

Building a Pulse Width Modulation (PWM) Signal Generator

*William Bretton
Ben Bretton*

*Triple Helix, the Menchville High School Robotics Team
FIRST Robotics Competition Team 2363*



NEED

As most FRC teams understand, time is critical during the build season. We have seen many instances where a prototype mechanism needed to be tested before any software was written. Often, the mechanism needed to be tested by controlling one or two motors through their full operating range from 100% in the forward direction to 100% in the reverse. There were also several other use cases where the ability to manually drive a motor or servo was needed:

- Controlling drive train motor(s) to ensure proper operation
- Controlling shooter mechanisms for tuning RPMs and testing different shooter wheels
- Controlling arm-like mechanisms requiring slow and precise movement
- Bench-checking motors

REQUIREMENTS

We used the following requirements to design and implement our solution:

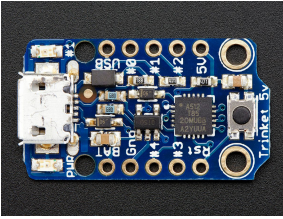
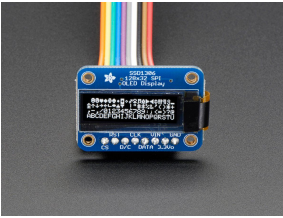
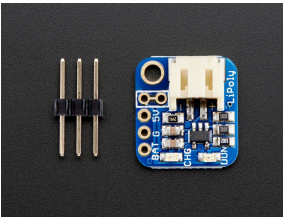

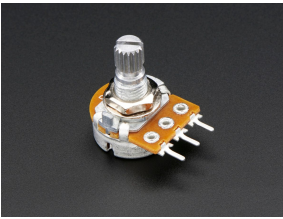
1. The device must be able to generate a Pulse Width Modulation signal compatible with common FRC motor controllers such as the Talon SRX.
2. The device must be able to generate a Pulse Width Modulation signal compatible with common servos.
3. The device must be battery powered and rechargeable using standard micro USB cable.
4. The device must be able to function using micro USB connection when battery is low.
5. The device must be portable enough to keep in a toolbox.
6. The device must allow for fine control throughout the motor controller's range.
7. The device must provide visual feedback of the duty cycle output from -100% to +100%.
8. The device must create a PWM signal between 1000 and 2000 microseconds.



PAYOFFS/BENEFITS

Building a DIY PWM driver also offered a great opportunity for the electrical team to gain a little more experience in designing a functional solution, sourcing quality components, programming a microcontroller, and some human-machine interface development experience.

