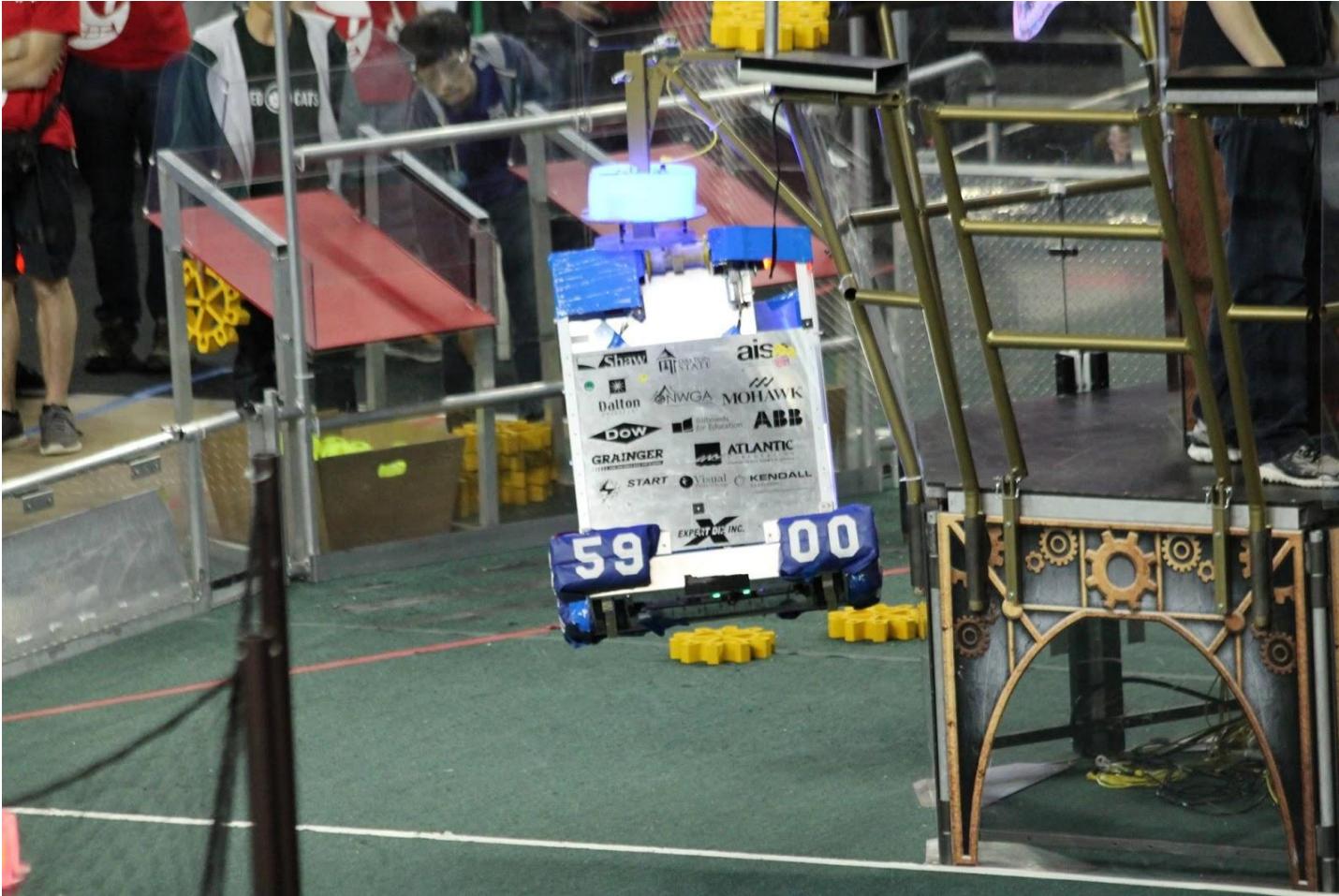
Gage Godfrey

Jon Tan

(Java)

Bot: Spoons





# PROGRAMMER FOR 2016 STRONGHOLD

Elijah Pemberton  
(LabView)

Bot: LemonSweets

