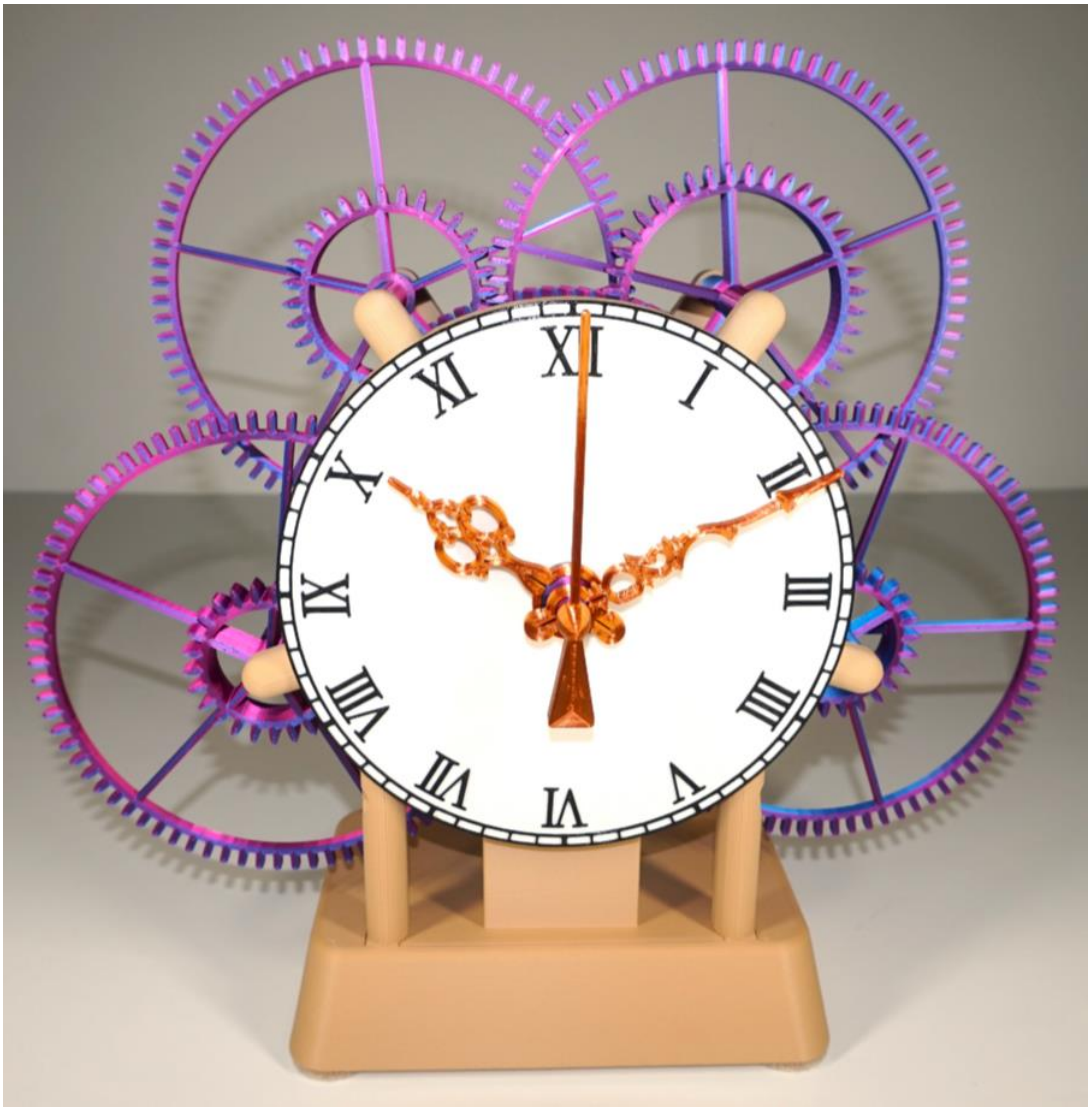


MOTOR DRIVEN DESK CLOCK

SP9 Assembly Notes

Instructions for building a large 3D printed desk clock
with an ultra-quiet stepper motor drive circuit

Steve Peterson
15-Apr-2023

Contents

Tables

Figures

Revision History

10-Feb-23	Original version.
15-Apr-23	Corrected gear names in print time table on page 20.

Description

This assembly guide describes a 3D printed desk clock with large exaggerated gears extending well beyond the dial. The new drive circuit provides silent power using a precision reference to maintain an accuracy of about a minute per year. This clock looks great sitting on any desk.

I started the design of this clock over three years ago, but it never felt complete until I discovered the new silent stepper motor driver. The original motor control circuit produced a slight rumbling noise. The recent discovery of an ultra-quiet drive circuit made this clock feasible to finish. The small version of the clock was enhanced to include the quiet drive circuit, but I prefer this larger more impressive design.

This has become my new favorite clock since it is quiet enough to sit in any room and maintains incredible accuracy. The motorized drive circuit never needs winding.

The following criteria were used while designing this clock:

1) The clock should look good.
2) It must be accurate.
3) It must be quiet.
4) Avoid using custom circuits or components.
5) The design should not be dangerous. No high voltages. The motor should have a low enough torque that it stalls if anything is stuck in the gears. etc.

Large exposed gears help satisfy the first criteria to look good. The solid dial makes it easy to read the time. The first prototype used a large dial, but I think it looks much better with a smaller dial to shift the focus back to the large gear train. The new control circuit satisfies all the remaining criteria. A precision real time clock reference holds incredible accuracy. The TMC2208 stepper motor driver provides smooth silent motion. All components used in the clock are readily available at a reasonable cost.

