UPDATED MOTOR CONTROLLER

SP6 Addendum

Instructions for an updated stepper motor drive circuit
that is quieter and using easy to find components

Steve Peterson
05-Jan-2023

Contents

Tables

Figures

Revision History

05-Jan-2023	Original release

Description

I designed the “silent” stepper motor driven desk clock in 2021 using an Arduino Nano and a custom driver circuit that made the clock reasonably quiet. I could not find any off-the-shelf stepper motor drivers that worked better than a custom designed circuit. The clock was quiet, but the gears would start to rattle over time. I wanted to find a better way to drive the clock.

Here are some of the things to improve on the original design:

1) Make the clock run quieter.
2) Make the clock more accurate.
3) Avoid using a custom circuit board. The custom circuit board sold for a few dollars, but international shipping and customs charges could be 3-4X more than the cost of the board.
4) Use a standard stepper motor. The original design uses a 34 ohm stepper motor that can be difficult to find. The new design should use the more common 2-3 ohm stepper motors.
5) Simplify the calibration routine.

This addendum describes a new and improved stepper motor drive circuit that improves all these criteria.

Early Experiments

Some of my early experiments used an A4988 stepper motor driver to drive NEMA17 stepper motors. The gears clicked as the motors jumped to each step even with microstepping enabled. It also required a separate power supply of at least 8V. This was not looking like a good solution.

The next experiment used cheap and easy to find 28BYJ-48 stepper motors. They were ruled out for several reasons. The internal gearing uses really tiny gears that started ticking within a day or two of unloaded testing. Maybe it was a bad motor, but the longevity seems questionable. They also have unusual gear ratios listed as approximately 63.68395:1. The very first motor I opened up had a gear ratio of exactly 60.000:1. A consistent gear ratio could be accounted for, but variations between different manufacturers makes them difficult to use in a clock. One clock would be accurate and the next would lose several minutes per day. It would be very frustrating for many builders. Also, the default configuration is 5 wire mode which makes microstepping difficult. I think I am completely done with considering these stepper motors for use in clocks.

A Possible Solution

A breakthrough came when I discovered a pre-made board called "CNC Shield V4". It is designed to power a 3 motor engraving machine using an Arduino Nano and 3 A4988 stepper driver modules. The A4988 driver had already been ruled out, so I used a newer Trinamic TMC2208 "silent" driver module. A DS3231 real time clock was added to set the reference time with an accuracy of about a minute or two per year. This looks like a workable solution.

Here are the parts used in the new driver circuit:

1) CNC Shield V4 – Amazon US$10.96 for 3 boards.
2) TMC2208 drivers – Amazon US$21.99 for 6 drivers.
3) Arduino Nano – Amazon US$16.99 for 3 modules.
4) DS3231 Real Time Clock – Amazon US$12.11 for 4 modules.
5) NEMA17 stepper motor – Amazon US$9.99 each.

Notes regarding each item:

1) The CNC Shield V4 has a well-known bug that needs to be fixed. The jumpers under the motor drivers are connected to Gnd instead of Vcc, so it needs a modification to support anything except full step resolution. I hard-wired the 3 jumpers to Vcc for 16X micro-stepping. A modified design (black and yellow) without this bug can be purchased from KeyeStudio on AliExpress for US$8.86 including shipping, but the fix is super easy so I went with the cheaper red version on Amazon.
2) The TMC2208 stepper motor driver is amazing. It supposedly supports up to 256 position microstepping, but I found that 16 positions is good enough to drive the clock and easier to configure. A great thing about the TMC2208 drivers is that it can bypass the internal regulator and run directly on 5V. Vref is set for the lowest possible motor current to minimize the risk of overloading the USB supply.
3) The CNC Shield V4 is designed to use an Arduino Nano that is now available with either an old style mini USB connector or a USB-C connectors. The updated clock base was modified to support either connector style, so select either style. Arduino Nanos with pre-installed header pins are also supported.
4) The DS3231 real time clock used is often listed as "DS3231 for Raspberry Pi". It needs to be the one with a 5 pin female socket and yellow battery (or capacitor?). It has voltage and temperature compensation to maintain an accuracy of about a minute or two per year. A slight wiring modifications allows the 5 pin socket to plug into one of the 4 pin stepper motor headers.
5) The TMC2208 drivers used in this design support the common 1.5A stepper motors used in many 3D printers. This makes it easy to find stepper motors to use for the clock. The drive current should be adjusted to be as low as possible while still powering the clock. I used StepperOnline part number 17HS15-1504S which is rated at 1.5A and 2.0 ohm coils. A short body 17HE08-1004S (1.0A and 3.6 ohms) is also functional with a slight modification to the stepper motor cable. The 34 ohm stepper motors used in the original design can also be used, but the lower ohm stepper motors are slightly preferred. Almost any NEMA17 bipolar stepper motor is likely to work with the TMC2208. The base has been designed to work best using motors with a body length of either 33mm or 39mm.

Wiring

A few wiring modifications were made to the CNC Shield V4 plus a tiny modification to the real time clock. These changes are smaller than the effort of using a custom circuit board. The CNC Shield V4 is significantly cheaper than anything I could produce in low volume, so we need to make a few wiring edits to make it work for our application.

The cheapest red colored CNC Shield V4 boards on Amazon have a bug that needs to be fixed to enable micro-stepping. KeyeStudio sells a black colored board without the bug. It is harder to find and more expensive, so I will describe how to modify the red board.

Follow the pictures on the next few pages.

