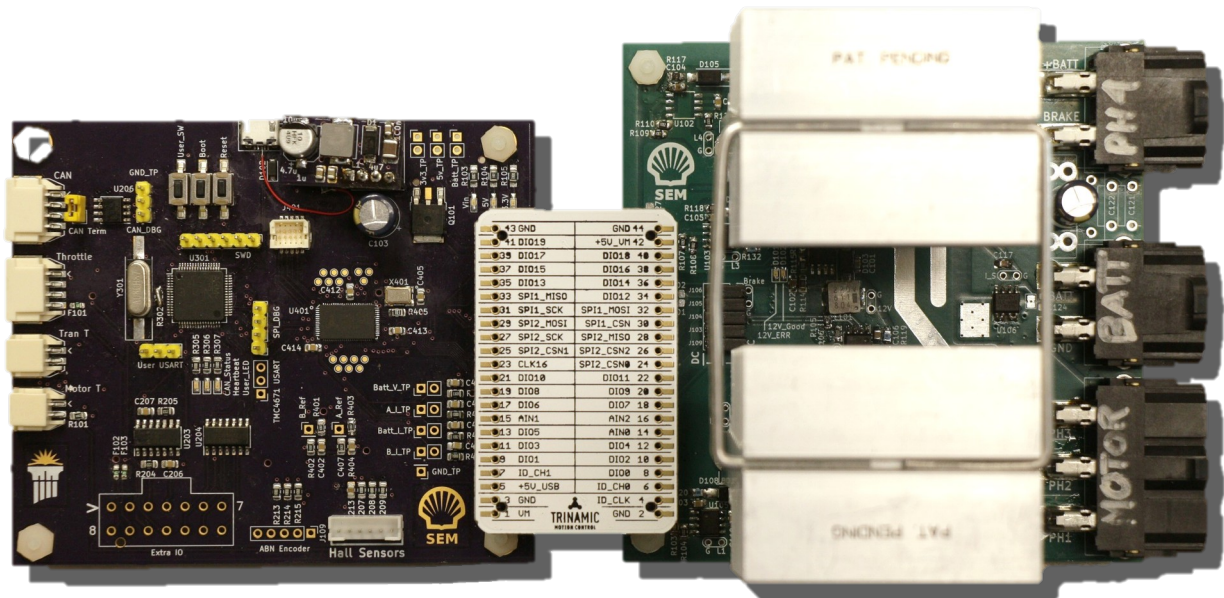


High Efficiency Motor Controller User Manual

May 3, 2019

Samuel Ellicott, Isaac Jones



Hardware Overview

Signal PCB

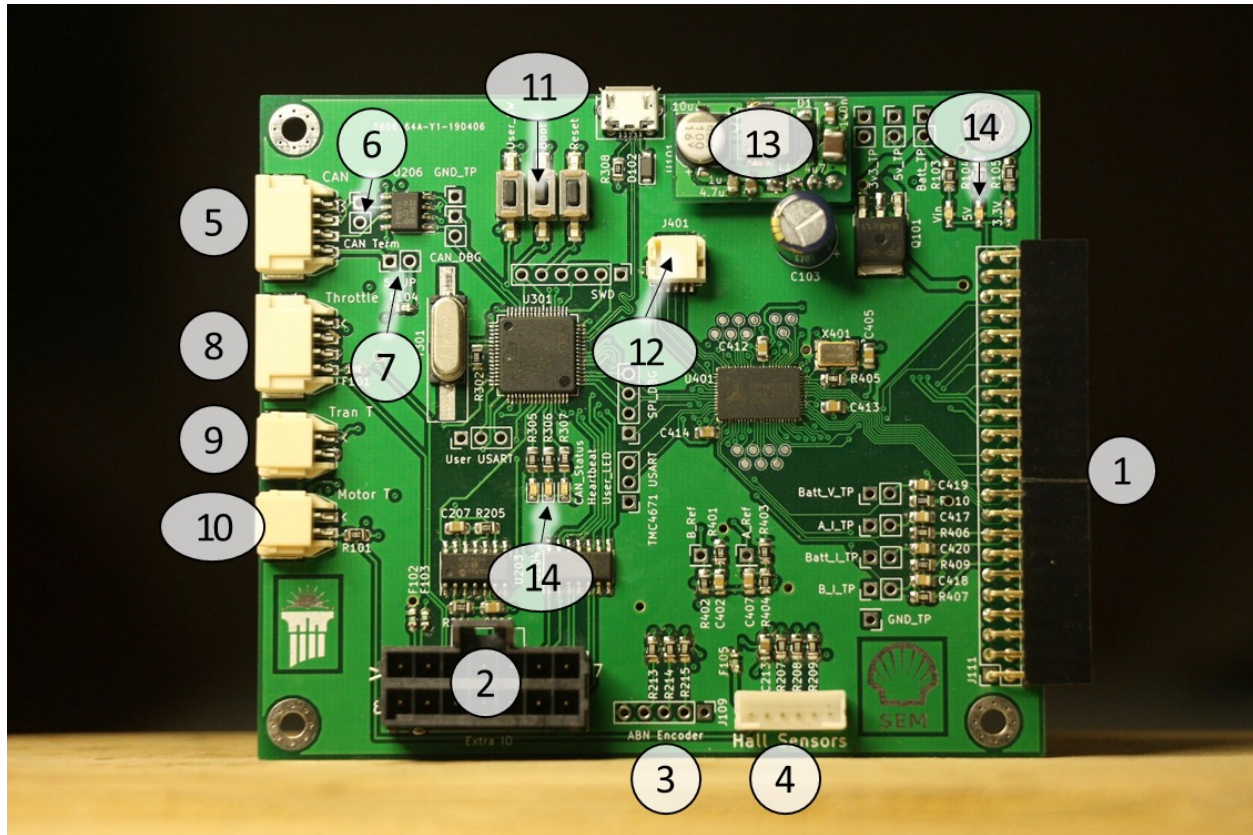


Figure 1: Signal PCB front view

Table 1: Important features of signal PCB

Reference Number	Description
1	44-pin header to power PCB
2	Extra IO
3	Motor digital position encoder connector (if equipped)
4	Motor digital hall sensor connector (also contains motor thermistor input)
5	CAN IO connector
6	CAN termination selector
7	CAN 5v power selector
8	Analog throttle input connector
9	Transistor analog temperature input connector
10	Motor analog temperature input connector
11	Pushbutton switches
12	Trinamic RTMI-USB-SPI debug connector
13	Power supply
14	Voltage regulator status LEDs
15	Microcontroller status LEDs

Power PCB

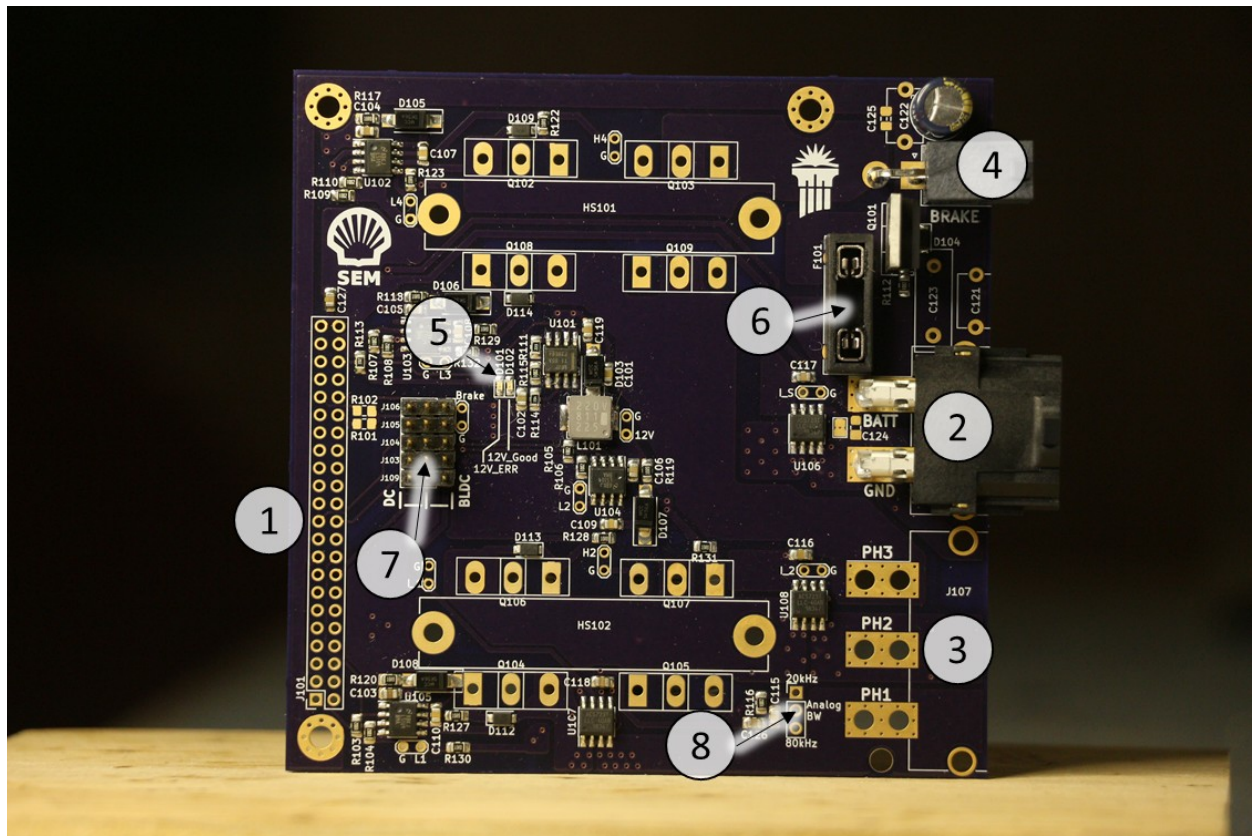


Figure 2: Power PCB front view (MOSFETs, heatsinks, motor connector J107, & header J101 removed)

Table 2: Important features of power PCB

Reference Number	Description
1	44-pin header to power PCB
2	Input DC power connector
3	Motor output power connector (not present in photo)
4	Resistive voltage brake/PH4 power connector (optional)
5	12V power supply status LEDs
6	Automotive blade fuse holder
7	DC/BLDC motor selector jumpers
8	Current sense bandwidth selector

Quick Start Guide

Safety: Always double-check wiring connections and controller menu settings before powering up.

We can't fix stupid. ☹

1. Signal Connections

a. For analog throttle

Connect a 0-5V analog voltage signal source to the signal PCB connector (Table 1 ref #8). Table 3 below shows the pinout for the analog throttle Molex connector. For the analog throttle, a low voltage corresponds to low speed, and near 5V corresponds to full speed.

Be sure to **initialize the control PCB software “Use Analog Throttle” parameter to ON**, as described in menu “General Settings:” below. Otherwise, the controller will not respond to analog throttle input. Also, **initialize the analog throttle range** using the menu prompts, described in 13 (“Throttle Setup:”).