The overall size is around 12.4” by 11.8” and 5” deep (315mm x 300mm x 125mm). The design prints best on a Prusa MK3S (250x210mm) or Ender3 (220x220mm) sized printer. The largest components have been partitioned so they can be printed on a Prusa Mini or similar machine with at least a 180x180mm print area.

Components

Here are the non-printed parts required to build the clock. Prices are listed for the electrical components ordered on Amazon around Dec 2022.

Component	Cost (USD)	Notes
CNC Shield V4	$10.96 for 3	Main circuit board that all components plug into.
Arduino Nano	$16.99 for 3	Processor that controls everything. The design supports both mini-USB and USB-C connectors. Either style will work.
TMC2208	$21.99 for 6	Ultra-quiet Trinamic stepper motor driver.
DS3231 real time clock	$12.99 for 4	Precision real time clock to provide a 1Hz heartbeat to the Arduino Nano. See the pictures a few pages ahead.
NEMA17 stepper motor	$10.99 each	This design supports almost any NEMA17 bipolar stepper motor. 2-3 ohm motors rated for 1-2A with body lengths of 33-39mm are preferable. The cable should have a standard 4 pin header with 0.1" (2.54mm) spacing.
Mini USB cable or USB-C cable		For powering and programming the Arduino Nano. Select the style that matches your Arduino Nano.
USB power source		To power the clock. Current is around 100mA.
4X #6x3/4" flat head wood screws (metric M3x20mm or M3.5x20mm)		Frame is assembled using small flat head wood screws. The segmented frame for smaller printers needs an additional 3 screws.
18" X 1/16" music wire (metric 0.5m X 1.5mm)		Arbors are made using hardened music wire rod cut into short segments (described later). Add an additional 18" (0.5m) if printing the segmented frame on a small printer.
1X small spring		Use a spring from a ball point click pen for the friction clutch.
1X 1/16" or 1.5mm shaft collar		Optional component used as part of the friction clutch. A printed component is included that can also be used.
6X M3x8mm screws		Socket heads are best, but most small diameter head styles are OK. M3x8mm length is preferred, but M3x6mm or M3x10mm may also work.
4X 1" felt pads		Optional pads for under the base of the clock.

Table 1: Non-printed components

Notes regarding a few items:

1) The CNC Shield V4 was designed to connect an Arduino Nano to 3 A4988 stepper motor drivers. We only need a single stepper motor driver and will be using a TMC2208 instead of the A4988. The CNC Shield V4 has a well-known bug that needs to be fixed. It is described in the next section. A modified design (black and yellow) without this bug can be purchased from KeyeStudio on AliExpress, but the fix is super easy so I recommend the cheaper red version.
2) 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 newer USB-C connector. The clock base can be printed to support either connector style. USB-C cables are easier to find, so the Arduino Nano with USB-C connectors are slightly preferrable.
3) The TMC2208 stepper motor driver is amazing. It supposedly supports up to 256 position micro-stepping, but 16 positions is good enough to drive the clock and easier to configure. The

TMC2208 driver will be wired to bypass the internal regulator and run directly on 5V. This allows the clock to run using a simple USB power cable. Vref will be set to the lowest possible motor current.

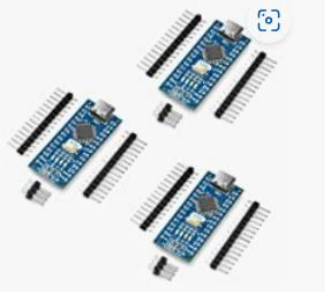
4) The DS3231 real time clock used is often listed as “DS3231 for Raspberry Pi”. It has a 5 pin female socket and yellow battery (or capacitor?). It has voltage and temperature compensation to maintain an accuracy of about a minute per year. A slight wiring modification allows the 5 pin device to plug into one of the unused 4 pin stepper motor headers.
5) Nearly any NEMA17 stepper motor can be used in this clock. The biggest limitation are the physical size. They should have a single shaft and a body length of less than 40mm. Many common 1.5A motors with 2-4 ohm winding resistance will work great. I tested StepperOnline part number 17HS15-1504S (1.5A, 2.0 ohms) and a short body 17HE08-1004S (1.0A and 3.6 ohms). Many other stepper motor brands will also work. I have not found any NEMA17 stepper motor that would not work. The only limitations are the physical size (body length under 40mm and a single shaft). The base works best using motors with a body length of either 33mm or 39mm. The 34 ohm stepper motors used in the original design can be used, but the more common 2-4 ohm stepper motors are preferred.

Here are some sample pictures of the electrical components on Amazon. Many are sold in quantities of 3 for a tiny bit more than the single prices. Many builders seem to make more than 1 clock, so go ahead order the bundles and build a few clocks. The first word in the descriptions (AITRIP, OSOYOO, Dorhea, etc.) seems to be random supplier names and not a great search keyword.

The CNC Shield V4 is often sometimes comes bundled with A4988 drivers. Order the bare versions since we will be populating them with TMC2208s instead. Make sure to order CNC Shield V4 and not CNC Shield V3.

Figure 1: CNC Shield V4

Order the Arduino Nano with pre-installed headers if you can find them for a good price. The ones listed below are $5.66 each for bare boards or $6.80 for pre-installed headers.