1) Connect the micro-stepping jumpers to Vcc to support 16X micro-stepping. Start by removing the 3 jumper blocks from the top side of the board. Solder a jumper wire on the back of the board from the 3 jumper pins to the nearest Vcc connection. The TMC2208 module labels the pin as VIO. We always want 16X micro-stepping, so the three jumpers will be hard wired to Vcc. We only need to fix one driver module to use it in the clock. **The top side jumpers under the TMC2208 socket must be removed to prevent shorting Vcc to Gnd.** The black KeyeStudio board could skip this step, but would still need to do all of the remaining wiring edits.
2) Connect the TMC2208 Vmot supply to Vcc to run the stepper motors using 5V from the USB cable. The easiest fix is to extend the wire from the step #1 to the capacitor on the left. Running the clock on 5V is easier and lower power compared to using an 8-12V DC power supply.
3) Connect the 4 wires from the SDA/SCL serial port to an unused stepper motor header. The unused motor header pins are floating when there is no motor driver plugged in, so one header is used by the real time clock. We will be using the header that is straight down from the serial port header. The SDA and SCL connections are about 1” long. Vcc and Gnd have shorter jumpers to closer locations. Follow the pictures.
4) Short the 5th position Gnd pin of the DS3231 RTC to the unused 4th position NC pin. This allows the 5 pin RTC to plug into the 4 pin motor header. This edit is done on the top side to the DS3231 real time clock module. It is shown a few pages ahead.

This is the back side of the CNC Shield V4 board with the relevant power and ground pins labeled. It is shown as a reference.

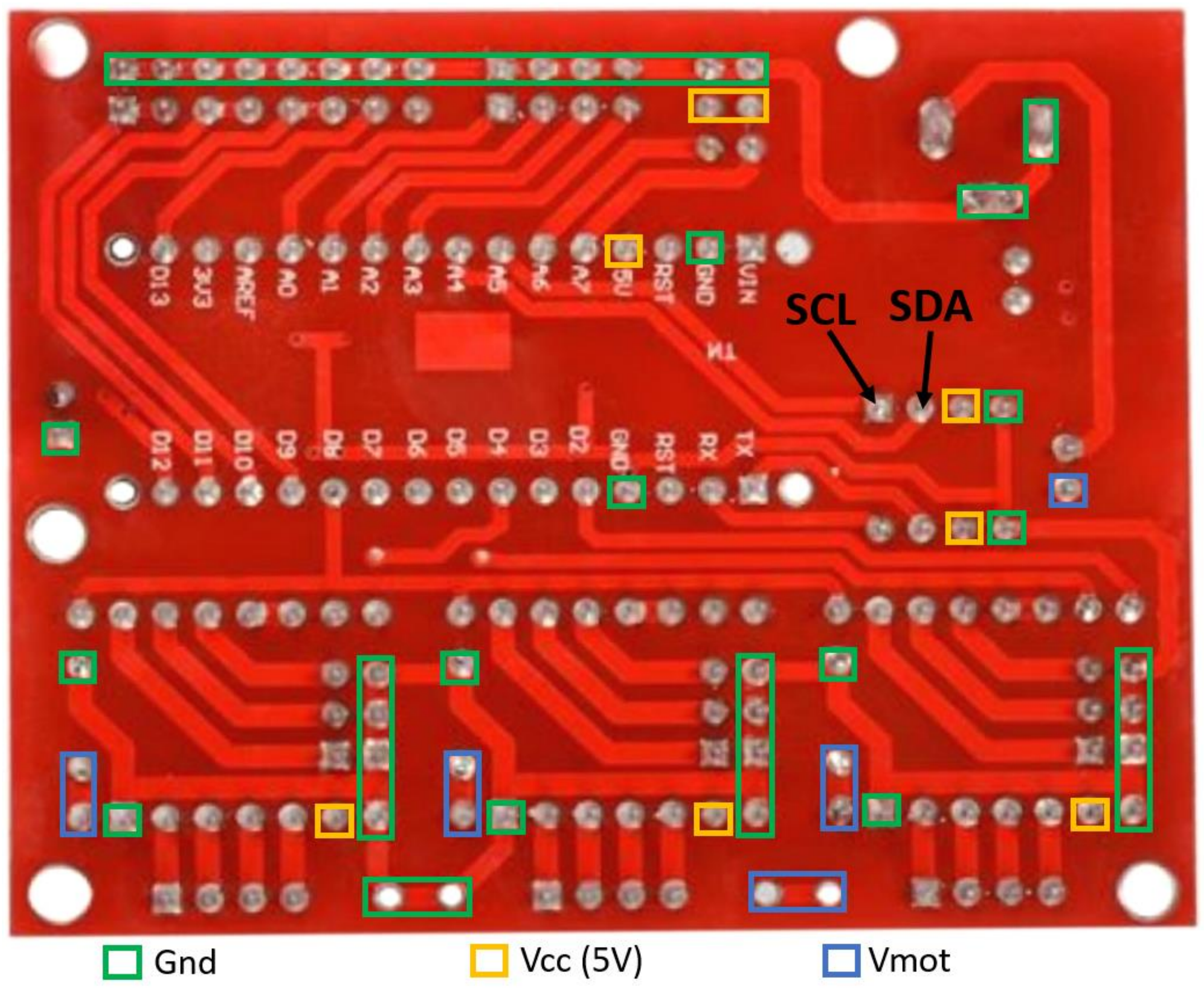

Figure 1: CNC Shield V4 Power Connections

The modifications to make the board work as a clock driver are shown below. Edit #1 connects the 3 jumper pins to Vcc. Edit #2 shorts Vmot to Vcc, bypassing the TMC2208 voltage regulator and allowing everything to run on 5V.

