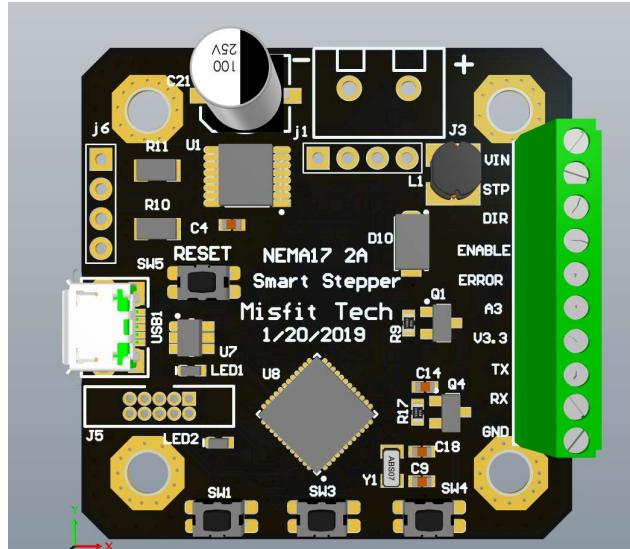


Smart Steppers



The MisfitTech' smart steppers turn a stepper motor into a closed loop servo.

The Smart Steppers uses a high-speed 32-bit processor combined with a 14bit (16384 pulse per revolution) encoder and motor driver to provide a closed loop PID control of the stepper motor.

Features

- Open source hardware
- Open source firmware
- Fixed point math for highest reliability
- High speed digital step and direction interface
- 256k flash, 32k SRAM
- Arduino compatible firmware
- Positional PID control
- Velocity PID control
- Hybrid Positional PID control
- On board Motion Planner for standalone operation
- Level shifted logic interface to work with 3.3V and 5V motion controllers
- Extra serial (UART) interface for connecting to external controllers
- Optional LCD for easy setup and configuration
- Command line interface through USB port for configuration and firmware updates

Specifications

The following specifications are based upon firmware version 0.39 and newer.

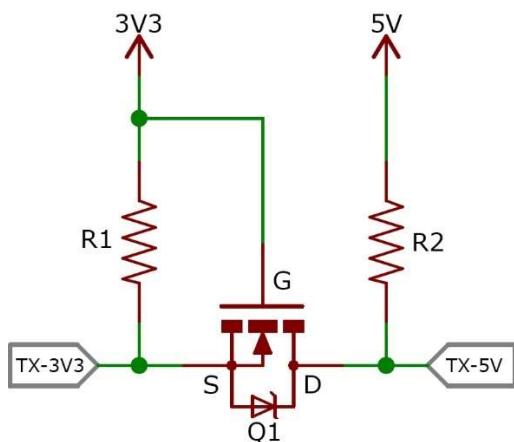
Absolute Maximum Ratings

Characteristic	Symbol	Notes	Rating	Units
Supply Voltage	V_{BB}		35	V
Motor Current (NEMA 17)	I_{OUT}		± 2	A
Motor Current (NEMA 23)	I_{OUT}		± 3.2	A
Operating Ambient Temperature	T_A		-20 to 80	C

Electrical Characteristics

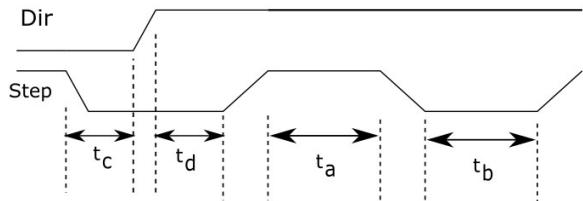
Characteristic	Symbol	Min	Typical	Max	Units
Motor Supply Voltage	V_{BB}	12	24	30	V
Logic Level	V_{IN}	3.3	-	5.0	V
Logic Input Voltage	$V_{IN}(1)$	1.6	-	V_{IN}	V
	$V_{IN}(0)$	0	-	1.5	V
Logic Input Current	$V_{IN}(1)$	0	-	0	mA
	$V_{IN}(0)$	-0.7	-	-1.3	mA

The logic pins (Step, Dir, Enable, TX, RX, Error) are level shifted to the V_{IN} . The level shifter works through a circuit like:



On the boards dated before 1-20-2019 the resistors were 10k ohms, however due to many customers having issues with noise coupling into the step and direction pins the resistors on the boards post 1-20-2019 have been changed to use 2.7k pull-ups.

Interface Timing



Time Duration	Symbol	Minimal	Maximum	Units
Step minimal, HIGH pulse	t_a	200	-	ns
Step minimal, LOW pulse	t_b	200	-	ns
Direction Setup Time	t_c	200	-	ns
Direction Hold Time	t_d	200	-	ns

The Smart Steppers have been tested to have no lost steps using the timing above.

When the smart steppers are connected to a controller with long wires for step and direction connections, the wire lengths can add capacitance to the signals, and you may need to increase the rise time and fall times to compensate for the wire capacitance.

Having a high current push pull driver on the controller for the step and direction pulses can also help mitigate wire capacitance and wiring introduced noise.

V_{IN} Pin

The Vin pin sets the interface logic level to the controller, for 5V controllers it should be connected to the 5V signal from the controller. For a 3.3V controller it should be connected to 3.3V.

Step Pin

A rising edge on the step pin causes the smart stepper to move one step. The edge selection of the step pin is fixed in firmware.

Direction Pin

The controller should normally toggle the level of the direction pin when the step pin is at a logic low state to maximize the setup and hold timing.

The direction of rotation for a logic high level on the direction pin can be changed through the LCD menu or through the *dirpin* command.

Enable Pin

The enable pin is sampled at 6000 times a second. Hence it is possible to have the stepper motor move if steps occur between the time the enable pin is asserted and the next time it is sampled in firmware.

The active logic level for the enable pin can be changed through the LCD menu or through command line interface.

Error Pin

The error pin will be asserted when the Smart Stepper sees a difference between the desired motor shaft angle and the actual motor shaft angle.

The amount of error that will cause the error pin to be asserted can be changed through command line interface.

A3 Pin

The A3 pin is a 3.3V logic level pin with no logic level shifters. This pin can be used as an analog input or for other purposes in the firmware. If you are not writing customer firmware for the Smart Steppers it is recommend that this connection be not used.

3.3V Pin

This is a 3.3V output from the smart stepper, if you are using the smart stepper without a motion controller then you might want to connect the 3.3V pin to the Vin pin to minimize chances of noise on the Step/Dir/Enable inputs.

TXD/RXD Pins

The TXD/RXD are pins connected to an internal UART. By default, in the firmware the TXD will output a serial diagnostic message stream for debugging. However, the RXD/TXD can be used by an external host microcontroller to send commands to the smart stepper just like the USB commands described below.

GND Pin

The ground (GND) pin should be connect to the ground on the controller board for a current return path. Note on the smart stepper board this is ground pin is electrically tied to the negative side of the motor power connection.

Operational Details

The smart steppers can do many things including closed loop position control, and closed loop velocity control. Closed loop torque mode is currently under development.