OSOYOO 3pcs LGT Nano for Arduino Nano Compatible with ATmega328p Chip Nano Board with USB-C Interface

4.5 stars 26

$16.99 $18.99

AITRIP 5pcs for Nano Board CH340/ATmega+328P Without USB Cable, Type-C Connection Compatible with Arduino Nano V3.0,Welded Module

4.4 stars 76

$33.99

prime One-Day

FREE delivery **Tomorrow, Feb 10**

Figure 2: Arduino Nano

The TMC2208 is usually sold in bundles of 4 to 6 for use in 3D printers.

6PCS TMC2208 3D Printer TMC2208 V1.2 Stepper Motor Driver, TMC2208 Stepstick Stepper Motor Driver Module with Heat Sink Screwdriver for 3D Printer Controller...

4.4 stars 18

$21.99

prime

FREE delivery **Sat, Feb 11**

Figure 3: TMC2208

This is the real time clock used as a time reference. Several other RTC styles will show up in the search. We need to use the one shown below with a 5 pin socket.

Dorhea 4PCS DS3231 Real Time Clock Module RTC Clock Memory Module 3.3V/5V for Raspberry Pi

4.5 stars 136

$12.99

prime

FREE delivery **Sat, Feb 11**

Figure 4: DS3231 RTC

Any of the following style stepper motors should work. They are the same ones used in many 3D printers. Make sure they have cables. The controller end needs to have a 4 pin socket to plug directly into the CNC Shield V4. A connector at the motor is helpful, but not critical.

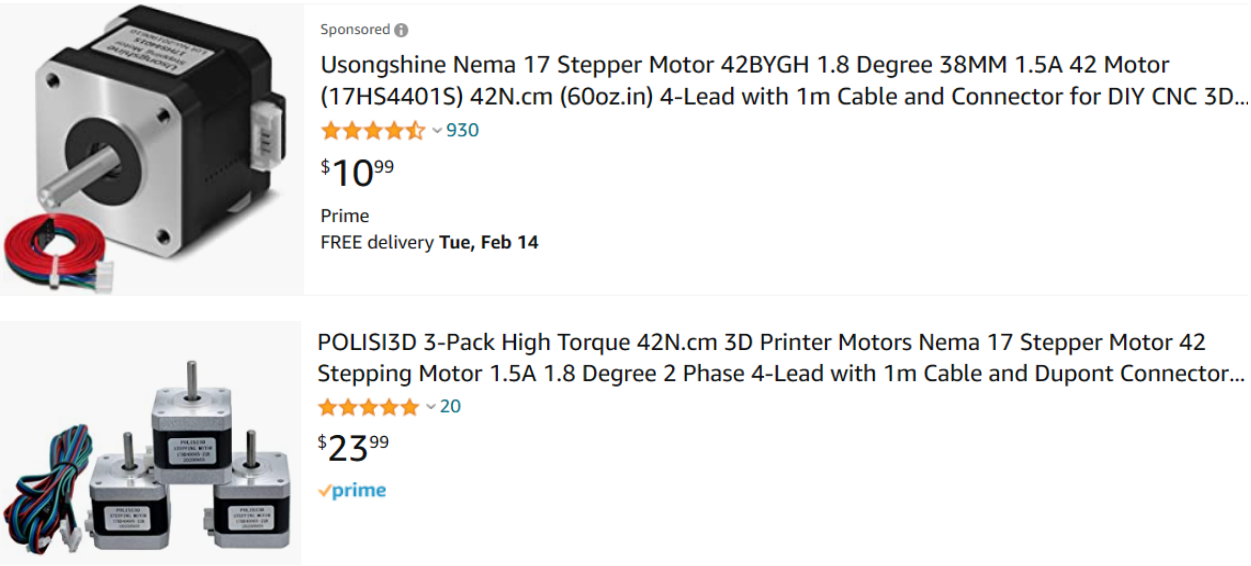


Figure 5: Stepper motors

Wiring

This section describes the wiring modifications to be made to the CNC Shield V4 and the DS3231 real time clock. A small custom circuit board could be designed to work without these edits, but the cost would be higher than the easily acquired standard parts.

The most common (and cheapest) red colored CNC Shield V4 boards have a bug that needs to be fixed to enable micro-stepping. KeyeStudio sells a black colored board without the bug, but it is harder to find, so I will describe how to modify the red board.

Make the following edits using the pictures on the next few pages as a reference.

1) Remove the 3 jumper blocks under the lower right TMC2208 socket. This prevents shorting Vcc and Gnd when step #2 is completed.
2) Solder a jumper wire on the back of the board from the 3 jumper pins to the nearest Vcc connection. We always want 16X micro-stepping, so the three jumpers will be hard wired to Vcc. We only need to fix the driver module that will be populated with a TMC2208.
3) 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 #2 to the capacitor on the left. This edit allows the clock to run on a standard 5V USB power source.
4) Connect the SDA/SCL serial port wires to an unused stepper motor header. We will be using the motor header that is straight down from the serial port header. The SDA and SCL connections are about 1” long.

5) Connect the real time clock Vcc and Gnd pins to the nearest supply pins. Edit #3 shorts Vcc and Vmot, so the closest Vcc connection is the former Vmot pin on the left.
6) 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 showing edits #2 through #5.