Edit #3 connects four wires from the serial port pins down to the unused stepper motor header. The Vcc and Gnd connections can be made to the closest available locations.

Edit #4 is made on the top side of the board and shown on the next page.

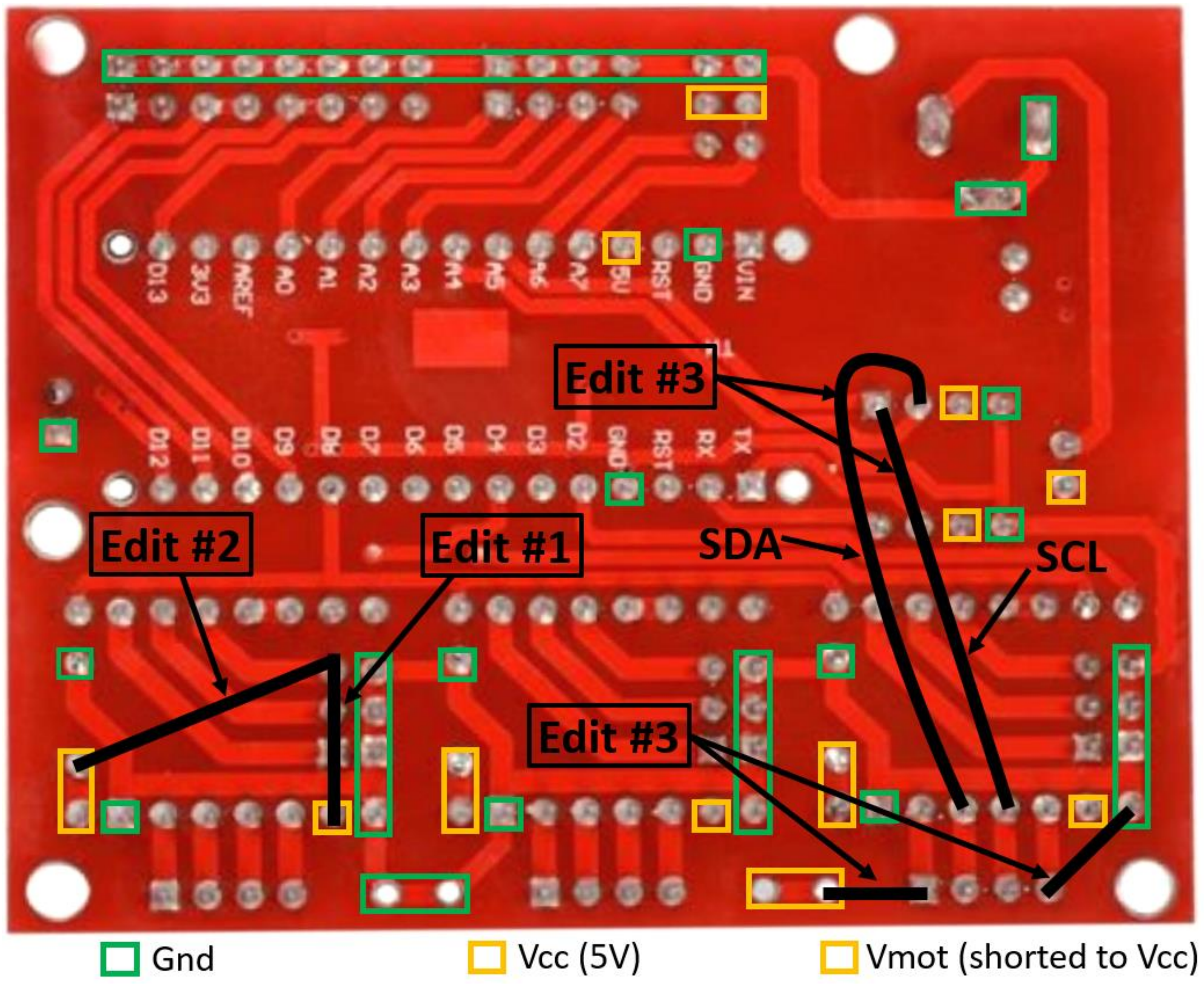

Figure 2: CNC Shield V4 Wiring Modifications

The top side of the board has minimal modifications.

1) Solder the headers onto the Arduino Nano and insert it in the orientation shown.
2) Remove the jumpers from under the lower right motor driver. This is important since the back side of the board has these pins hard wired to Vcc. Leaving the jumpers in would short 5V to Gnd.
3) Insert the TMC2208 into the lower right driver socket. Turn the Vref potentiometer to the left to lower the current as much as possible. The TMC2208 heat sink might not be needed with the current set so low. The base was modified to allow room for it, so add it to the TMC2208.
4) Insert the DS3231 real time clock into the lower left stepper motor header. It should be shifted to use only the pins labeled NC, C, D, and "+". The "-" (Gnd) pin needs a small wire to connect it to the NC position. This allows the 5 pin RTC to plug into the 4 pin header.
5) The 8 pin header near the top of the board is used to change motor direction and select a few different speeds. They are visible in the picture on the next page. Adding a jumper in the left position will change the motor direction. The three header positions on the right are used to select one of 8 pre-programmed speeds.