Position control modes

In the position control modes, the smart stepper counts the number of forward and reverse steps seen on the step pin. For example if the motor is a 200 steps per rotation motor with 16x micro stepping and

we have had 1800 clockwise steps, then we know the shaft angle should be $360/(200*16) * 1800 = 202.5$ degrees. The encoder allows us to measure the actual motor shaft angle, for example it could measure our actual angle is 202.2 degrees. Then we would have an error of 0.3 degrees. This means we need to apply some current (force) to the motor shaft to make it move. To do this we use the current error, past error, and rate of change of error to determine what force to apply to motor. To put this in a mathematical equation we have:

$$u(t) = K_p e(t) + K_I \int_0^t e(t) dt + K_d \frac{de}{dt}$$

Where $u(t)$ is the force (current) we estimate we will need to apply to move the motor at this point in time, $e(t)$ is the error, and K_p, K_I, K_d are the user adjustable PID terms. By adjusting (tuning) the three P,I,&D terms control over how quick the motor responds and how it settles and how much noise it has can be adjusted.

Note there is a lot more detail to the PID algorithms than just the above equation, including integral roll up problems, and numeric precision issues. The smart stepper firmware has been carefully designed to manage these issues. For example, the calculations are done with fixed point math instead of floating point such that the numeric precision errors are controlled. Where floating point math is used the firmware is careful to make sure it does not cause any precision issues. For more about floating point math errors see:

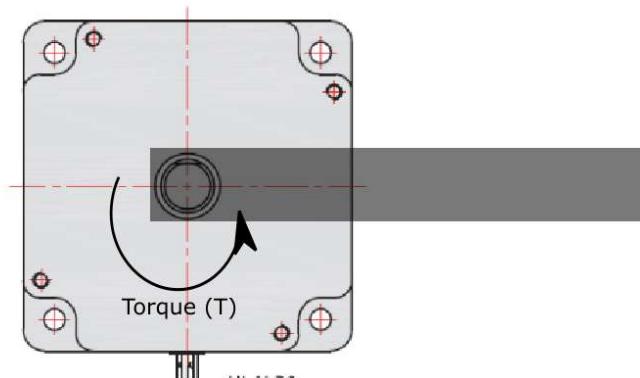
https://en.wikipedia.org/wiki/Floating_point_error_mitigation

The process of adjusting the PID terms is beyond the scope of this document and it is recommended that a web search on “PID tuning” be done to learn more.

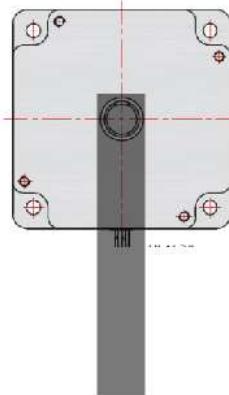
The smart steppers implement two different PID modes one is the traditional PID control loop which is called the positional PID (pPID). The second is a simple hybrid PID (sPID or simple).

Simple PID Mode

The simple PID loop is a hybrid of the PID and traditional stepper motor mode of positioning. This mode came about because of a very simple problem of motor holding a bar horizontal as shown below:



For the motor to hold the bar horizontal the smart stepper must apply some non-zero torque (force/current) to the motor. When the bar is vertical however the motor needs to apply zero torque to keep the bar in position.



This means the load on the motor is varying based on the shaft angle. Varying loads are very hard to make stable with traditional PID loops. A traditional stepper motor is very stable with this type of load as they can control position using the stepping nature of the motor. However, the traditional stepper motor driver provides full motor current all the time, which causes motor to get hot and you do not have any closed loop control of the motor.

The smart steppers implement a hybrid or simple PID, by allowing the user to specify a holding torque (hold Current) that will be the minimal current applied to motor. For example, if we require only 100mA of motor current to hold the shaft above horizontal then we would set our hold current to 100mA. The smart stepper would still run PID mode to get the shaft to correct angle, which could involve higher currents on the motor. However, it will not put less than 100mA of current into the motor. This means that even when the shaft is vertical there is still 100mA going into motor.

From testing it has been found that using this simple PID mode with stepper motors is very powerful as it makes the tuning of the PID less likely to oscillate and thus is more stable while providing most all the benefits of closed loop control. The only down side is that in some situations like the bar being vertical there is some wasted energy and heat. However, it is far less than traditional stepper drivers waste and worth the cost for a more stable system, therefore the smart stepper ship with this as the default mode of operation.

Velocity PID Mode

For some applications people want the motor to run at a constant velocity. To do this the smart stepper has implements a PID mode where the error term is based on the velocity. The velocity PID mode works with high rotational speeds (high RPMs) but can have problems at low RPMs. Specifically the control loop runs at 6000 times a second, so if you are running at slow speeds say 1 RPM then each cycle of the control loop the change in the motor shaft is 0.001 degrees. This is well below the accuracy we

can expect from the encoder after calibration and thus the velocity PID controller would not have smooth motion but more of jerky motion.

To get around this problem the smart stepper has implemented a motion planner and the move command. Specifically, if you want to turn 10 rotations clockwise at 1 RPM through the command line interface you can issue the move command with the number of degrees to move and the RPM which to move, 'move 3600 1'. The move command then moves the motor at the required number of degrees at the required speed by breaking up the movement into smaller periodic movements which causes smooth low speed motion.

Torque PID Mode

At the time of this writing there is no torque PID mode. Each time someone has asked for a torque-based mode of operation, it has been found that they actually wanted a position based PID with the ability to limit the maximum torque, which can be done by using the positional PID modes and setting the motor's hold current and maximum currents.

If you do need a torque PID mode or any other mode not implemented the firmware for the smart stepper is open source and thus you can change it for your needs.

Speed

The maximum speed of a motor using the smart stepper has several different factors. The first thing to understand is acceleration that is you have some mass the motor is moving and so you need a force to cause the mass to accelerate to the desired speed, ie $F=ma$. Like a car with a big engine versus a car with a small engine, the acceleration is lower when your motors can not apply the needed force to accelerate. The maximum current on the smart steppers limit the maximum force, but there is also a time aspect to this force. The smart steppers are driving inductive loads which resist the changes in current. To change the current in the motor faster you need higher voltage. Therefore, it is best to use the highest voltage power supply you can on the smart stepper. For example, if possible, use a 24V power supply versus a 12V one.

Note the smart stepper supply voltage has nothing to do with the maximum motor voltage. The smart stepper will limit the current into the motor (see the hold and motor current settings) which will in turn limit the maximum voltage on the motor.

With the smart stepper's control loop running at 6000 samples a second and with the maximum of one step per sample, then with a 1.8 degree per step motor we in theory can run at $6000*1.8/360=30$ rotations per second or 1800 RPM. However due to the limits in the rate of change of the current and load on the motor the speed will normally be lower than this.

Phase Advancement

An experimental feature in the firmware for the smart stepper is to monitor the velocity of the motor, and the advance the phase of the current to the motor. The basic idea is that if we are running at 1800 RPM then by the time, we measure the position of the motor shaft using the encoder the motor shaft has already moved 1.8 degrees. If we look at the velocity of the motor, we could predict where the motor shaft will be by the time the control processing is done and basically skip steps on the motor.

Experimental code has been developed for this and in testing has allowed a stepper motor to run at 20,000 RPMs. However, some users have reported errors when the phase advancement feature is

enabled and therefore it has been turned off in the stock firmware. If you want to help with the development of this feature, please let us know.

In theory the phase advancement can provide higher torque from the stepper motor at higher RPMs and thus increase the motor speed in real world machine operation.