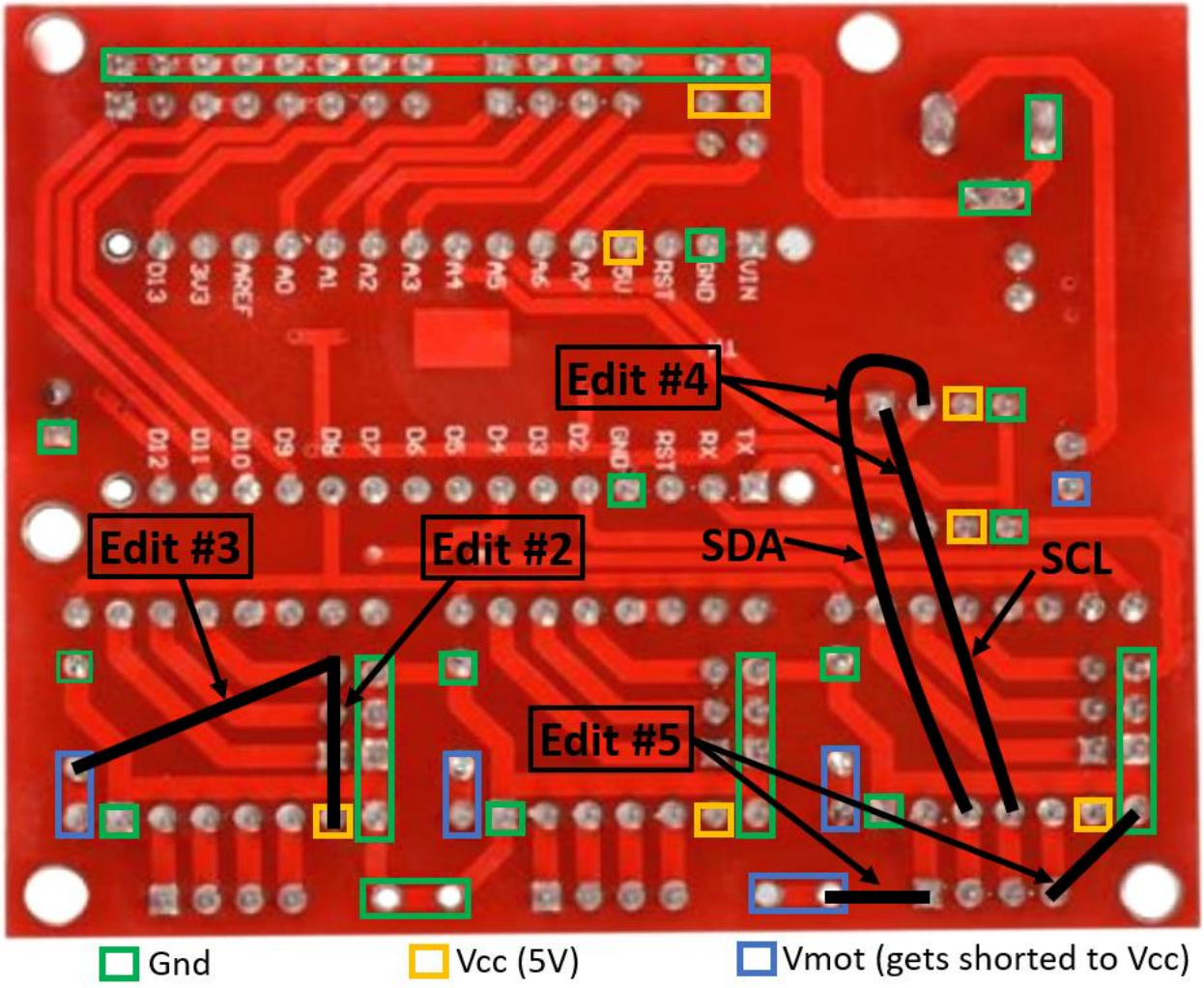


Figure 6: CNC Shield V4 Wiring Modifications

The top side of the board only needs edits #1 and #6.

Insert components into the CNC Shield V4 to match the picture below. The Arduino Nano will need the header pins soldered on if it didn't come pre-assembled. The USB port points to the right.

Add the heat sink to the TMC2208 to minimize any heat buildup. Turn the current adjust potentiometer all the way to the left so the motor currents are as low as possible.

Insert the DS3231 real time clock into the lower left stepper motor header. It is inserted to only use the pins labeled NC, C, D, and "+". Edit #6 connects the Gnd pin to the NC position. This allows the 5 pin RTC to plug into the 4 pin header.

Save the jumpers that were removed from below the TMC2208. They can be used as motor direction and speed selection jumpers.

Here is a picture of the top side of the board.