Warning: never power up a controller in analog throttle mode without an analog throttle attached – doing so causes a floating analog input, which could cause the controller to engage a motor uncontrollably. Always initialize software and signals before attaching power connections!

Initialize the rest of desired motor settings, then **set “Enable Outputs” to ON**.

Save motor settings through the menu.

Table 3: Signal PCB analog throttle pinout

Pin Number	Function
1 (marked with ‘<’)	Ground
2	Throttle analog IN
3	5V supply
4	5V supply

b. For CAN connections

Connect the car’s CAN bus to the signal PCB CAN connector using Table 1 below as reference. Be sure to **disable analog throttle** in the control PCB software “Use Analog Throttle” parameter as described in menu “General Settings:” below; this will enable throttle control over CAN.

Warning: never power up a controller in analog throttle mode without an analog throttle attached – doing so causes a floating input, which could cause the controller to engage a motor uncontrollably. Always initialize software and signals before attaching power connections! Initialize the rest of desired motor settings, setup CAN using the motor controller software menu and setup CAN on your peripherals to communicate with the motor controller.

Set “Enable Outputs” to ON.

Save motor settings through the menu.

Table 4: Signal PCB CAN pinout

Pin Number	Function
1 (marked with ‘<’)	Ground
2	CAN low differential signal line

3	CAN high differential signal line
4	5V supply (enable with jumper “5v JP”)

2. Power Connections

- a. Connect car traction battery to power PCB through in-line fuse (40A) and a pre-charge resistor. The pre-charge resistor allows for the power PCB bulk capacitors to charge without blowing the fuse and without arcing.

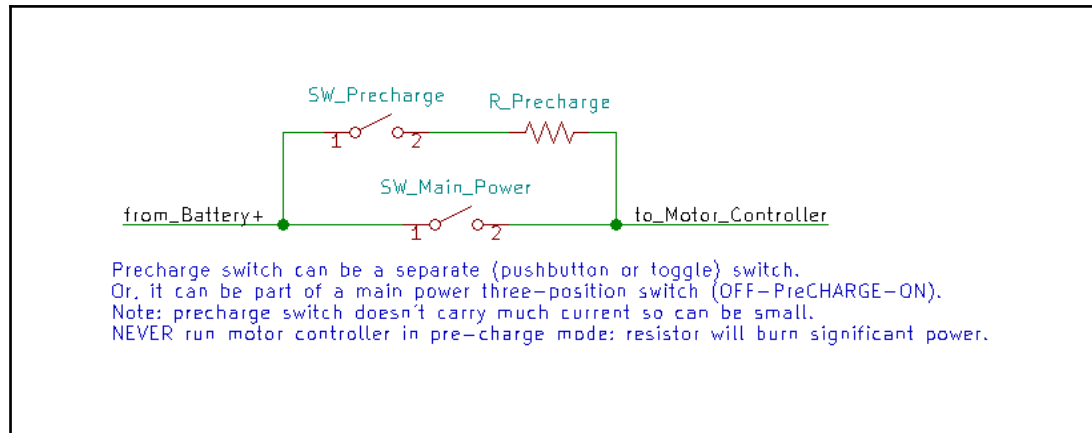


Figure 3: Precharge resistor allows motor controller to power up without danger of damage from inrush current

When turning ON main power to the motor controller, turn the power switch from OFF to Pre-CHARGE and wait for a second or two, then turn to ON.

Below in Figure 3 is a suggested circuit diagram for the pre-charge resistor power switch. We recommend a precharge resistance of $\geq 1000\Omega$ and a power rating $\geq 5W$.

- b. Brushless DC (3-phase AC) motor only:

This type of motor requires three (3) power wires – phases 1, 2, & 3. Simply connect the three motor wires to the power PCB through Molex crimp connector. Note: motor wire order is not important, as swapping two of the three wires simply reverses BLDC motor direction, and direction can also be changed via USB or CAN interfaces. Also connect the motor Hall-effect sensors to the “Hall Sensor” input on the signal PCB.

Be sure to configure the motor controller (via USB) for a BLDC motor with the appropriate number of magnetic poles.

c. Brushed DC motor only:

This motor controller offers two options for wiring a brushed DC motor, depending on power levels required.

The low-power configuration: set the DC/BLDC jumpers on the power PCB to “BLDC” mode and connect the DC motor positive wire to phase 1 and the negative wire to phase 2. (Polarity isn’t very important here as direction can be changed in software or via CAN.)

The high-power configuration: set the DC/BLDC jumpers to “DC” mode. This parallels phases 1 & 4 and phases 2 & 3 for driving high-current DC motors. See Figure 4 below for a suggested brushed DC wiring harness. Motor connections are: connect motor positive wire to both phases 1 and 4, and connect motor negative wire to both phases 2 and 3.