Installation

Attach Magnet to back of stepper motor

The NZS feedback loop is done by measure the rotation of the magnet included with your NZS. This magnet needs to be connected to the shaft on the stepper motor. This is done by using epoxy or super glue. The magnet will be glued to the back of the stepper motor on the shaft, it is recommended for the maximum accuracy that a nylon washer be glued between the stepper motor and magnet.

The nylon washer between the motor shaft and the magnet reduces the magnetic loss through the motor shaft, thus increasing the magnetic field for the encoder to read. The magnet shipped with the NZS is a “strong” magnet and will work without using the nylon washer between motor and magnet but the increased magnetic field with nylon washer is better for encoder.

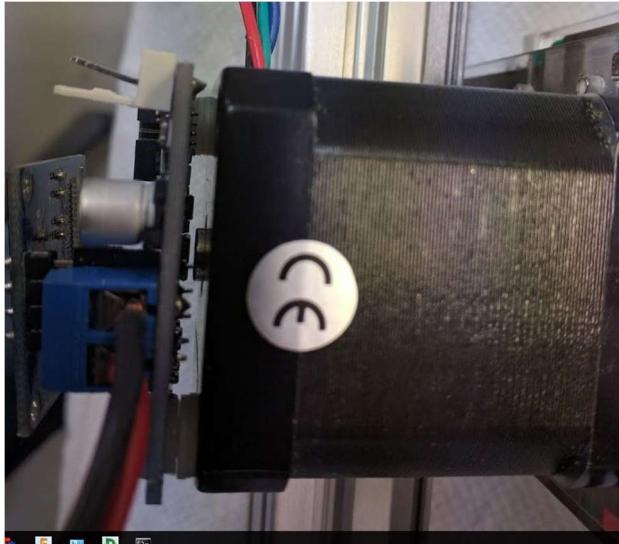
Care should be taken to try and center the magnet on the shaft.



The glue should be allowed to dry before proceeding to attach PCB board.

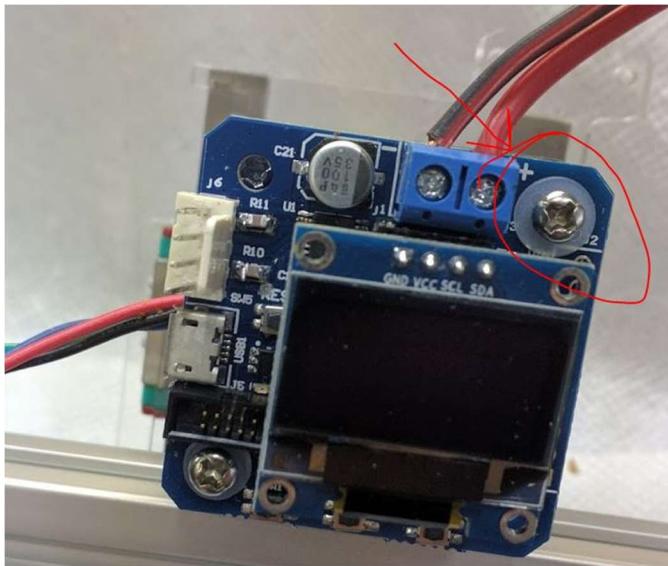
Attaching PCB

The next step is to remove the four screws on back of the stepper motor in prep for installing the PCB board. Then using the screws and nylon washers included with the NZS attach the PCB to back of the motor leaving a small gap between the top of the magnet and the sensor IC chip as shown below.



Please note that different stepper motors might require different length screws for attaching the NZS to motor. The kit comes with 40mm long screws but if you need a different length screw please contact us and we will try and get you the right length screws for your motor.

The PCB design layout has ground pads such that the motor housing is grounded to the PCB, this reduces the noise and possibility of ESD

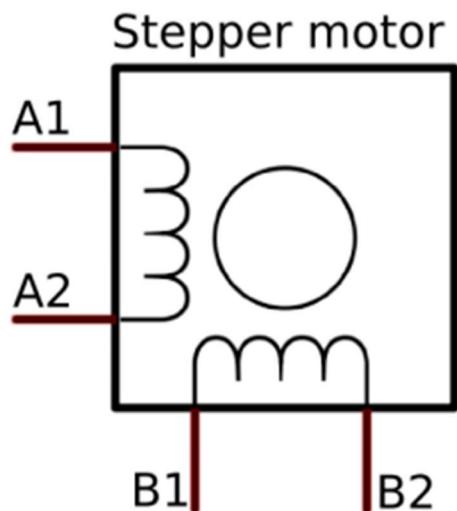


Connecting Power

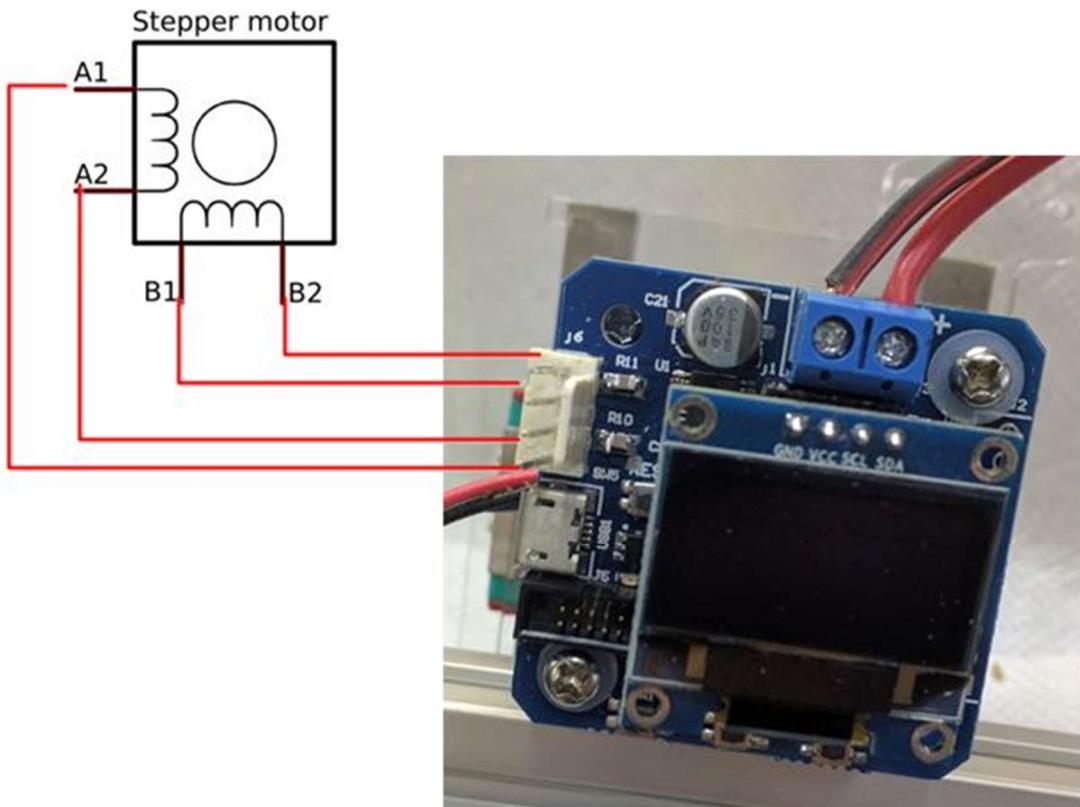
The blue screw terminals are for the Motor Power connections. This power is normally 12V or 24V and should not exceed 40V. The positive terminal for the power connection is closest to the mounting screw, the negative is closest to the capacitor can. Please note that wiring the polarity of the motor power in reverse will fry the board.