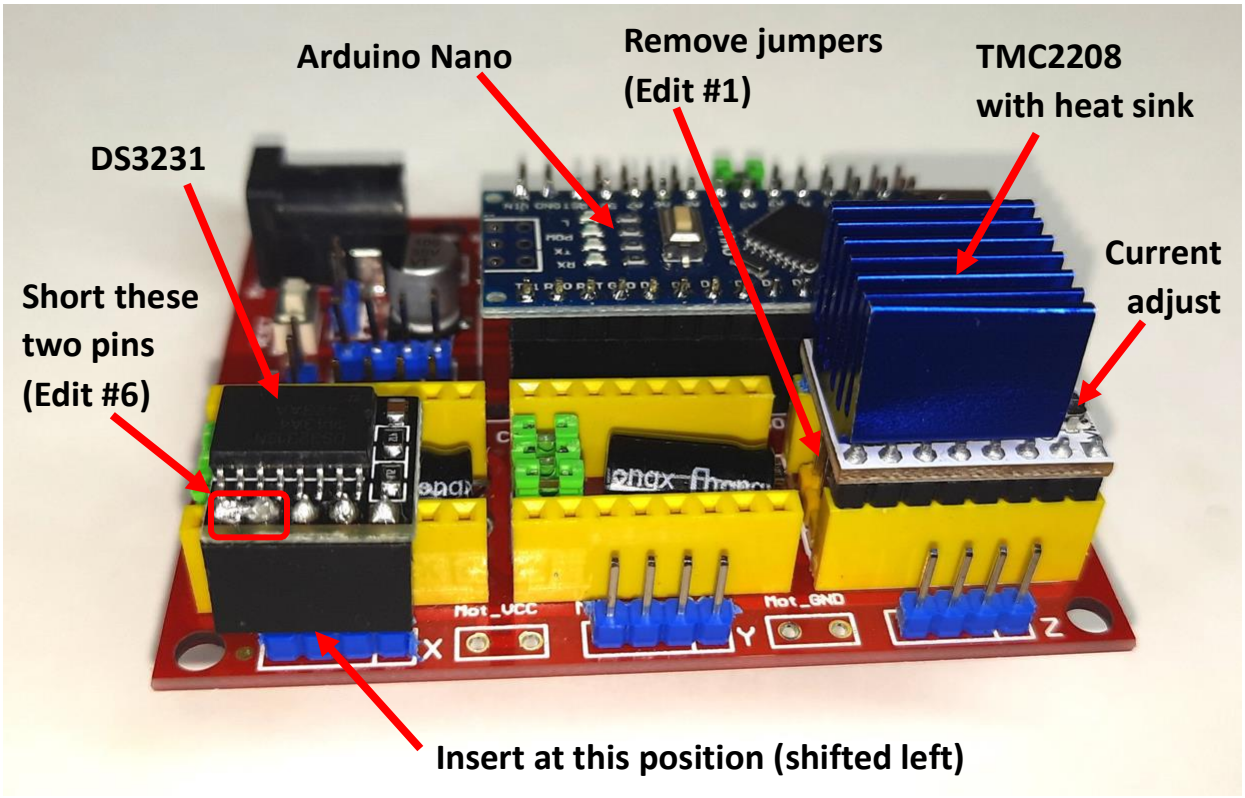


Figure 7: Top Side Close-up of Components

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. The base has room for it, so add it to the TMC2208, leaving the current adjustment potentiometer accessible.

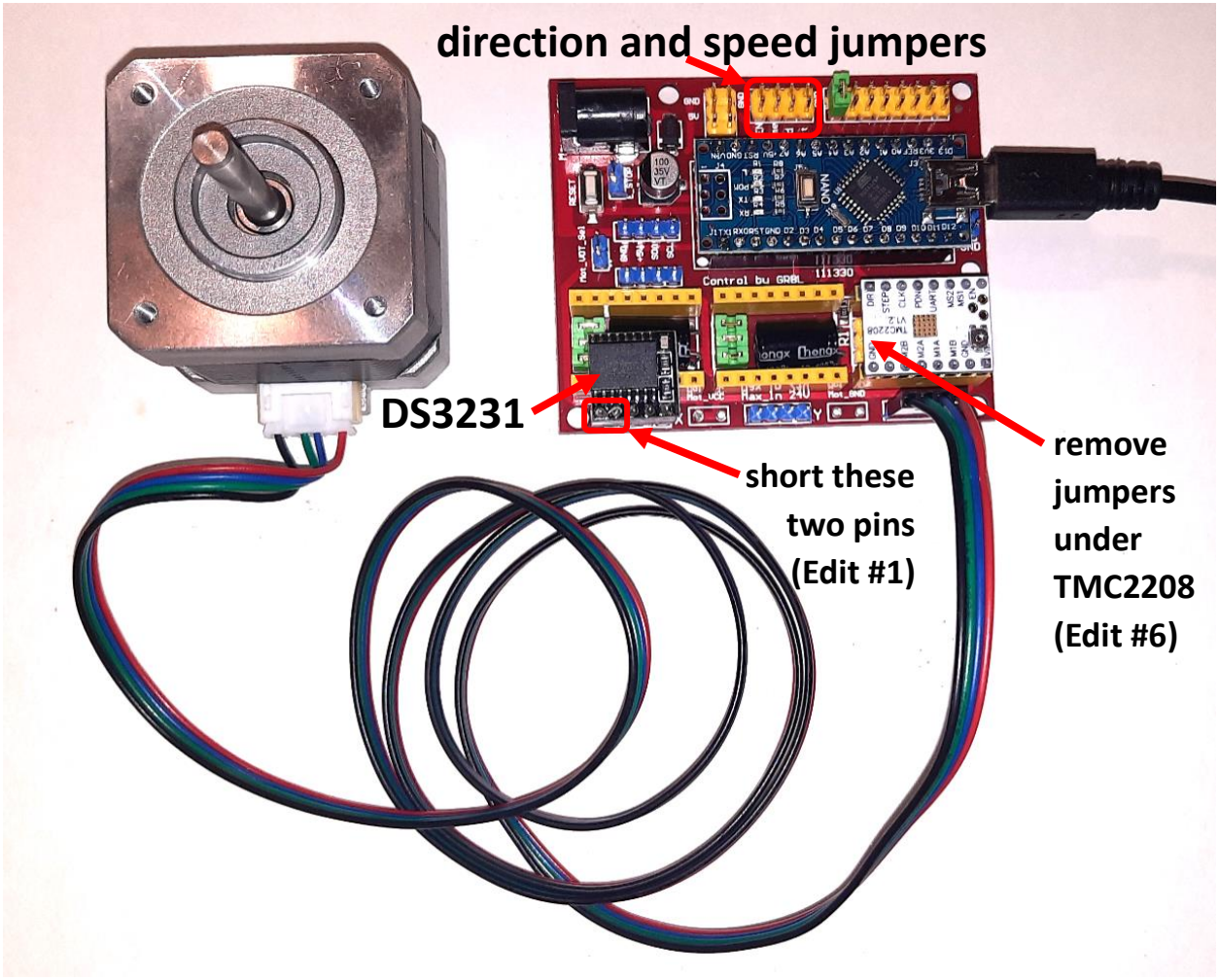


Figure 8: Assembled Driver Ready to Test

Motor Test

It is a good idea to test the circuit before assembling everything into the 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 wiring modifications previously described and plug everything together as shown in the diagram on the previous page.