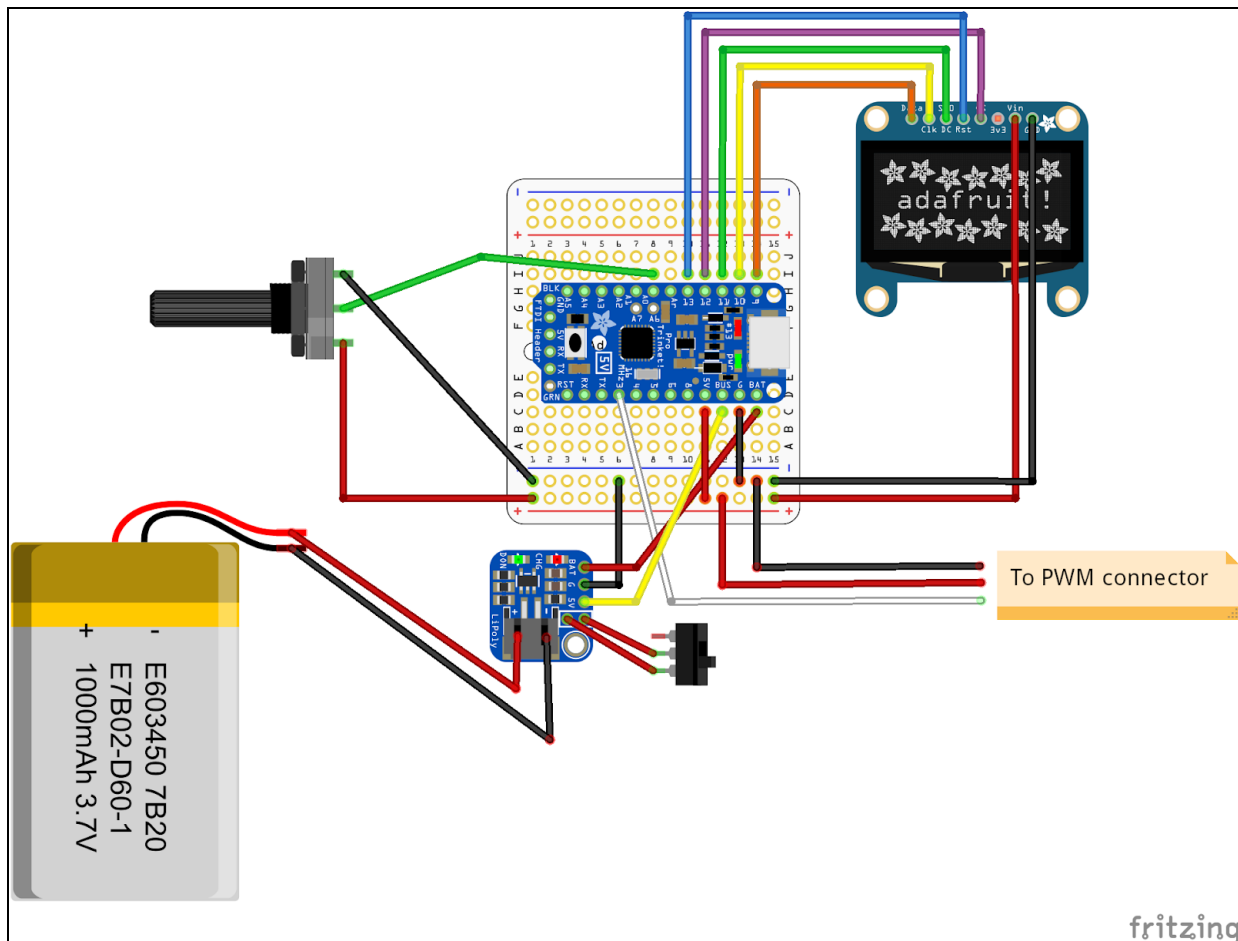
PARTS LIST

Note: Prices shown are date of document creation.

Item	Description	Vendor and part #	Cost
Pro Trinket 5V 16MHz		Adafruit 2000	\$9.95
Monochrome 0.96" 128x64 OLED graphic display		Adafruit 326	\$19.50
Lilon/LiPoly Backpack Add-On for Pro trinket		Adafruit 2124	\$4.95
Potentiometer knob - Blue		Adafruit 2048	\$0.50
Panel mount 1k potentiometer		Adafruit 1789	\$0.95

Lithium Ion Battery 3.7v 2000mAh		Adafruit 2011	\$12.50
Clear Case, Hammond 1591ATCL or similar		Digi-Key HM957-ND	\$6.46