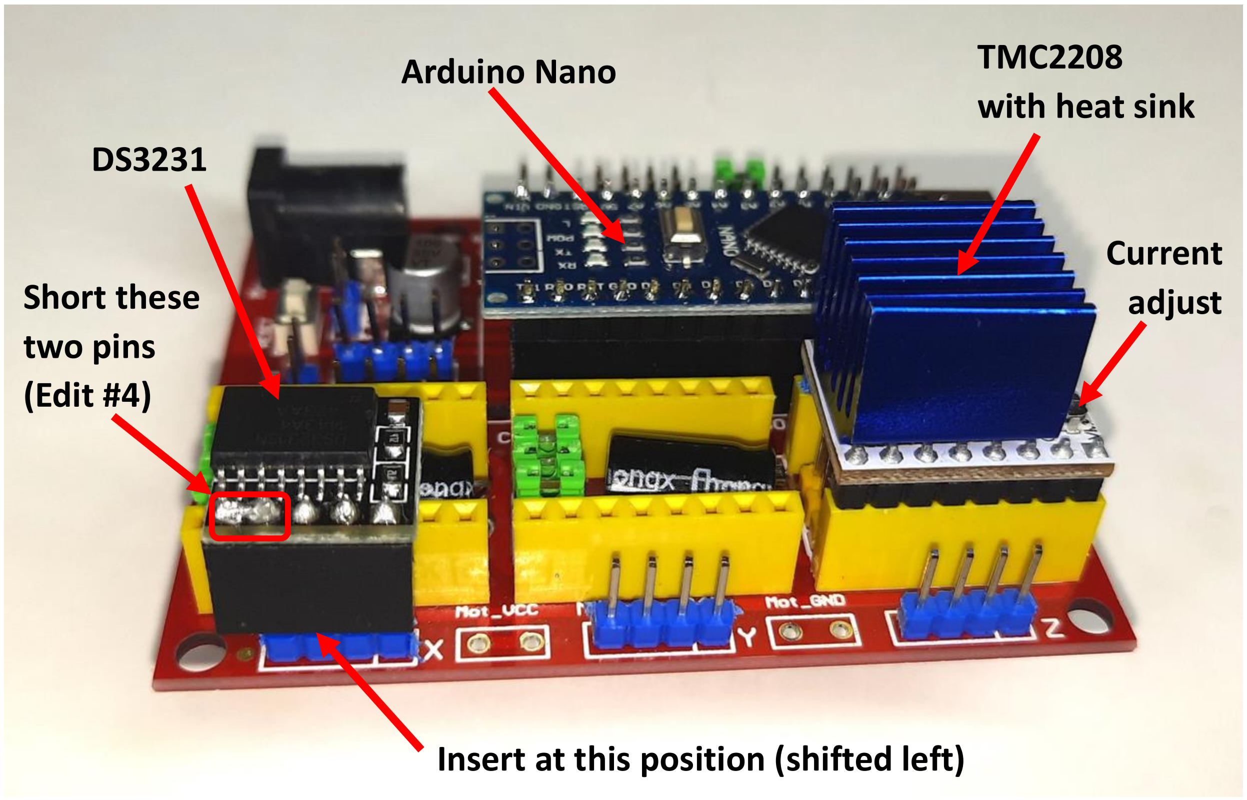

Figure 3: Top Side Close-up of Components

This is the complete board with a stepper motor attached and ready to test. The TMC2208 heat sink might be optional when adjusted for really low currents, but I decided to make room in the base to include it.