Motor Connection

The NZS is designed for dual phase motors, ie stepper motors with 4 wires. The four wires on the stepper motor are connected internally to two sets of coils as shown below:



The easy way to determine which wires are which is by using an ohm meter, if there is resistance between two wires then the two wires are connected to the A or the B coil. The wiring to the NZS should be done as shown below.



It does not matter which coil (A or B) is wired to the top two terminals on the NZS or which direction the coils are wired as the firmware will detect this and correct for reverse motor wiring.

First Power Up, Calibration and LCD Menu

When the smart stepper is first powered on it will check and see if motor power has been applied. If motor power is not applied it will show a screen indicating it is waiting for motor power:



Once motor power is applied it will take a few steps to make sure the encoder is working and figure out the rotation direction of the motor. After which it will detect that the unit has not been calibrated and request that you calibrate.



To select the calibration option (menu item shown in yellow) press the center button on the unit. Once selected the motor will start turning through a full rotation measuring the encoder at each step. Note for the motor calibration it is recommend that the motor be disconnected from the machine such that there is no load on motor shaft. Removing belt is usually good enough.

After calibration the rest of the menu system will appear, with the second item being "Test Cal" for testing the calibration.



It is recommended that you run the test calibration, to do this press the left button to scroll menu up such that the "Test Cal" is shown in yellow on top line.



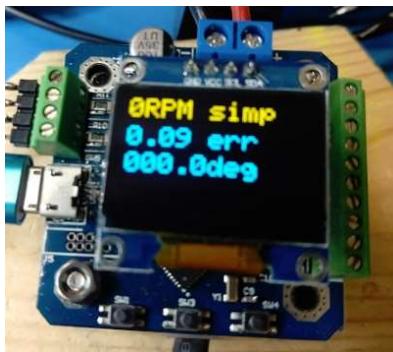
You can select this menu item by pressing the center button. Once selected the motor will again rotate a full rotation and test the calibration. After the test has ran the LCD will show the calibration error.



Typical calibration errors are around 0.1 degree or better. If your calibration is worse than this make sure your board is tightened down to motor on all four screws and run the calibration and test again. If the error is still high then you might want to check and make sure magnet is glued to the shaft properly.

To exit the calibration error screen press the center button to return to the menu.

The LCD menu can be exited by pressing the right button, at which point the LCD will show the RPM PID mode, position and error on the screen.



You can return to the menu by pressing the right button.

The other menu items include the items below

Motor mA

This menu allows you to set the maximum peak motor current in millamps

Hold mA

For the Simple PID mode of operation this menu item allows you to select the hold current for the motor. Please see the section on the operational modes.

Microstep

This option allows you to select the number of microsteps the controller should use. Note with a 1.8 degree motor (200 steps per rotation) and a calibration error of around 0.1 degrees going beyond 16x microstepping will not improve your accuracy.

EnablePin

This menu item lets you select what logic level on the enable pin to use for disabling and enabling motor.

Dir Pin

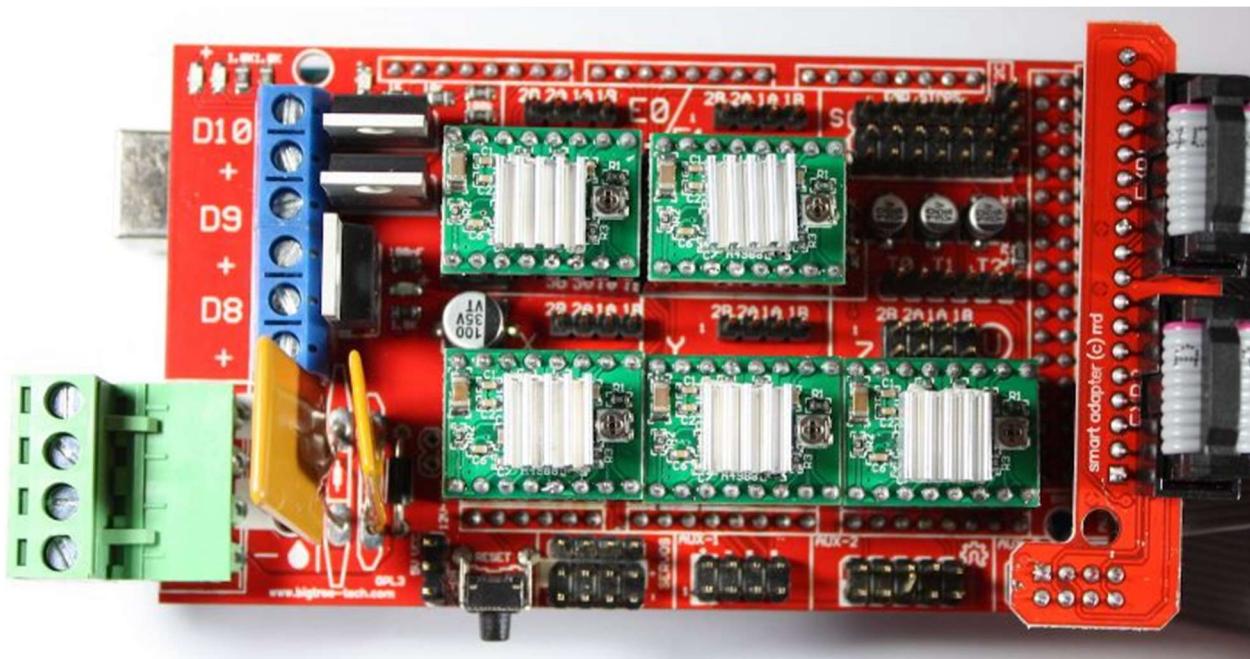
This menu item lets you select what logic level to use for clockwise and counter clockwise rotation.

Note that the command line interface through the USB allows for far more advanced configuration options.

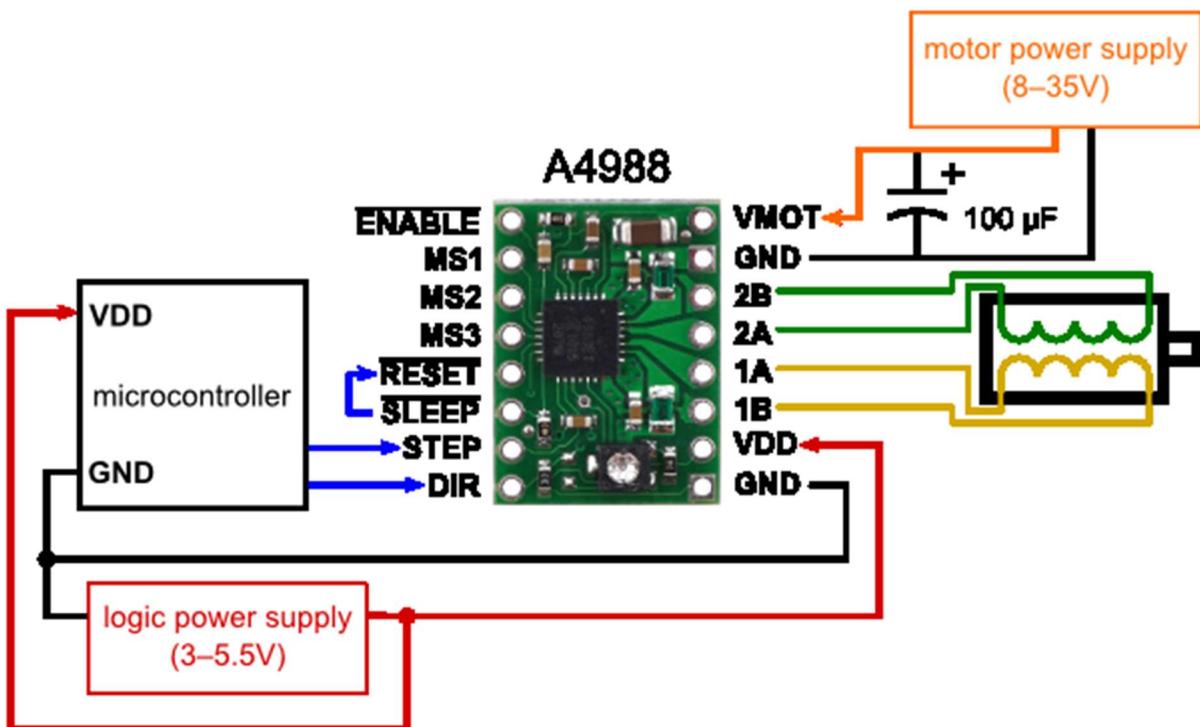
[Connecting to Motion Controller](#)