CIRCUIT DIAGRAM

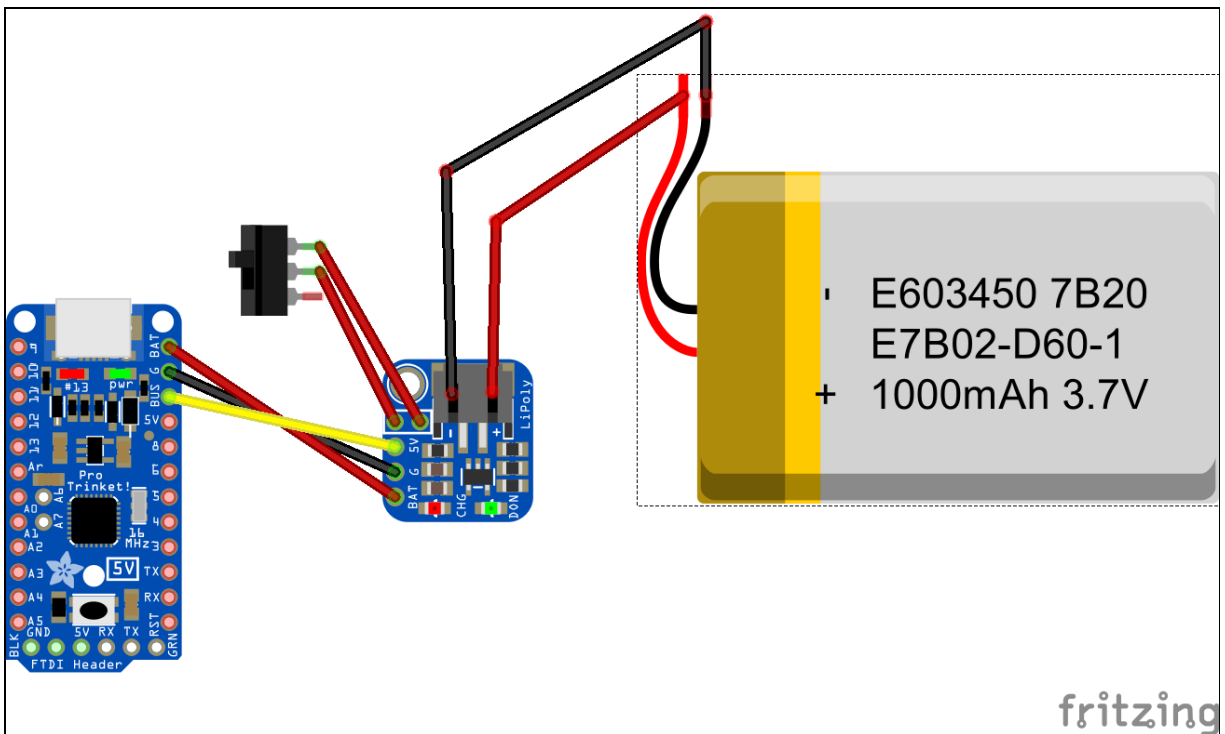


LiPoly/Lilon Backpack

The LiPoly/Lilon Backpack allows charging of the battery while the Pro Trinket is connected to 5V USB. When the battery is charged, unplugging the USB power from the Pro Trinket will automatically switch over to the battery.

The small Backpack board is designed to be mounted directly on top of a Pro Trinket if desired, so the BAT, G, and 5V pins connect to the corresponding BAT, G, and 5V pins on the Pro Trinket. An On/Off switch is connected to the 2 pins labelled “Pwr Switch”, and a LiPo battery is plugged into the LiPoly Backpack.

The default charging rate of 100 mA is for batteries less than 500mAh, so if you’re using a 500 mAh or larger battery, you can short the indicated pads on the back of the board by soldering. This puts the charging board into 500mA mode. See the Adafruit page for more details. (<https://learn.adafruit.com/adafruit-pro-trinket-lipoly-slash-liion-backpack>)



Monochrome OLED Display

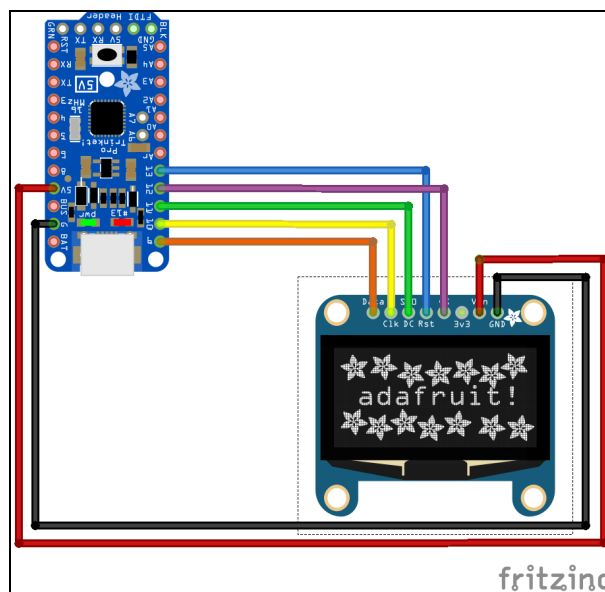
A 0.96" 128x64 OLED display is used for the user interface. To wire the OLED to the Pro Trinket using Serial Peripheral Interface (SPI):

OLED Pin	Pro Trinket Pin
GND	GND
Vin	5V
DATA	9
CLK	10
D/C	11
RST	13
CS	12

Demo programs are available for this OLED display from Adafruit, so it can be tested with the Pro Trinket before running the PWM Driver code.