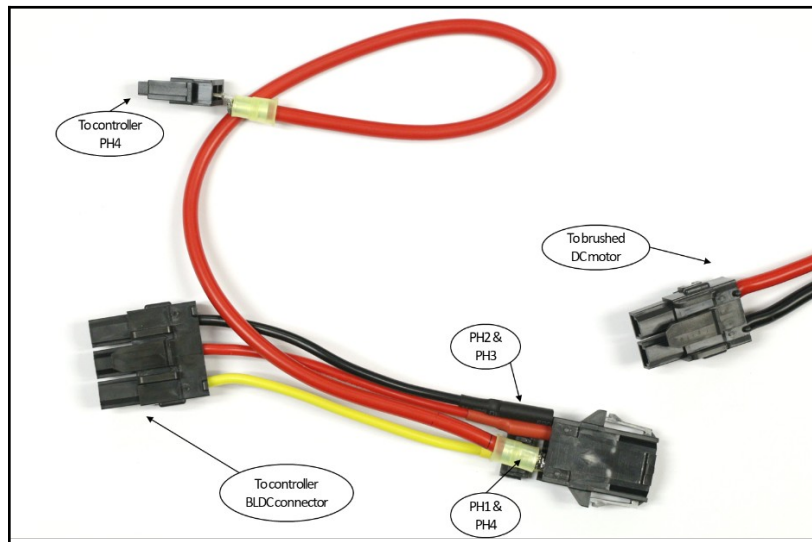


Figure 4: High-current brushed DC wiring harness

Be sure to configure the motor controller (via USB) for a brushed DC motor. Unless you have configured a position encoder/hall-effect sensor on your DC motor shaft, the **controller will work with DC motors only in “Torque” mode.**

This configuration disables the electrical overvoltage-protection brake.

d. Voltage-protection brake (optional):

You **MUST** connect a resistor brake to the power PCB connector (Table 2 ref #4) if there is a chance the DC supply to the motor controller is >50V, even in transient spikes. Examples of this situation include: if the motor controller is powered from a generator (instead of a battery), or if the motor controller is powered from a lab bench DC supply. These power sources may induce high voltage spikes (especially if you’re using regenerative braking), which will permanently damage delicate electronics on the motor control PCBs. **Safety first!**

Software Setup

Basic Setup

1. Install PuTTY

- Go to <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html> (or search for PuTTY on the internet)
- Download the alternative 64-bit putty.exe executable (Figure 5)

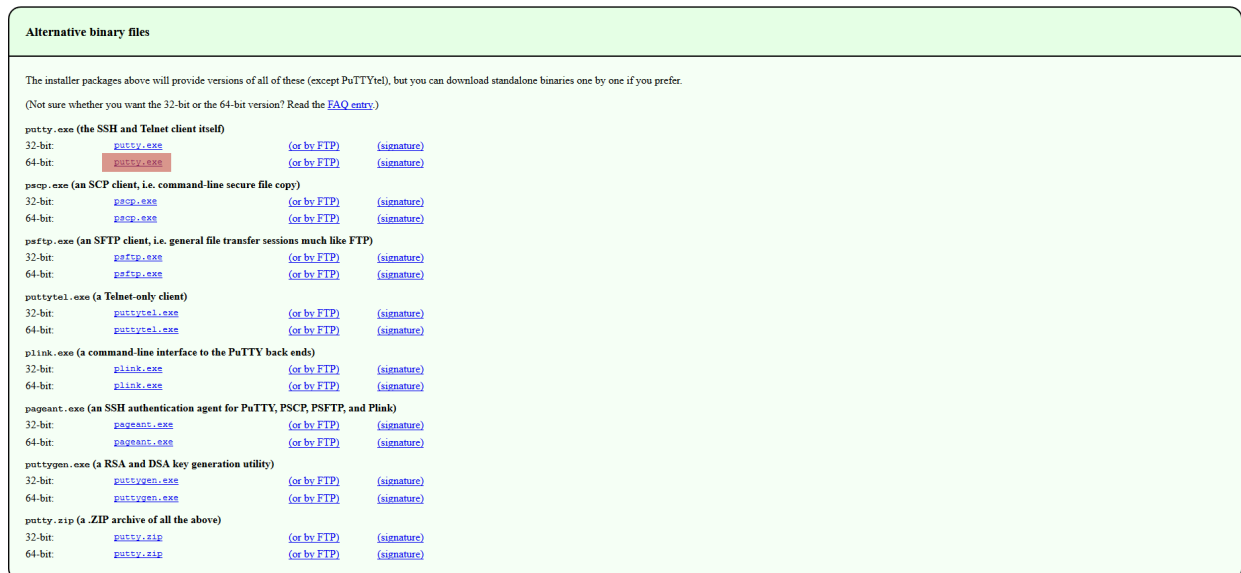


Figure 5: PuTTY Download webpage with correct download highlighted

- This version of the program does not need installed, just move the .exe file to your preferred directory.
- ### 2. Setup PuTTY
- Find the COM port number of the motor controller by looking in the Windows Device Manager. Admin privileges are not necessary. Figure 6 shows the device manager with the motor controller highlighted. If you are unsure what port the motor controller is using, disconnect and reconnect the motor controller and see what port it shows up on.