[Ramps Controller](#)

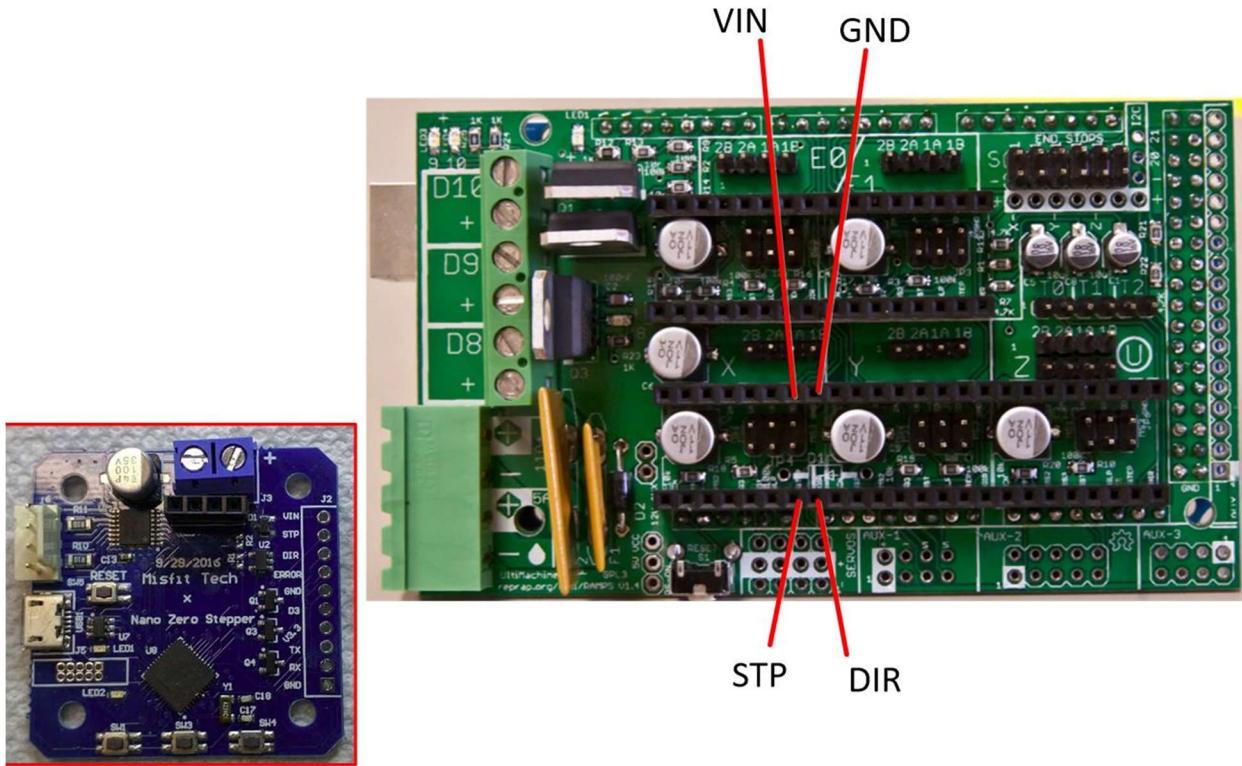
The next step is to wire the NZS to your controller, it will be assumed that your controller uses Step Sticks (green modules shown below) like the Ramps controller one shown below.



With the NZS you will no longer need the step stick for your motor as the NZS has built in driver. On the axis you are putting the NZS on remove the step stick from the controller and then wire the power, ground, step and direction to the NZS. The wiring for the step sticks is shown below.



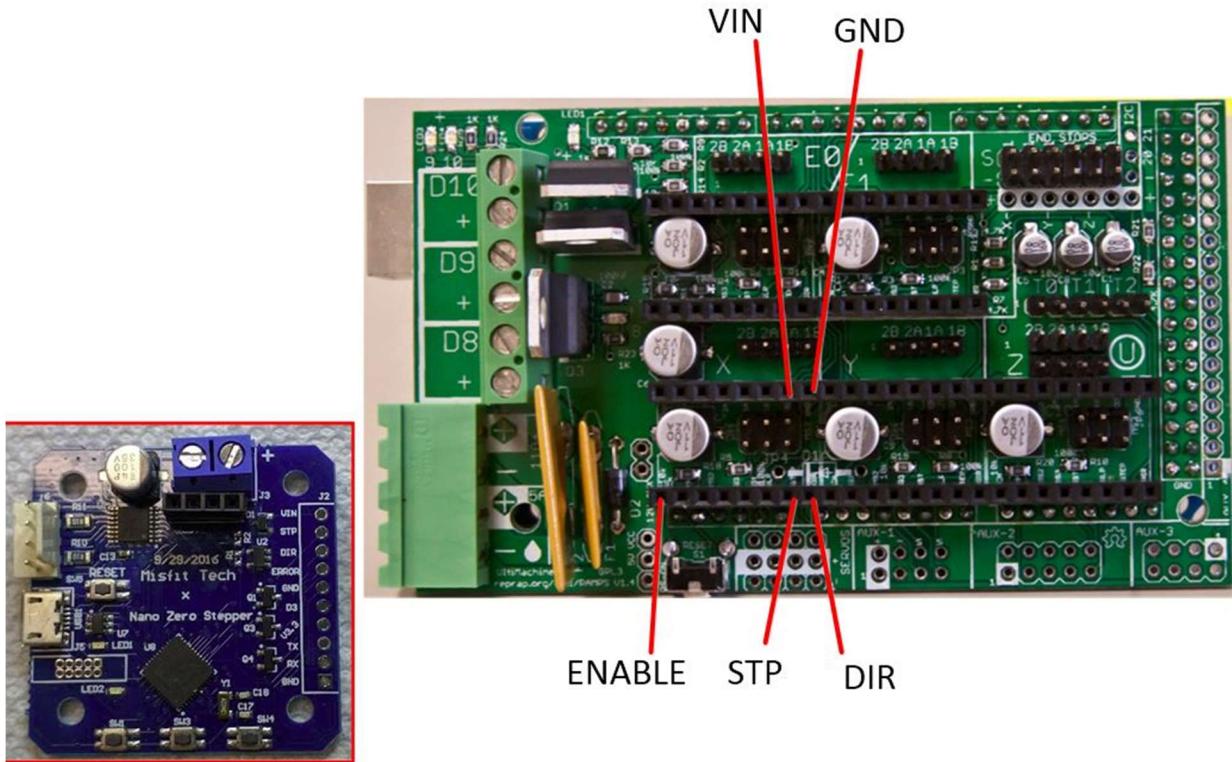
To replace the step stick with the NZS you would wire the NZS pins to pins on the RAMPs board as shown below.