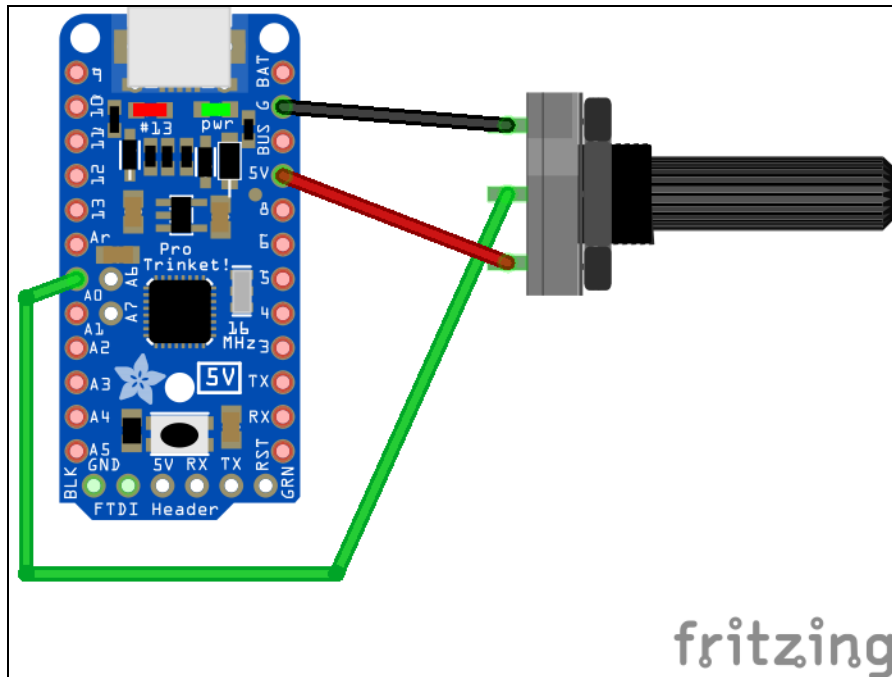
Note: Using this display requires that the Adafruit SSD1306 and Adafruit GFX libraries are installed into your Arduino IDE. See the Adafruit site for details.

(<https://learn.adafruit.com/monochrome-oled-breakouts>)



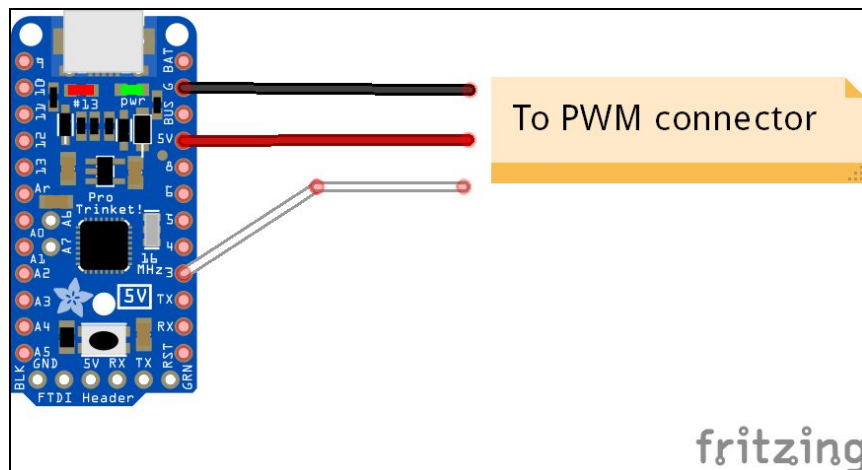
Potentiometer

Any basic potentiometer can be used. It is connected to the Pro Trinket by the potentiometer center pin (wiper) connected to analog input pin A0. The power and ground pins are connected to 5v and GND on the Pro Trinket.



PWM Cable

A PWM cable is connected to the Pro Trinket 5v, GND, and pin 3. This cable can be connected to any compatible motor controller or servo. Y-cables can also be used if you wish to drive 2 motor controllers at once.



OPERATION

Starting the PWM driver with the potentiometer knob fully counter-clockwise allows it to act as a Servo driver and provide the required signals that Servos expect. Starting the PWM driver with the knob vertical puts the device into Motor mode so that it can output a PWM signal used by motor controllers. Each of the two modes uses a slightly different display.

During operation using a motor controller, the raw potentiometer value is shown along with the percentage -100 to 100 and the PWM value of 1000 to 2000 (ms) for motors and from 0 to 180 for servos when using the Servo library. This value represents the duty cycle length in microseconds. Thus, 1000 is 100% reverse while 2000 is 100% in the forward direction. Stopping the motor at 0% input will show 1500 as a PWM value.

While displaying only the percentage (-100% to 100%) would have been sufficient, the potentiometer value and PWM values are shown so the user can see details of the potentiometer state and the values sent to the controller or servo.