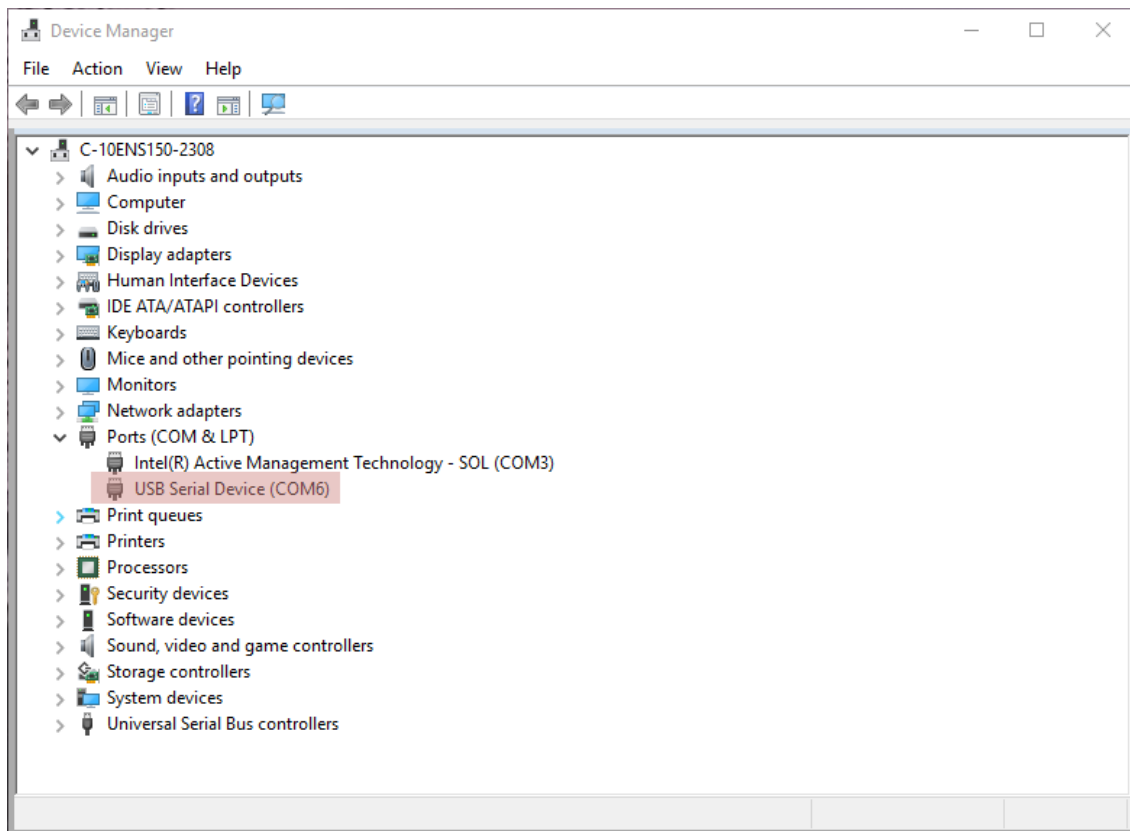


Figure 6: Device Manager with the motor controller COM port highlighted

b. Run PuTTY, you should see the following window

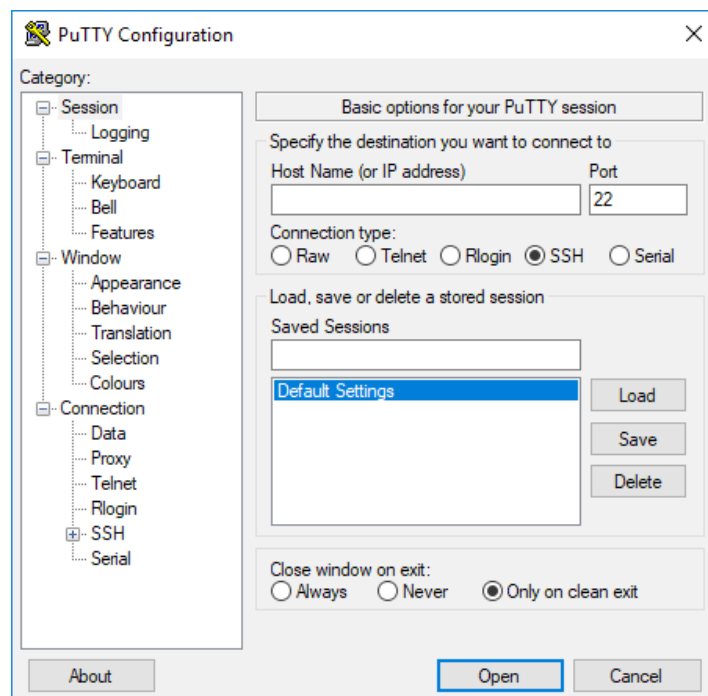


Figure 7: Default PuTTY Window

- c. Change the settings to match the following picture. The serial line should be the COM port determined previously.