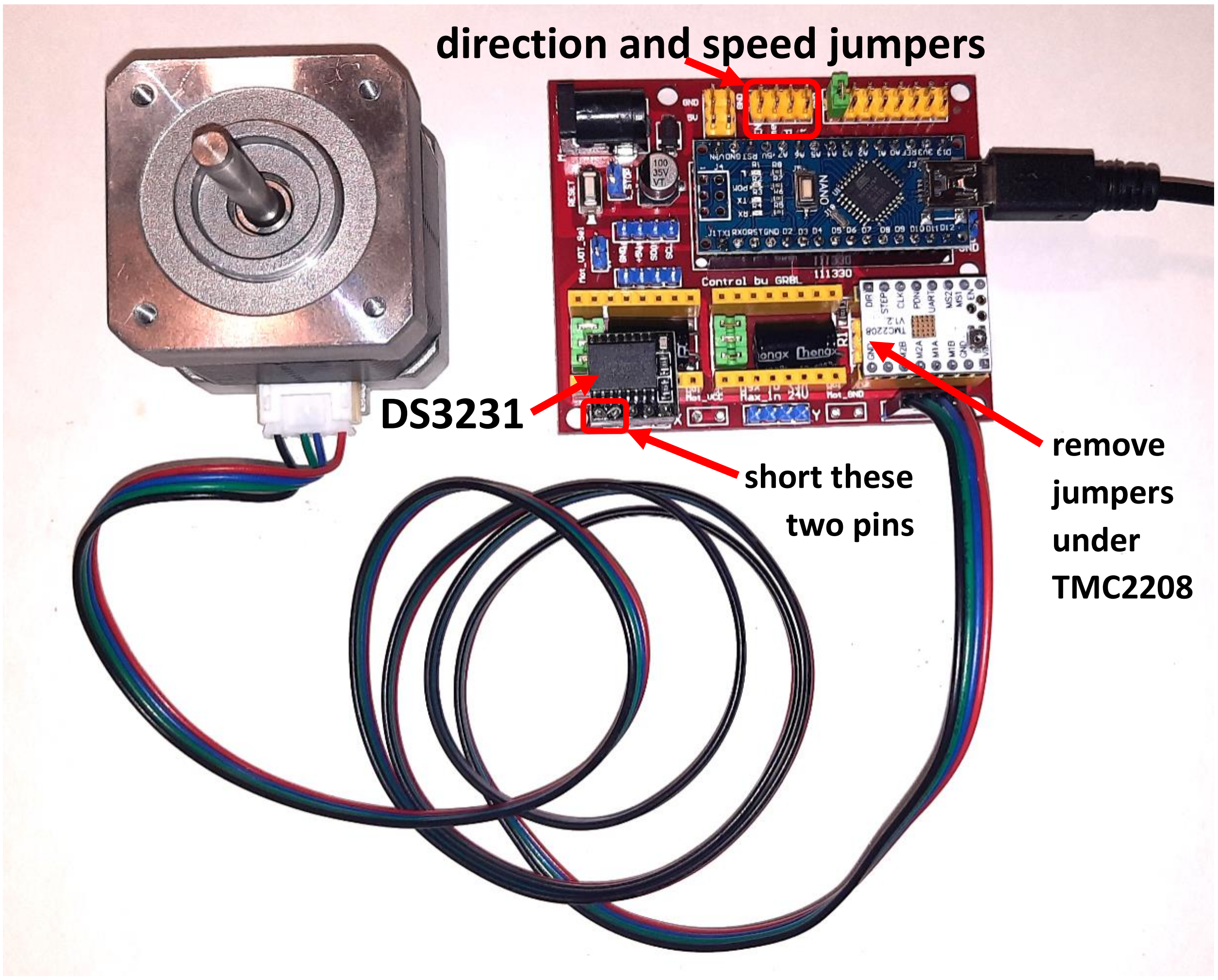

Figure 4: Assembled Driver Ready to Test

Initial Tests

The first tests of this new design look incredibly promising. The circuitry inside the TMC2208 keeps the motors running smoothly with almost no noise. They are a significant improvement over the A4988. My resistor based design has a very slight humming noise when running stand-alone and some gear noise when inserted into a clock. This new design runs much quieter than the resistor based design.

The total power might be slightly higher than my original design, since there are additional components. It is still low enough to run on a standard USB cable. The TMC2208 potentiometer should be adjusted to just enough power to run the clock. The clock runs great with TMC2208 Vref potentiometer turned to nearly the lowest possible setting.

Programming the Arduino

It is a good idea to test the circuit before assembling everything into a completed clock. This allows you to adjust the TMC2208 current levels to be as low as possible and still allow the clock to operate.

Perform all the required wiring modifications previously described and plug everything together as shown in the diagrams.

Download the CNC Shield V4 Arduino sketch from https://www.stevesclocks.com/sp6 and load it into the IDE. The SP6 silent stepper motor clock has a 54 tooth second hand gear driven by a 15 tooth pinion on the stepper motor. The speed selection jumpers should be “out”, “in”, and “in”. Other clock designs with different gear ratios would need different jumper settings. The leftmost jumper sets the motor direction and can be added or removed depending on what your clock needs.

The final speed settings might change slightly as new clocks are designed. Here are the latest values that should work for some of the clocks that are being designed:

Table 1: Speed Selection Jumpers

Setting	Jumpers	Gear Ratio	Examples	Notes
0	000	2.4:1	24:10, 36:15	
1	001	2.667:1	32:12	Small wood clock might use this
2	010	3:1	30:10, 45:15	
3	011	3.6:1	36:10, 54:15	SP6 desk clock uses this
4	100	4.5:1	36:8, 54:12	Large printed clock might use this
5	101	5.4:1	54:10	Large wood clock might use this
6	110	6:1	48:8, 60:10	
7	111	10:1	60:6, 80:8	Fast mode for debug

Using speed selection jumpers will allow the CNC Shield V4 to be moved between different clocks without having to re-program the Arduino Nano. New timing parameters are calculated any time the jumpers are moved.

Connect the USB cable to the Arduino Nano using any available USB port on your computer. Select the appropriate COM port using "Tools→Port:→COMx", where COMx might be COM2 through COM6. Compile the sketch and download it to the Arduino Nano. I have some Arduino Nano devices that use the old bootloader and some that use the new bootloader. If the sketch compiles, but gives an error during download, try switching to the other bootloader using either "Tools→Processor:→ATmega328P" or "Tools→Processor:→ATmega328P (old bootloader)".

I recently discovered a very useful debug feature of the Arduino IDE. Click the icon that looks like a magnifying glass in the upper right corner if the IDE window. A new window will pop up with debug statements sent from the Arduino using Serial.print commands. The first line shows the program description and revision. The next few lines show the motor delay parameters. Everything after this is a record of the algorithm in action. Each "+" indicates that the algorithm needs to speed up slightly and each "-" indicates that it is on track or needs to slow down slightly. Any duty cycle between 10% and 90% indicates proper tracking. The number in parenthesis at the end of each line is the total number of minutes elapsed since the algorithm started running. It will also show hours and days after the clock runs long enough. The algorithm uses long integers that can run for 68 years before overflowing.

The serial monitor might look like this:

```
CNC Shield V4 clock movement - Rev 1.00

speed 1
steps 142 (plus 2/9)
min_delay 3380
max_delay 3556

++-+-+-++-+-+-+-++-+-+-+-++-+-+-+-++-+-+-+-++-+-+-+-++-+-+-+ (1m)
-++-+-+-++-+-+-+-++-+-+-+-++-+-+-+-++-+-+-+-++-+-+-+-++-+-+- (2m)
+-++-+-+-+-++-+-+-++-+-+-+-++-+-+-+-++-+-+-+-++-+-+-+-++-+-+ (3m)
-+-++-+-+-+-++-+-+-+-++-+-+-++-+-+-+-++-+-+-+-++-+-+-+-++-+- (4m)
```

This example shows the values used with a speed setting of 1. The jumpers for speed 1 are "out", "out", and "in" for a binary setting of 001. This clock has 32 teeth on the main gear and 12 teeth on the pinion rotating at 2.667RPM. The motor needs 142.222 steps per second. This speed setting will use 142 full steps plus an additional 2 steps every 9 seconds. The min and max delay values are the fast and slow delay values that the motor uses to track the real time clock.