Here are the steps to program the Arduino Nano:

1) Perform the previously listed wiring modifications to the CNC Shield V4 and DS3231.
2) Assemble the CNC Shield V4.

    2a) Solder the header pins onto the Arduino Nano if it doesn't already have them.

    2b) Insert the Arduino Nano into the CNC Shield V4 in the orientation shown.

    2c) Insert the TMC2208 into the lower left socket. Pay attention to the orientation.

    2d) Add the small heat sink to the top of the TMC2208.

    2e) Turn the Vref potentiometer all the way to the left for the lowest possible current.

    2f) Add the stepper motor cable to the header by the TMC2208.

    2g) Add the DS3231 RTC in the position shown in the pictures.

3) Download the Arduino IDE at https://www.arduino.cc/en/software selecting the option for Windows, Linux, or Mac.
4) Plug the Arduino Nano to your computer using a mini USB or USB-C cable.
5) Open the IDE and configure for Arduino Nano.

    5a) Tools → Board → Arduino Nano

    5b) Tools → Processor → ATmega328P (Old Bootloader)

    Note: your board might use the old or the new bootloader. Select the one that works.

    5c) Tools → Port → "Varies, usually COM3 or COM4 on my PC, and never COM1"

    5d) Tools → Programmer → USBasp

6) Download the CNC Shield V4 Arduino sketch from https://www.stevesclocks.com/sp9 and load it into the IDE. The file extension might need to be renamed from *.txt to *.ino. Alternatively, open the file in a text editor to cut and paste the code into the IDE.
7) Install the real time clock library.
    7a) Sketch → Include Library → Manage Libraries…
    7b) Search for RTC and install the RTC library by Manjunath CV.
    7c) Close the library manager popup window.

8) Verify (compile) the code by clicking the "check mark" icon in the upper left corner of the IDE.

9) Upload the code by clicking the "➔" icon in the upper left corner of the IDE. The Arduino LEDs should blink for a second or two as the algorithm is uploaded.
10) Open the Serial Monitor debug port by clicking the "magnifying glass" icon in the upper right corner of the IDE. Alternatively, open the window using "Tools → Serial Monitor". A debug window should pop up showing the parameter settings and tracking indicators that updates once per second.

If everything works as expected, the stepper motor should start rotating at a few revolutions per minute. The exact speed is determined by the speed jumpers.

Follow the debug steps in the next section if it does not rotate.

Motor Debug

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

Adjust the TMC2208 potentiometer slightly higher if the motor seems to move but has almost no power. I find that many motors will operate at the lowest possible setting or close to the lowest setting. We want to keep the motor current low to minimize heat and avoid overloading the USB power supply. The clock needs very little power, so use the lowest stable setting. The motor should spin but be easy to stop 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 to swap the middle two wires on the 4 pin connector. See the photos on the next page.

Gently lift the tab holding the wires in place and remove the center two wires. Lift the tabs just enough to pull out the wires and the attached connectors. Swap their positions and insert them back into the 4 pin connector.