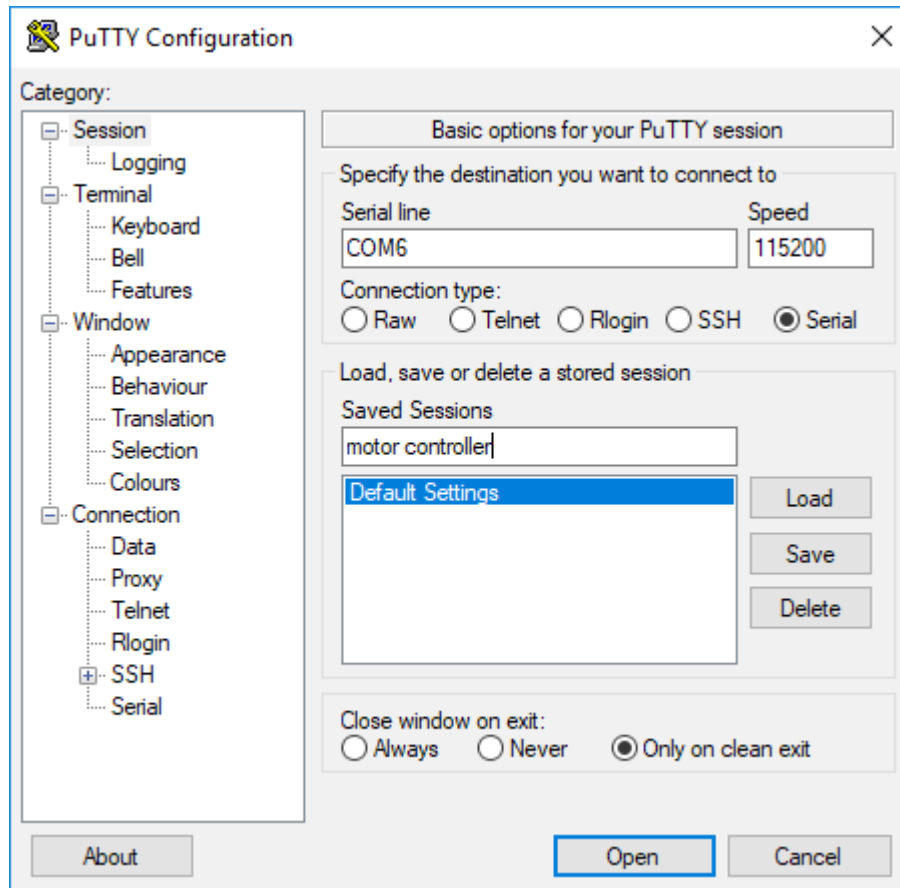
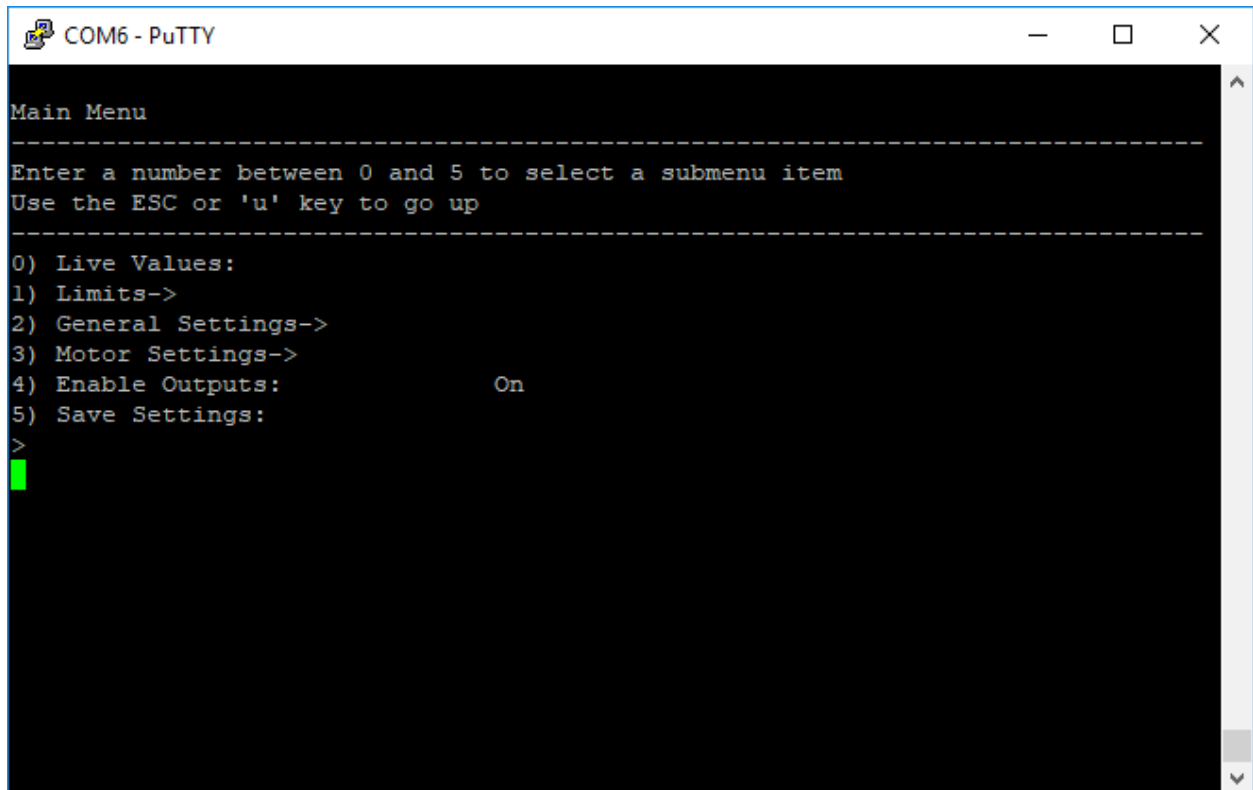


Figure 8: PuTTY setup

- d. Enter a name into the “saved sessions” bar and click the save button. In the image above I have entered “motor controller” but have not saved it yet. This step is optional, but allows for quick access of settings.
- e. Click the “Open” button to start the terminal. You should see the terminal shown in Figure 9
3. Using the menu
- Basic instructions for using the menu are given in the interface. Useful things to know are that the interface only accepts integer numbers, the ESC key, the ‘u’ key, and backspace. The enter key must be pressed after entering a number for the command to be accepted. After settings are changed, they will immediately effect the motor controller operation (unless otherwise noted). However the **changes will not be persistent across power cycles until they are saved to the flash memory.**
- a. Navigate to the menu item you wish to change (enter the number that corresponds to the menu item).

- b. Change the setting by entering the value you wish to change the item to, then press enter. If the item can only take on a limited set of values, they will be listed in the menu.
- c. After you are satisfied with the changes, save them using the “Save Settings” menu item in the main menu. This will make the changes persistent.



```
COM6 - PuTTY

Main Menu
-----
Enter a number between 0 and 5 to select a submenu item
Use the ESC or 'u' key to go up
-----
0) Live Values:
1) Limits->
2) General Settings->
3) Motor Settings->
4) Enable Outputs:      On
5) Save Settings:
>
█
```

Figure 9: Serial Terminal Main Menu

List of Menu Items and Descriptions

This section provides a reference list of the various options available in the menu system.