The VIN, GND, STP, and DIR should be connected to the pins on the NZS with the same names. The above diagram shows the connection for the X axis on the RAMPS only, the other axis have similar wiring.

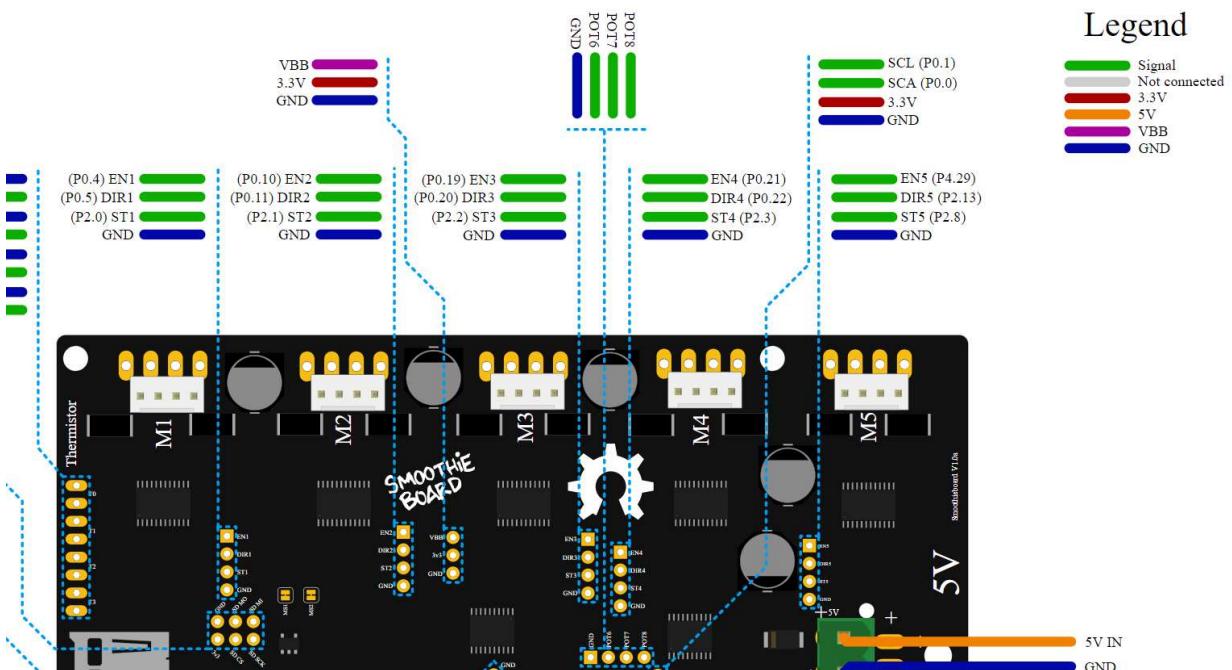
The RAMPS boards have 5V logic levels, hence the VIN pin should be connected to the 5V VIN pin on the RAMPS board. Other boards like Smoothie which uses 3.3V logic will have slightly different wiring.

Note if you have the firmware set up for the ERROR pin on the NZS to be an ENABLE then you would include the enable wire as shown below, again the ENABLE would be connected to the ERROR pin on the NZS. If you are not sure which way your firmware is configured, do not connect the enable pin.



Smoothie Board

The smoothie board pin out is documented here: <http://smoothieware.org/pinout>



The process is the same as the ramps board, basically each smart stepper needs the following connections:

1. Vin – connect to 3.3V as shown as red lines in smoothie figure above.
2. GND – Connect to the GND as shown as blue lines in smoothie figure above.
3. Step – connect to respected motor step pin (ie ST1, ST2, ST3, etc)
4. Dir - connect to the respected direction pin (ie, DIR1, DIR2, DIR3, etc)
5. Enable – optionally connect to the respect enable pin (ie EN1, EN2, EN3, etc)

This process is very much similar to connecting external stepper driver, see

<http://smoothieware.org/general-appendices>

Make sure your smart stepper microstepping and the smoothie have matching microstep rates.