Testing the Motor

There are a few things to check if the motor does not spin or if it spins erratically.

Adjust the TMC2208 potentiometer if the motor seems to move but has almost no power. I find that the potentiometer can be turned to the lowest possible setting or close to the lowest setting. Too high of a setting seems to stall the motor when running at 5V. A clock needs very little power, so use the lowest stable setting. You should be able to easily stop the motor by holding the shaft.

If the shaft oscillates back and forth without rotating, then it might need a cable modification. The TMC2208 header expects a motor wired as 1B, 1A, 2A, and 2B. Pins 1A and 1B connect to one motor coil. Pins 2A and 2B connect to the other coil. Some stepper motors plug straight in. Other stepper motors are connected as 1B, 2A, 1A, and 2B. These motors need the middle two wires on the 4 pin connector to be reversed.

You can also use an ohm meter to check if the stepper motor needs the wiring change. The original cable will work if you can measure resistance between positions 1-3 and 4-6 at the stepper motor connector. You will need the modification if you measure resistance between 1-4 and 3-6.

Gently lift the tab holding the wires in place and remove the middle two wires. Swap their positions and insert them back into the 4 pin connector.

Here is the original unmodified stepper motor cable

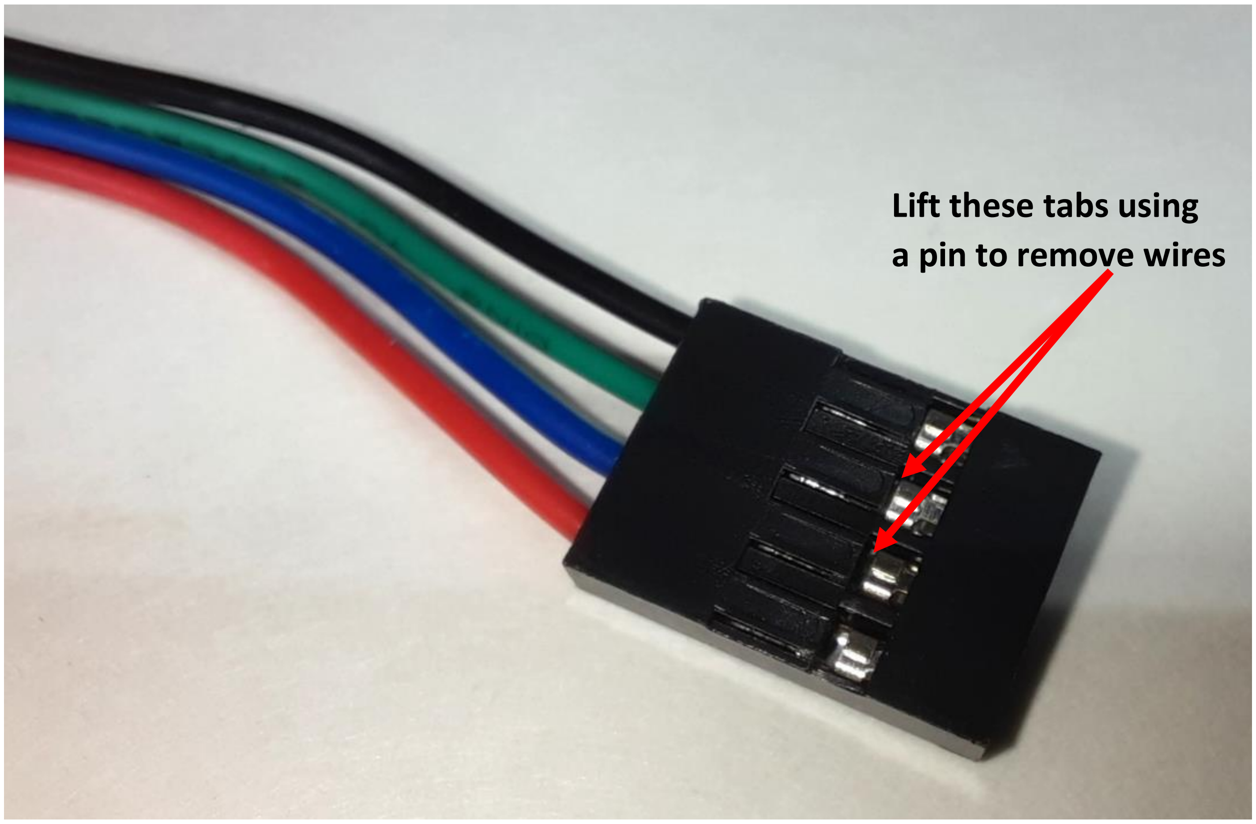

Figure 5: Standard Stepper Motor Cable

This is the modified cable with the middle two pins swapped.