Main Menu

1. Live Values:

Display a list of current motor controller values in real-time.

- The current set-point (in mA or RPM depending on if the motor is in Torque or Velocity mode respectively).
- The motor velocity (in RPM). Note this may not work on a brushed motor if it does not have an encoder.
- The motor current (in amps)
- The battery voltage (in Volts).

Note that due to a hardware issue in the TMC4671, this might not produce the most accurate readings

- The battery current (in Amps).

Note that due to a hardware issue in the TMC4671, this might not produce the most accurate readings

- Motor temperature (in Celsius)
- Transistor/heat sink temperature (in Celsius). This thermistor input has not been calibrated, and has not been performing properly.

1. Limits Menu:

Maximum values for the motor controller to use during operation

2. General Settings:

Contains settings that we think will be accessed frequently, this includes the motor control mode and direction, along with CAN settings.

3. Motor Settings Menu:

Settings used to setup constants for the motor. Also includes PI controller constants.

4. Enable Outputs:

Enable or disable the high power outputs from the motor controller.

5. Save Settings:

Most of the settings changes in the menu take effect immediately, however **they must be saved in order to persist over a power cycle**. This menu item allows for the various settings to be saved.

Limits Menu:

Maximum values for the motor controller to use during operation

1. Up:

Go up one menu level

2. Max Current:
Maximum usable current in all modes of operation.
There is an internal limit to this value of 60000 mA (60A) however a practical limit is about 30A. Changing this value will affect how fast the car will accelerate, and limit the regeneration current if using regenerative braking. When in Torque mode the maximum analog throttle set-point is mapped to this value.
3. Max Velocity:
Maximum RPM limit for velocity mode. **Note this setting only takes effect in velocity mode.**
When in velocity mode, the maximum analog throttle set-point is mapped to this value.
4. Max Acceleration:
This value is currently unused. To limit acceleration use the maximum current setting.

General Settings:

Settings that will probably be accessed fairly frequently. This includes the motor direction, control mode, and CAN settings.

1. Up:
Go up one menu level
2. Motor Control Mode:
Select between Torque and Velocity modes of operation.
 - The torque mode controls the current flowing through the motor, and the RPM varies depending on load. In order to perform regenerative braking, the motor direction must be reversed and a negative current used for a set-point (until the car has stopped).
This mode should be used for brushed DC motors.
 - Velocity mode controls the motor RPM and the current is only limited by the maximum current.
This mode will be the easiest to enable regenerative braking in, as it performs that function simply as a consequence of the motor control loop. Decreasing the velocity set-point will result in regenerative braking until the new set-point is reached.
3. Motor direction:
Control the motor direction. Forward or Reverse.
4. Motor controller CAN ID:
Set the base ID for the motor controller this will also change the IDs which the motor controller sends and receives data on.

This menu item will also list the various IDs that the controller is set to send and receive on.

5. Throttle CAN ID:
Set what CAN ID to listen for set-point commands on.
6. Use Analog Input:
Select whether to use CAN based set-point input or take input from the analog throttle.
7. Throttle Setup:
Find the minimum and maximum values for the analog throttle input. This is used to calibrate the analog throttle for mapping it to set-points. To perform this operation enter the menu item, put the throttle at its minimum value, then enter 1. This will set the minimum throttle value. Then put the throttle at its maximum range; while holding it there, enter 1. This will set the maximum throttle value. Now, when using the analog throttle input, the minimum value will be mapped to a zero set-point, and the maximum value to the maximum set-point.

Motor Settings:

Settings used to setup constants for the motor. Also includes PI controller constants.

1. Up:
Go Up one level.
1. Motor Type:
Select between using a brushless DC motor (BLDC) or a brushed DC motor. Note that the hardware may need to be configured differently depending on the desired setup.
2. Pole Pairs / KV rating:
Select how many pole pairs the BLDC motor has. This is used to calculate the proper RPM for the motor. If using a brushed motor this sets the motor KV rating. As of right now this information is not used in brushed DC mode, but could possibly be used in the future to get an approximate speed from a brushed motor.
3. PI Constants menu:
Various proportional (P) and integral (I) constants for the motor controller IC control loops. These are set to their default values. You probably should not change these unless you know what you are doing. By experimentation we determined that the integral constant dominates the controller. Default value is 255.
4. Open Loop Settings:

Values for running the motor in open loop mode. As of the writing of this manual, these settings are not used.

5. Hall Effect Settings menu:

Menu for changing Hall Effect settings for a BLDC motor. **These values should not be changed if the motor is working correctly** (i.e. spinning). If these values are set wrong, then the motor will not spin. Unfortunately different motors need different settings here. The best way to set these currently is to use the Trinamic IDE to find the correct values, then copy them in manually to the settings.

6. Hall Effect Auto Setup:

This menu item is a placeholder for some code to automatically determine (and set) the correct values for the Hall Effect settings menu. Due to time limitations, at the time of writing this manual this feature has not been implemented.

CAN

There are a number of different messages that are sent out over CAN. Table 6 lists the messages and ID's used by the motor controller. Note that the base ID and the throttle ID can be set by the user from the motor controller menu.

A note about the formatting of the CAN messages, the first byte of each CAN message contains formatting information (from the C++ CAN library). In most cases this can be safely ignored and the data extracted from the remaining bytes of the message. Table 5 shows the byte-level arrangements of the message types used by the motor controller. The Int16 Arr data format can send between 1 and 3 integers, if the data to be sent is a 2 integer array, then the 3rd integer will not be sent in the message (replace with XX).