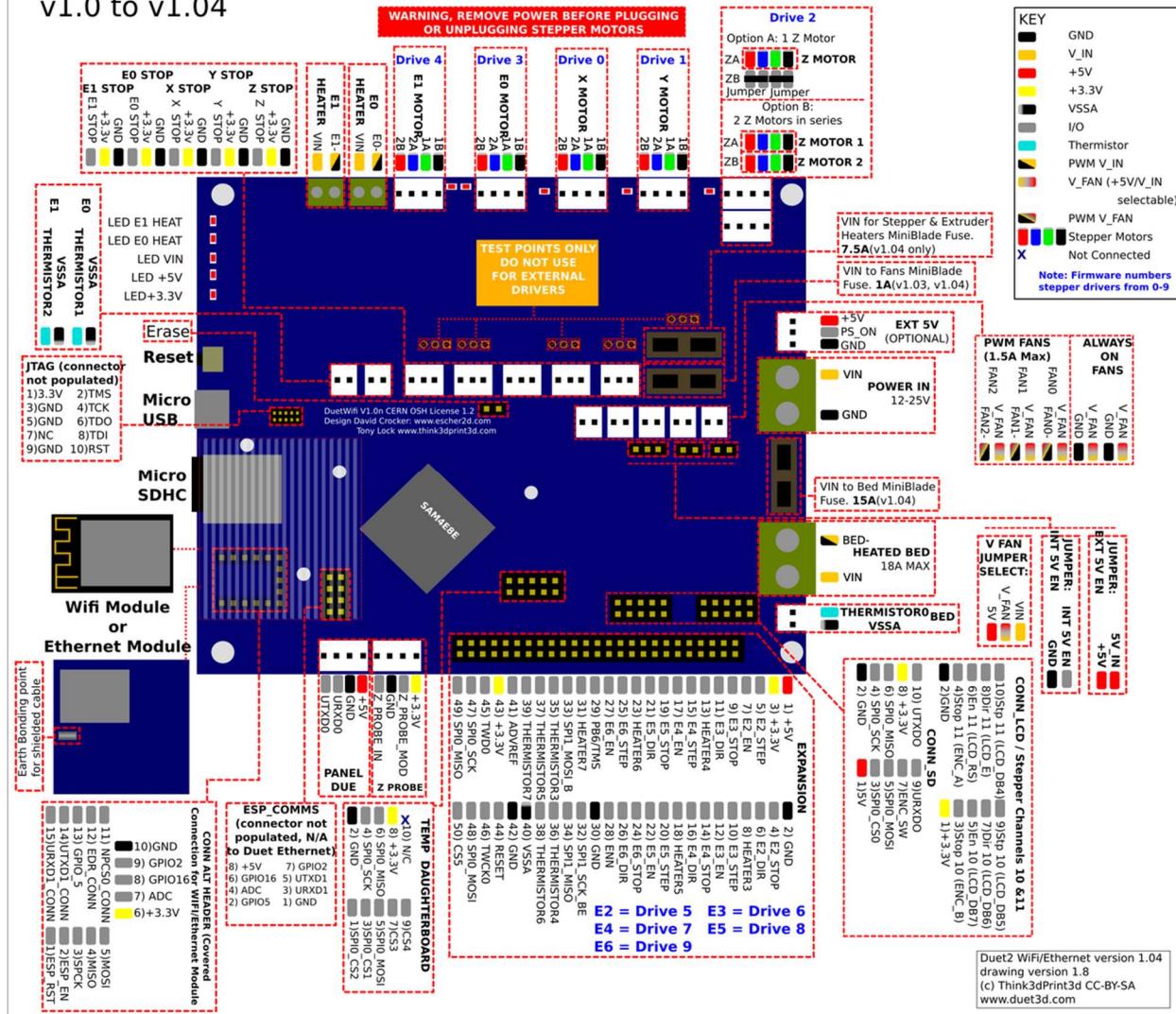
Duet

Some of the duets have the Trinamic drivers, the documentation for the Duet with Trinamic indicates that external stepper drivers should not be used. We have found that the Duet with Trinamic do cause extra pulse on the enable line, but we were able to get the duet working by not connecting the enable line. As we understand it Duet is working on a fix where the firmware will work with the Smart Steppers.

The pin out for the duet is shown below:

Duet Wifi / Duet Ethernet Connections

v1.0 to v1.04



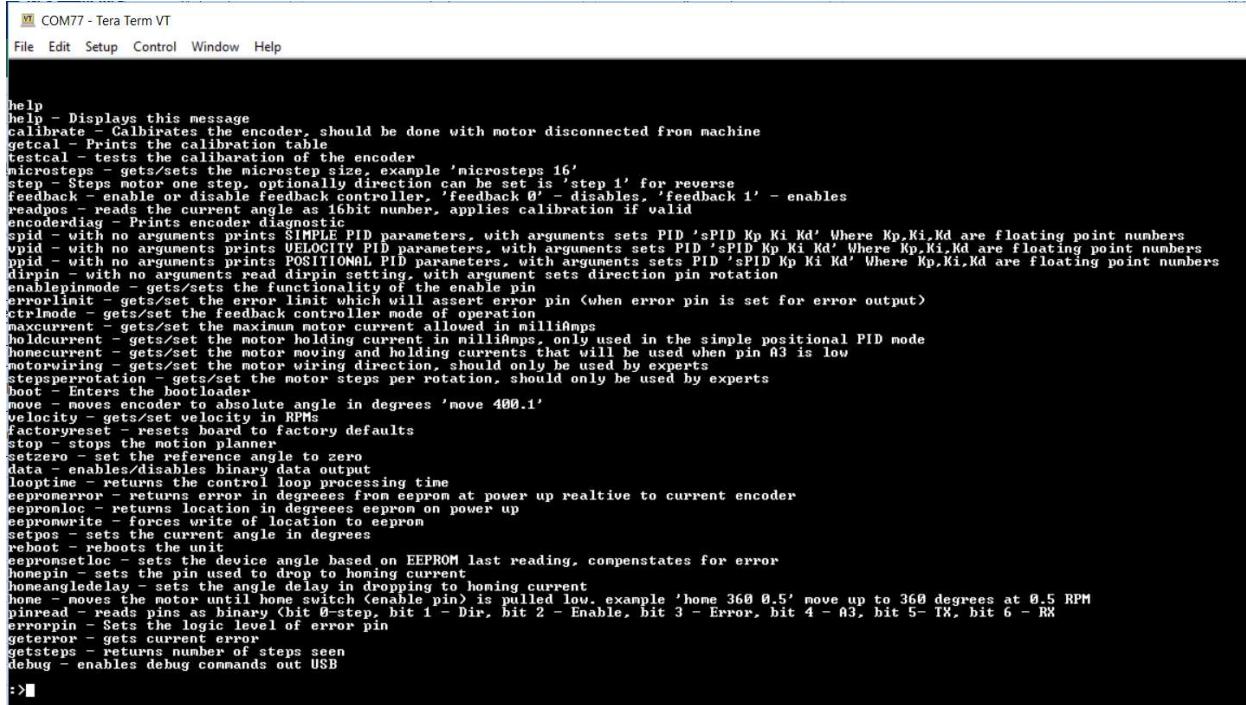
The pins that are labeled “Test Points Only do not use for external drivers” actually are the step, direction and enable pins. However here again we have found that the Duet is toggling the enable pin which may be why they do not recommend for external steppers drivers.

The connection is very similar to the Ramps and Smoothie:

1. Vin – connect to a 3.3V pin on the duet
2. GND – Connect to a ground pin on the duet
3. Step – connect to respected motor step pin
4. Dir - connect to the respected direction pin
5. Enable – Leave floating until Duet has firmware with external stepper driver support

Command Line Interface

The command line interface provides a DOS/BASH like command line interface to the unit through the USB port. The USB port will enumerate as a USB serial device such that standard terminal interfaces can be used, like TeraTerm or Putty.



```
COM77 - Tera Term VT
File Edit Setup Control Window Help

help - Displays this message
calibrate - Calibrates the encoder, should be done with motor disconnected from machine
getcal - Prints the calibration table
testcal - tests the calibration of the encoder
microsteps - gets/sets the microstep size, example 'microsteps 16'
step - Steps motor one step, optionally direction can be set is 'step 1' for reverse
feedback - enable or disable feedback controller, 'feedback 0' - disables, 'feedback 1' - enables
readpos - reads the current angle as 16bit number, applies calibration if valid
encoderdiag - Prints encoder diagnostic
epid - with no arguments prints SIMPLE PID parameters, with arguments sets PID 'sPID Kp Ki Kd' Where Kp,Ki,Kd are floating point numbers
vpid - with no arguments prints VELOCITY PID parameters, with arguments sets PID 'sPID Kp Ki Kd' Where Kp,Ki,Kd are floating point numbers
ppid - with no arguments prints POSITIONAL PID parameters, with arguments sets PID 'sPID Kp Ki Kd' Where Kp,Ki,Kd are floating point numbers
allowinhibit - gets/sets the inhibit pin, sets direction pin rotation
enablepinmode - gets/sets the functionality of the enable pin
errorlimit - gets/set the error limit which will assert error pin (when error pin is set for error output)
ctrlmode - gets/set the feedback controller mode of operation
maxcurrent - gets/set the maximum motor current allowed in milliAmps
holdcurrent - gets/set the motor holding current in milliAmps, only used in the simple positional PID mode
homcurrent - gets/set the motor moving and holding currents that will be used when pin A3 is low
motorwiring - gets/set the motor wiring direction, should only be used by experts
stepsperrotation - gets/set the motor steps per rotation, should only be used by experts
boot - Enters the bootloader
move - moves encoder to absolute angle in degrees 'move 400.1'
velocity - gets/set velocity in RPMs
factoryreset - resets board to factory defaults
stop - stops the motor
setzero - set the reference angle to zero
data - enables/disables binary data output
looptime - returns the control loop processing time
eepromerror - returns error in degrees from eeprom at power up relative to current encoder
eepromloc - returns location in degrees eeprom on power up
eepromwrite - forces write of location to eeprom
setpos - sets the current angle in degrees
reboot - reboots the unit
eepromsetloc - sets the device angle based on EEPROM last reading, compensates for error
homenpin - sets the pin used to drop to homing current
homeangledelay - sets the angle delay in dropping to homing current
home - moves the motor until home switch (enable pin) is pulled low, example 'home 360 0.5' move up to 360 degrees at 0.5 RPM
pinread - reads the pin state (bit 0-step, bit 1 = Dir, bit 2 = Enable, bit 3 = Error, bit 4 = A3, bit 5 = TX, bit 6 = RX)
eeprompin - Sets the logic level of error pin
geterror - gets current error
getsteps - gets number of steps seen
debug - enables debug commands out USB
:>■
```