Here is the original unmodified stepper motor cable

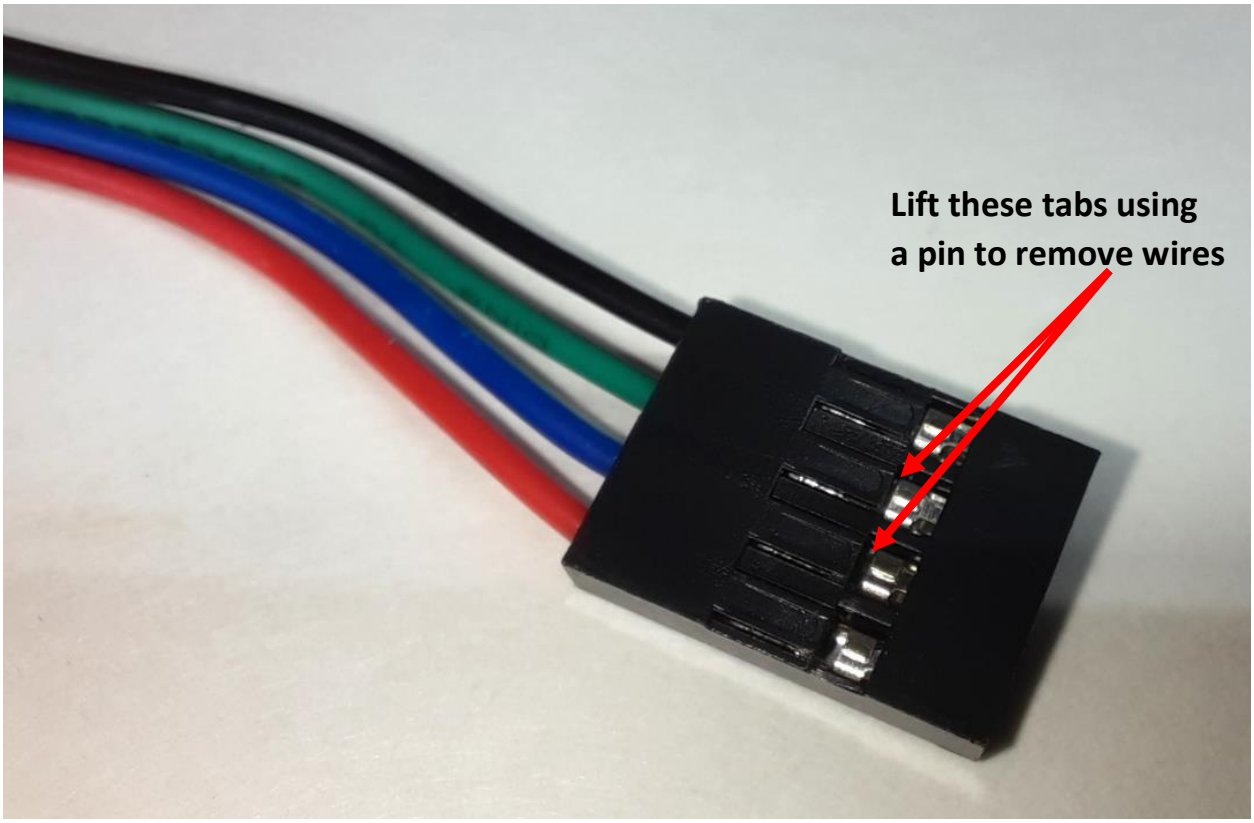


Figure 9: Standard Stepper Motor Cable

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

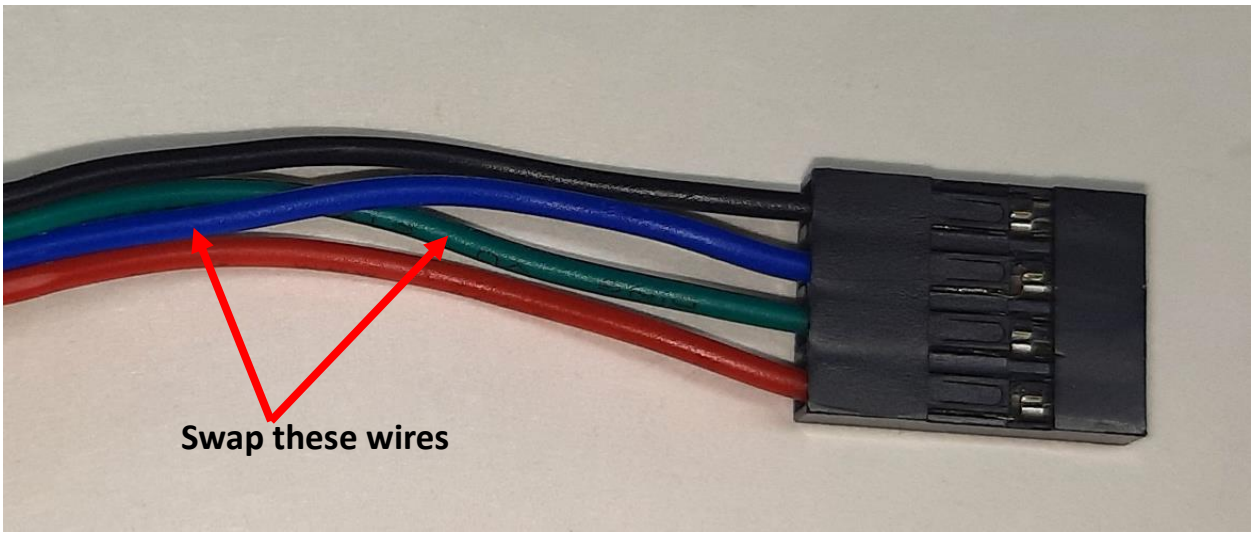


Figure 10: Modified Stepper Motor Wiring

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.

Speed Selection

The exact speed and direction of the motor is determined by four jumper pins in the 8-pin header near the top of the CNC Shield V4. The left-most jumper determines the motor direction. Add or remove this jumper so the motor spins clockwise when it is sitting on the test bench.

The remaining three jumpers select one of eight motor speeds. This clock has an 18 tooth pinion on the motor driving an 84 tooth main gear. The motor rotates at 4.667RPM using jumper settings of 100 ("in", "out", and "out"). Other clock designs with different gear ratios would need different jumper settings.

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:

Setting	Jumpers	RPM	Examples	Notes
0	000	2.4	24:10, 36:15	
1	001	2.667	32:12	Small wood clock might use this setting
2	010	3	30:10, 45:15	
3	011	3.6	36:10, 54:15	SP6 desk clock uses this setting
4	**100**	**4.667**	**56:12, 84:18**	**SP9 large printed clock uses this setting**
5	101	5.4	54:10	Large wood clock might use this setting
6	110	10		
7	111	60		Fast mode for debug

Table 2: Speed selection jumpers

Using speed selection jumpers allows the CNC Shield V4 to be moved between different clocks without having to re-program the Arduino Nano.

Debug Monitor

The Serial Monitor debug window is very useful to see when the algorithm is working properly and the stepper motor is tracking the real time clock. Click the icon that looks like a magnifying glass in the upper right corner if the IDE window. A window will open with debug statements sent by the Arduino using Serial.print commands.

The debug monitor will show the program name and sketch revision. The next few lines show the motor delay parameters. Everything after this is a monitor 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 status LED on the Arduino Nano will also blink every time a "+" is displayed.