Table 5: CAN Message Formats (XX is an unused byte and will not be sent)

Message Format	CAN Data Bytes							
	0	1	2	3	4	5	6	7
UInt8	0x00	data(7:0)	XX	XX	XX	XX	XX	XX
Int32	0xA0	data(7:0)	data(15:8)	data(23:16)	data(31:24)	XX	XX	XX
UInt32	0x80	data(7:0)	data(15:8)	data(23:16)	data(31:24)	XX	XX	XX
Float	0xC0	data(7:0)	data(15:8)	data(23:16)	data(31:24)	XX	XX	XX
Int16 Arr	0x60	data0(7:0)	data0(15:8)	data1(7:0)	data1(15:8)	data2(7:0)	data2(15:8)	XX

Table 6: Motor Controller CAN IDs (Transmit is from the motor controller)

Name	ID	Format	Direction	Description
Base ID	Base ID	RTR	Receive	Send out current motor controller data on RTR request.
Temperature	Base ID + 1	Int16 array	Transmit	Sends the motor and transistor temperatures in a single CAN message (in that order)

Motor RPM	Base ID + 2	Int32	Transmit	Sends the motor RPM
Motor Current	Base ID + 3	Float	Transmit	Sends the motor current (in amps)
Battery Voltage	Base ID + 4	Float	Transmit	Sends the battery voltage (in volts)
Battery Current	Base ID + 5	Float	Transmit	Sends the battery current (in amps)
Motor Direction	Base ID + 6	UInt8	Receive	Listens for the motor direction. 0: Forward 1: Reverse
Control Mode	Base ID + 7	UInt8	Receive	Listens for the control mode. 0: Torque (current) 1: Velocity (RPM)
Max Current	Base ID + 8	UInt32	Receive	Listens for the maximum current (in mA)
Max RPM	Base ID + 9	UInt32	Receive	Listens for the maximum RPM (only effective in velocity mode)
Max Acceleration	Base ID + 10	UInt32	Receive	Listens for the maximum acceleration (Not implemented, use the maximum current to limit acceleration)
Enable Pulse	Base ID + 11	Any Data	Receive	Listens for any valid data. A signal must be received every 500ms to keep the motor controller active in CAN mode.
Setpoint	Throttle ID	UInt32	Receive	Listen for a setpoint. Torque mode: Motor current in mA Velocity mode: Motor RPM

The “Enable Pulse” CAN ID is a safety feature we implemented for the CAN mode of operation. It is used to disable the motor controller, if for some reason during a race the CAN-based throttle was disconnected from the motor controller or the CAN bus is compromised. Because the motor controller will maintain a set-point until it receives another command, there needed to be a method to turn off motor power if the CAN bus goes into an error state. We thought that the best method to determine if the CAN bus is still functioning properly would be to use an ID that sends out a pulse (some piece of data) at least once every 500ms. If the motor controller can still “hear” this ID, it means that the bus is still working properly, and the set-point is still valid. If for some reason the motor controller cannot hear this ID then power to the motor is shut off.

Installing the STM32 development Toolchain

This is a quick guide on how to install a toolchain (compiler, debugger, program installer, and necessary extra tools) for STMicroelectronics STM32 microcontrollers. This is a guide for Windows computers, I am also assuming that you have, or can get, administrative privileges for the computer you are using.

This guide will be broken into three parts

- GNU toolchain (GCC, Git, Make) - applicable to all ARM Cortex Processors
- STM32 tools (STM32Cube, STM32CubeProgrammer) - applicable to STM32 parts
- Extra tools (VSCode and extensions) - My preferences for development tools

GNU Toolchain Installation

1. Download Packages
 - a. Download the latest release of the GNU Arm Embedded toolchain from developer.arm.com
 - b. Download Git from git-scm.com
 - c. Download Make from gnuwin32.sourceforge.net (You can either download the complete package setup file, or the binaries zip and the dependencies zip).
2. Install Packages
 - a. Install the GNU toolchain
 - b. Be sure to add the toolchain to your system path (this is an option during installation)
 - c. Install Git
 - d. If you chose to download the complete Make setup file, install it normally, then add the executable location to your system path.
 - e. If you chose to download the two zip archives, unzip them.
 - i. Find the location of your Git installation
 - ii. A single user installation of Git is in your Windows home directory 'C:\Users\<username>\AppData\Local\Programs\Git'
 - iii. Copy the 'bin', 'man', and 'share' directories from the unzipped make-binaries folder to the 'usr' folder in the Git directory.
 - iv. Copy the 'bin' and 'share' directories from the unzipped make-dependencies folder to the 'usr' folder in the Git directory.
 - f. Make will now be usable from within "Git Bash"

STM32 Tools Installation

1. Download Packages (Download links are at the very bottom of the page. Click "Get Software")
 - a. Make a [STMicroelectronics account](#)
 - b. Download [STM32CubeMX](#)
 - c. Download [STM32CubeProgrammer](#) (This includes drivers)
2. Install Packages

- a. Unzip packages
- b. Install like normal (double click .exe file in unzipped folder)

Extra Tools installation

1. Download [Visual Studio Code](#)
2. Install Visual Studio Code
3. Configure Visual Studio Code
 - a. Open visual studio code
 - b. In the extensions menu on the left hand side install:
 - c. C++ intellisense (from Microsoft)
 - d. Cortex - Debug
 - e. Vim - if you want vim shortcuts when editing
4. Restart visual studio code

Appendix: Useful Links

Github hardware repository: <https://github.com/HEEV/motor-controller-2018>

Github software repository: <https://github.com/HEEV/motor-controller-2018-sw>