The command line interface provides several command and options as shown in the snapshot above. Since the command line interface is constantly growing as features are added to the firmware it is always a good idea to type “help” to get the latest list of commands and a brief description of the commands.

help

The help command will return a list of commands that the smart stepper supports.

getcal

This command will print out the 200 point calibration table. This is useful if you are doing firmware development and do not want to calibrate each time you update firmware. You can take this table and copy it into the nonvolatile.cpp file as shown below

calibrate

This will run a 200 point calibration of the encoder.

testcal

This will test the calibration and report the maximum error in degrees.

step

This will move the motor one step clockwise, the step size is based on the current microstep setting. To move the motor counterclockwise use “step 1”. To move the motor clockwise 16 steps use “step 0 16” to move motor counterclockwise 16 steps use “step 1 16”

feedback

This commands disable/enables feedback control loop. The plan is to discountinue this command in the future and use the “controlmode” command to put controller in open or one of the many closed loop operational modes.

readpos

Reads the current motor position and reports it as degrees.

encoderdiag

This command will read and report the AS5047D internal registers for diagnostic purposes.

microsteps

This command gets/sets the number of microsteps the smart stepper will use for the step command and the step pin. The number of microsteps does not affect the resolution of the controller but rather how fine you can set the position.

spid

This command sets the Kp, Ki, and Kd terms for the simple positional PID controller.

ppid

This command sets the Kp, Ki, and Kd terms for the positional PID controller.

vpid

This command sets the Kp, Ki, and Kd terms for the velocity PID controller.

velocity

This sets the velocity to rotate motor when unit is configured for velocity PID mode of operation.

boot

This command will put the microprocessor in the boot loader mode. Alternatively this can be done by double pressing the reset button.

factoryreset

This erases the calibration and other system and motor parameters such that unit is reset to the factory ship state. After this command the unit will need to be calibrated to the motor again.

dirpin

This command sets which direction the motor will rotate when direction pin is pulled high. The direction pin is only sampled when the step pin has a rising edge. ‘dirpin 0’ will set the motor to rotate clockwise when dir pin is high ‘dirpin 1’ will set the motor to rotate counter-clockwise when dir pin is high

errorlimit

Gets set the maximum number of degrees of error that is acceptable, any positioning error about the error limit will assert the error pin, when error pin is set as error output. For example: :>errorlimit 1.8 Will set the error limit to 1.8 degrees.

ctrlmode

Gets/Sets the feedback controller mode of operation. The command takes an integer from 0 through 4 to set the control mode per table below: Controller off - 0 -- this is not currently used Open-Loop - 1 -- this is open loop with no feedback Simple PID - 2 -- simple positional PID, which is factory default Positional PID - 3 -- current based PID mode, requires tuning for your machine Velocity PID - 4 -- velocity based PID, requires tuning for your machine and speed range

If you are unsure what you are doing leave unit in the Simple PID mode of operation.

maxcurrent

This sets the maximum current that will be pushed into the motor. To set the current for maximum of 2.0A you would use command “maxcurrent 2000” as the argument is in milliAmps.

holdcurrent

For the Simple Positional PID mode the minimal current (ie current with no positional error) is the hold current. You set this current based on the required holding torque you need for your application. The higher the hold current, the hotter and noisier the motor will be but also the larger the holding torque. For the Positional PID mode the PID tuning params have to be set correctly such that the control loop will dynamically determine the holding torque. This tuning of the PID can be difficult, hence the simple PID mode will work most of the time out of the box by setting maximum current and holding current.

motorwiring

The firmware always uses a positive angle as a clockwise rotation. A stepper motor however could have wiring done with one coil reversed wired, which will cause motor to normally operate in opposite direction. The Smart Stepper firmware will detect the motor wiring direction, using the encoder, and the firmware will compensate for a reverse wired motor. The reverse or forward wiring of a motor is detected on first power up after factory reset. If the wiring changes after that you can compensate using this command. **HOWEVER** it is better to do a factory reset and recalibrate motor if wiring changes.

stepsperrotation

The Smart Stepper firmware will with first power on after factory reset detect the number of full steps per rotation for the stepper motor and store in flash memory. This command will read this parameter from flash and allow user to change this parameter if motor is changed. **HOWEVER** it is better to do a factory reset and recalibrate motor if motor changes.

move

The move command will request the motor to move to an absolute angle position. Optionally the user can specify the rotational speed (RPMs) by which the move should happen. For example, if the current motor position is at angle 0 and you issue ‘move 3600’ the motor will turn 10 rotations clockwise to angle of 3600 degrees. If issue the ‘move 3600’ again nothing will happen as motor is already at angle 3600. If motor is at angle 0 and user issues the command ‘move 3600 20’ then motor will move to 10 rotations clockwise to angle of 3600 at a rate of 20 RPMs.

stop

If user issues a move command that takes a long time and wants to stop the move before completion, then user can issue the stop command command which will stop a move operation.

setzero

This command will take the current motor position and set it to absolute angle of zero. Note that if you are in the middle move it will take the position at the time of the command and use it, thus it is recommended a move be stopped or wait for completion before issuing the setzero.

Hardware Revisions and Errata

NEMA 17 Boards

3-21-2017 boards

This board may have errors on the silkscreen for the Error, Enable and A3 pins specifically the pin assignment from top to bottom are:

1. Vin
2. Step
3. Direction
4. Enable
5. Error
6. A3
7. V3.3
8. TX
9. RX
10. GND

1-20-2019 boards

The 1-20-2019 boards added additional capacitors to the motor voltage to help reduce noise on wires/cables. Additionally, the pull up resistors for the level shifters were changed to 2.7k ohm to help reduce effects of the capacitance with long runs of cabling and noise on the wires.

Firmware Revisions

The firmware revision notes are contained in the board.h file. A major change in the firmware happened in February 2019. This included changing the USB vendor ID to a MisfitTech owned vendor ID

and changing the bootloader to include a mass storage option for easier firmware updates. Note the new vendor ID will require new driver install.

[Future Changes](#)

There is a plan to migrate away from the Arduino IDE for the project as there are numerous bugs in the Arduino libraries as well as design issues with the libraries that limit the project.

[License](#)

The smart stepper related hardware is released under the Creative Commons Attribution Share-Alike 4.0 License as much of the work is based on Mechaduino project by J. Church.

<https://github.com/jcchurch13/Mechaduino-Firmware>

The firmware is licensed as GPL V3 for non-commercial use. If you want to release a closed source version of this product, please contact MisfitTech.net for licensing details.