Use of a PWM Y-cable can be used if more than one motor controller is to be tested. If motors are needed to operate in opposing directions, a simple crossover cable can be made using Anderson Powerpoles that inverts the red/black wires connected to motors. This is extremely helpful when testing dual-motor shooters as well as both sides of drive train.

CODE

This is the program used for our device. It's not perfect, nor is it intended to be. Feel free to modify it for your own use. Other displays can easily be used in place of the OLED.

```
/******  
  
    PWM Driver - A utility program that generates a PWM signal for the  
                purpose of sending test inputs to a motor controller or servo.  
  
    W. Bretton Jan 2016  
  
*****/  
  
#include <SPI.h>  
#include <Wire.h>  
#include <Servo.h>  
#include <Adafruit_GFX.h>  
#include <Adafruit_SSD1306.h>  
  
// These connections are for using SPI:  
#define OLED_MOSI  9  
#define OLED_CLK   10  
#define OLED_DC    11  
#define OLED_CS    12  
#define OLED_RESET 13  
Adafruit_SSD1306 display(OLED_MOSI, OLED_CLK, OLED_DC, OLED_RESET, OLED_CS);  
  
// Device will generate a signal that is capable of driving either a servo or a motor  
controller.  
#define SERVO_MODE 1  
#define PWM_MODE   0  
  
int    op_mode = PWM_MODE;  
int    cnt = 0;  
int    potValue;  
int    servoValue;  
int    motorSpeed;  
int    pwmValue;  
int    outPin = 3;  
Servo  myDevice;  
  
/* Uncomment this block to use hardware SPI  
#define OLED_DC      6  
#define OLED_CS      7  
#define OLED_RESET   8  
Adafruit_SSD1306 display(OLED_DC, OLED_RESET, OLED_CS);  
*/  
  
#if (SSD1306_LCDHEIGHT != 64)
```

```

#error("Height incorrect, please fix Adafruit_SSD1306.h!");
#endif

void setup() {
  myDevice.attach(outPin);

  display.begin(SSD1306_SWITCHCAPVCC);

  // Initialize OLED and print a welcome. Change this as required.
  display.clearDisplay();
  display.setTextSize(3);
  display.setTextColor(WHITE);
  display.setCursor(0,0);
  display.println("Triple");
  display.setCursor(0,32);
  display.println("Helix!");
  display.display();
  delay(3000);

  // Get the initial potentiometer value. If it's nearly fully counter-clockwise,
  // use this as a SERVO driver. Otherwise, assume motor controller.
  potValue = analogRead(A0);
  if (potValue < 100)
    op_mode = SERVO_MODE;
} //setup

// Main
void loop() {
  // get the raw pot value and do quick a few quick conversions
  potValue = analogRead(A0);

  // for display only, show the direction and speed as %
  motorSpeed = map(potValue, 0, 1023, -100, 100);

  // PWMs want to see a duty cycle between 1000 and 2000
  pwmValue = map(potValue, 0, 1023, 1000, 2000);

  // servos want to see 0 - 180
  servoValue = map(potValue, 0, 1023, 0, 180);

  if (op_mode == SERVO_MODE) {
    //put out value for a servo
    myDevice.write(servoValue);
    displayText4("Pot Value",
                String(potValue),
                "Servo",
                " " + String(servoValue));
    delay(15);
  }
  else {