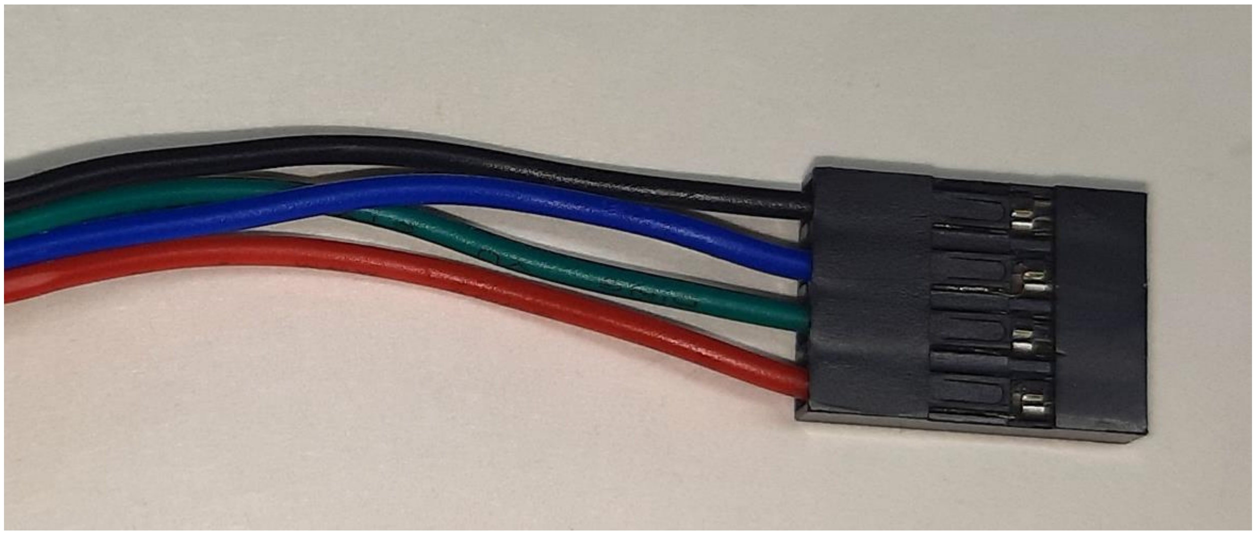

Figure 6: Modified Stepper Motor Wiring

The modified cable seems to be needed for about 30% of the small sample of motors I tested. The modification is easy to make, so don’t worry about which type of stepper motor to order. Almost any bipolar NEMA17 stepper motor should be able to work using this new driver. The best motors are the ones with coil resistances of 2-3 ohms. The 34 ohm 0.4A motors used in the original design also work, but they have much more torque than needed even when Vref is set to the lowest setting. The motor holder is designed to directly support motors with a body length of 33mm or 39mm.

Algorithm

The clock algorithm is fairly simple. It uses two delay values. The minimum delay is about 4% faster than required and the maximum delay is about 1% slow. The algorithm switches between fast and slow delays to track the real time clock. The second hand will still appear to be moving at a uniform speed and likely never deviates more than a few milliseconds from the ideal position.

The primary part of the algorithm allowing the motor to track the real time clock is only about 20 lines of code. It can be downloaded from https://www.stevesclocks.com/sp6

The flowchart is shown below:

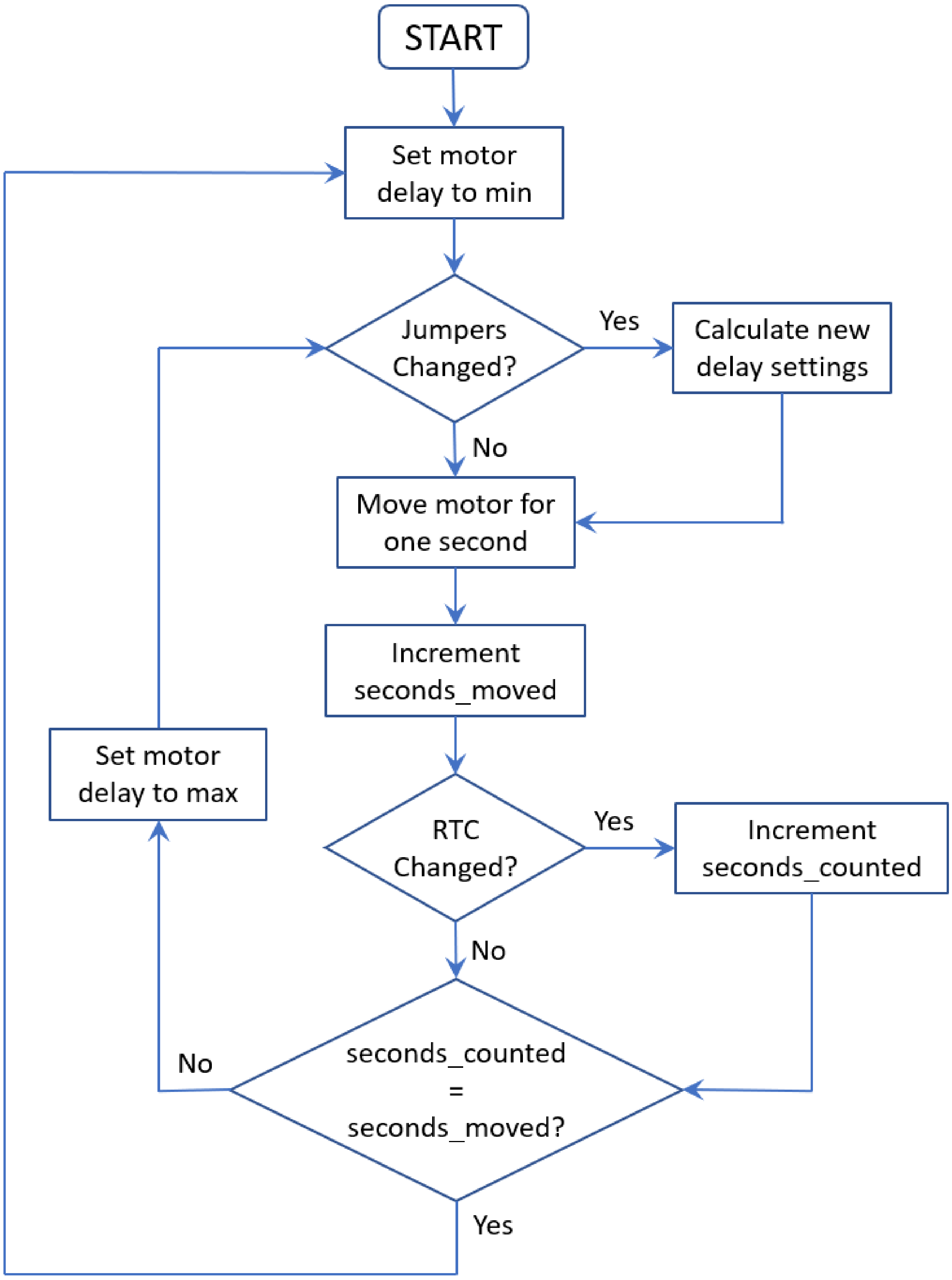

Figure 7: Algorithm Flowchart

Printing New Parts

The CNC Shield V4 board is significantly larger than the original design. A larger base and a few additional components were designed to accommodate the larger board. The upper portion of the clock remains the same as the previous design except for a new counter-weighted second hand.

Print the following components to replace the equivalent parts in the original design:

File Name	Replaces	Print	Time	Filament	Notes
base_back_power_shield	base_back_power	1	9h 48m	45.39	Base with power in the back
base_side_power_shield	base_side_power	0	9h 49m	45.05	Alternate base with power on the side
hand_second_weighted	hand_second	1	0h 24m	1.12	New counter weighted second hand that helps reduce gear noise. Print with 14 perimeters so it is completely solid.
holddown_back_shield	N/A	1	1h 29m	4.97	Retaining clip to use with base_back_power_shield
holddown_side_shield	N/A	0	1h 19m	4.55	Retaining clip to use with base_side_power_shield
motor_mount_33mm	motor_mount	0	2h 21m	11.15	updated motor mount for 33mm short body motors
motor_mount_39mm	motor_mount	1	2h 21m	11.11	updated motor mount for 39mm long body motors
Total		**4**	**14h 2m**	**62.59m**	

Table 2: Component list and print times

Most parts are optimized to print using 4 perimeters with a 0.4mm nozzle. The counter weighted second hand needs 14 perimeters or 100% infill to make it completely solid so it is properly weighted. The base print time can be reduced by using 3 perimeters and 15% infill.

The motor mount should be selected based on the motor size that you ordered. The electronics work with many different motor types and sizes as long as they are NEMA17. The hole for the wires to pass into the base is the only difference between the two motor mounts. It is OK to manually enlarge the hole using a Dremel or chisel if needed.

Assembly and Debug

Assemble the clock using the same procedure listed in the original assembly guide. The only differences are related to the larger base holding the new electronics.

Start the debug using these steps:

1) Rotate the stepper motor 90 degrees when adding it to the motor holder so the motor cable will run to the side instead of into the base. This will allow you to adjust the configuration without taking the clock apart. Use two M3x6mm or M3x8mm socket head screws to attach the motor.

2) Place the stepper motor and motor holder onto the empty base. Add the top portion of the clock using the same procedure as listed in the original assembly guide, but with the wiring and electronics sitting outside the base.
3) Position gear 1 onto the stepper motor shaft to be centered with gear 2. It should attach using a M3x10mm socket head screw. It is OK for the stepper motor shaft to extend through gear 1.
4) Program the Arduino Nano and make sure the clock runs properly. You may need to configure the direction jumper to make the motor rotate in the proper direction.
5) Adjust the Vref potentiometer to the lowest possible setting where the clock still runs.
6) Check the serial monitor to ensure that the real time clock is properly being tracked.
7) Compare the clock speed to make sure that the speed selection jumpers are set properly for your clock.

Assemble the completed clock after the previous debug steps are finished.

8) Remove the base from the debug setup and rotate the stepper motor to the proper orientation with the wires passing into the base. It is OK to enlarge the hole on the motor holder to make room for the cable if your motor size is slightly different.
9) Place the CNC Shield into the base and add the holddown to keep the electronics in place. Make sure that the USB cable will insert into the Arduino Nano. This new design supports Arduino Nano controllers using either mini USB cables or USB-C cables.
10) Plug the stepper motor cable into the stepper motor and CNC Shield V4 header. Coil the cable to get the motor cover to fit onto the base.
11) Add the top portion of the clock and secure it using four screws from the bottom.
12) Add second_hand_weighted to the clock instead of using the second hand from the original design. This new counter-weighted second hand helps keep the clock running quietly.
13) Plug the clock in using a dedicated USB charging adapter. The USB plug in my PC can pause the clock for a second or two whenever the PC comes out of standby.
14) Enjoy your new super quiet and super accurate desk clock.

Final Comments

This new circuit appears to really meet the primary design goal of having a silent desk clock. The previous design seemed quiet at first, but had just enough gear noise that I wouldn't it next to my bed.

The second goal of improving accuracy was also met. The DS3231 real time clock should have an accuracy of a minute or two per year. I have not detected more than a few seconds of drift in over a month of runtime.

The design uses parts that are easier to find with cheaper shipping costs for builders around the world.

A larger version of the clock using the old driver design was too noisy to release when SP6 was designed. This new circuit will allow it to be released. A wooden gear version of the clock is also being planned.

Stay tuned for new releases using this new and improved clock motor driver.

Steve