```

```

        //output the value that the motor controller wants to see
        myDevice.writeMicroseconds(pwmValue);
        displayText4("Pot Value",
                    String(potValue),
                    "Motor/PWM",
                    String(motorSpeed)+ "/" + String(pwmValue));
    }

} // main loop

// Used to make the display output easier to format and use the 4 lines
void displayText4 (String lin1, String lin2, String lin3, String lin4) {
    display.clearDisplay();
    display.setTextSize(2);
    display.setTextColor(WHITE);
    //display.setTextColor(BLACK, WHITE);
    display.setCursor(12,0);
    display.println(lin1);

    display.setCursor(42,16);
    display.println(lin2);

    //display.setTextColor(BLACK, WHITE);
    display.setCursor(12,32);
    display.println(lin3);

    display.setCursor(6,48);
    display.println(lin4);

    display.display();
} //displayText4

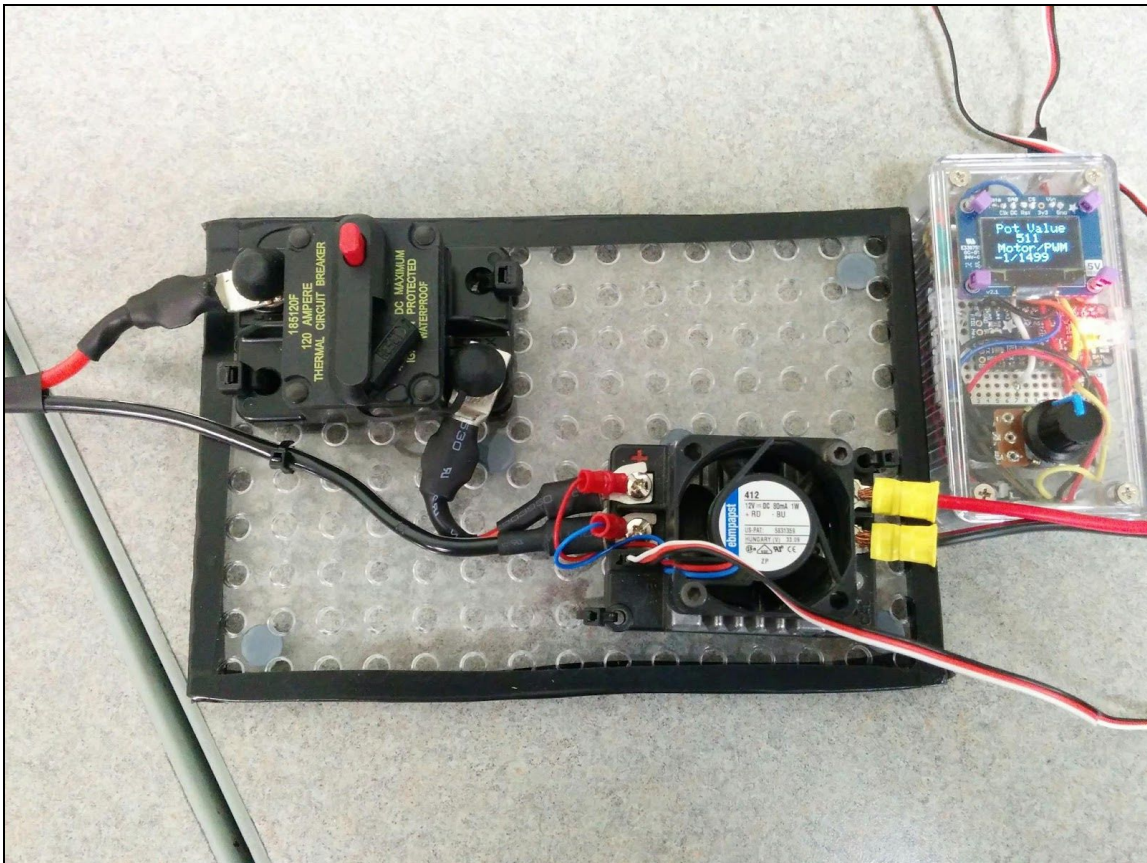
```

Note: While the writeMicroseconds() method would have worked fine when using either a motor controller or a servo, I chose to use the Servo.write(servoValue) method when driving a servo thinking that a 1-180 value would be easier to understand than sending 1000-1500 to a servo. This simply reflects a preference and can be changed if desired.

SETUP

A sample setup is shown below. A typical FRC battery can be used and connected to a motor controller with a FRC main breaker inline. The PWM driver is connected to the PWM port on the motor controller and the controller is wired to any motor that is to be tested.

We routinely connected two motor controllers to the 12V power source when needing to test 2 motors. A crossover power connector was used on one motor when two motors were to be driven in opposite directions.



Sample setup using a main breaker connected between a battery and a Talon motor controller. PWM Driver is connected to the Talon.

NEXT STEPS

Our team is planning for the next version of the PWM driver to have the following modifications:

- Breaker, 2 motor controllers, display, potentiometer(s) and various switches integrated into a single project box
- Toggle switches to provide inputs for two outputs
 - One potentiometer controls both motor controllers (same direction)
 - One potentiometer controls both motor controllers (opposing direction)
 - Two potentiometers control two motor controllers

Rather than having the PWM driver provide PWM cable outputs that require a separate setup of breaker, motor controllers, etc., the next version will be an integrated device that will use a 6 AWG Anderson connector to a battery and two Powerpole outputs to connect to two or more motors.