The serial monitor might look like this:

```
CNC Shield V4 clock movement - Rev 1.02

speed 4
steps 248 (plus 8/9)
min_delay 1915
max_delay 2036

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

The header shows the motor configured as speed setting of 4. The motor uses 248 full steps per second plus 8 extra steps every 9 seconds. This corresponds to 248.888 steps per second and an overall motor speed of 4.667RPM. The motor will toggle between delay values of 1.915ms and 2.036ms to meet the target speed.

Algorithm

The clock algorithm is fairly simple. It uses two delay values. The minimum delay is slightly fast and the maximum delay is slightly slow. The algorithm switches between fast and slow delays as needed to track the real time clock. The speed difference is not noticeable and the second hand only deviates from the target position by a fraction of a second.

The algorithm operates similar to following a car that is travelling at exactly 60MPH when your car can travel either 59MPH or 61MPH. Start at a fixed distance behind the pace car. Travel at 59MPH until you fall behind, then switch to 61MPH until you catch up. Your average speed will be 60MPH.

The actual algorithm allowing the motor to track the real time clock is only about 20 lines of code. The rest of the code is overhead for setup, reading the speed jumpers, and debug monitor output.

The flowchart is shown below:

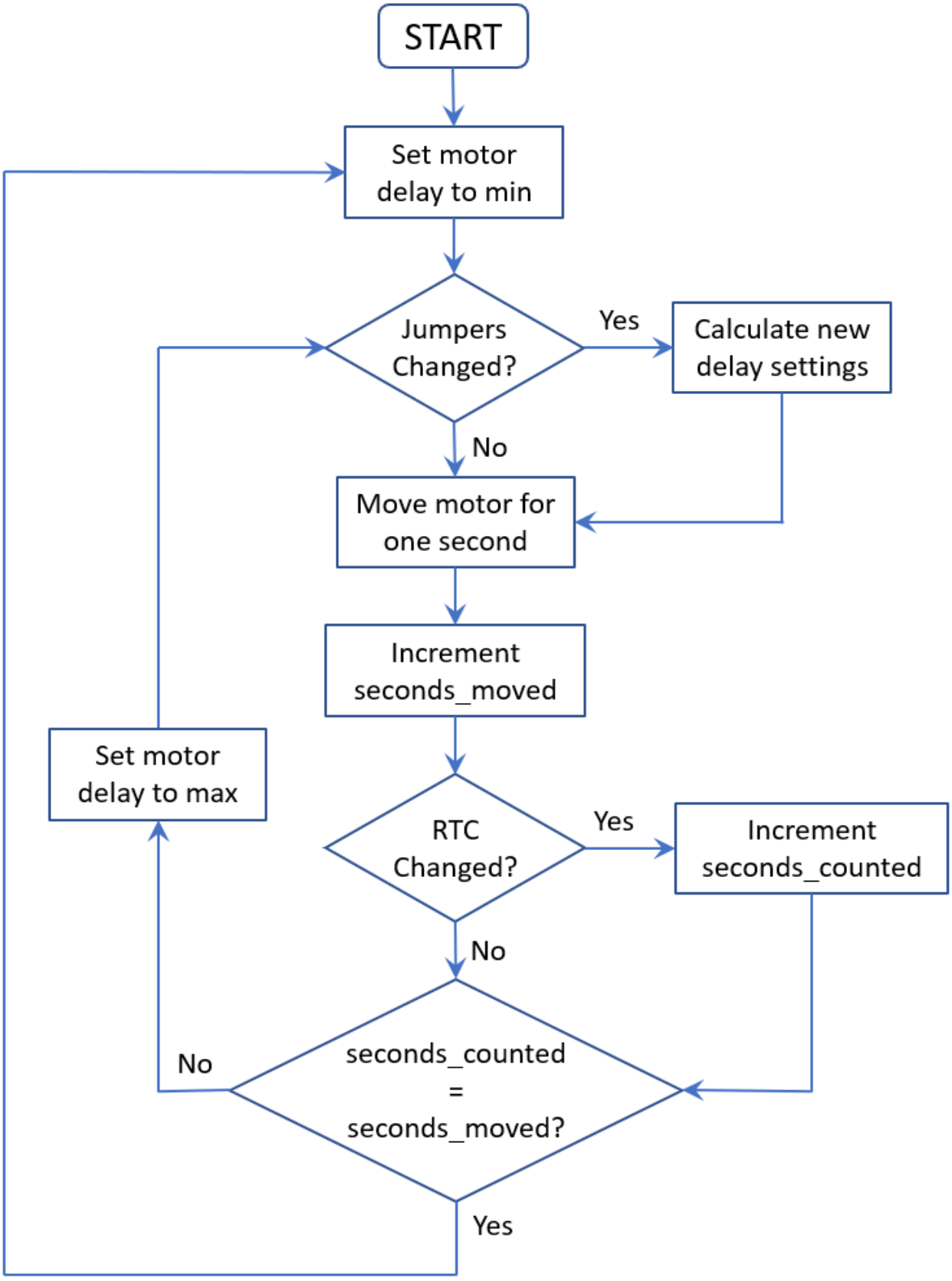


Figure 11: Algorithm flowchart

Cut Metal Parts

This clock has a relatively simple metal parts list. The design works equally well with 1/16” or 1.5mm music wire since they are very close to the same size. Use a 1/16” or 1.5mm drill to enlarge any hole sizes to fit if needed. Most parts should be a loose fit on the arbors.

Music wire comes in a hardened state which is great for the clock, but can be difficult to cut. A Dremel cut-off disk or cutters with hardened jaws will work well. Cheap wire cutters might not be tough enough.

The number of cut metal parts is smallest is you can print the one-piece frame components. The split back frame needs two additional alignment rods and the split front frame needs four additional rods. The split frame performs identically, with the exception of a few slightly visible screw heads.

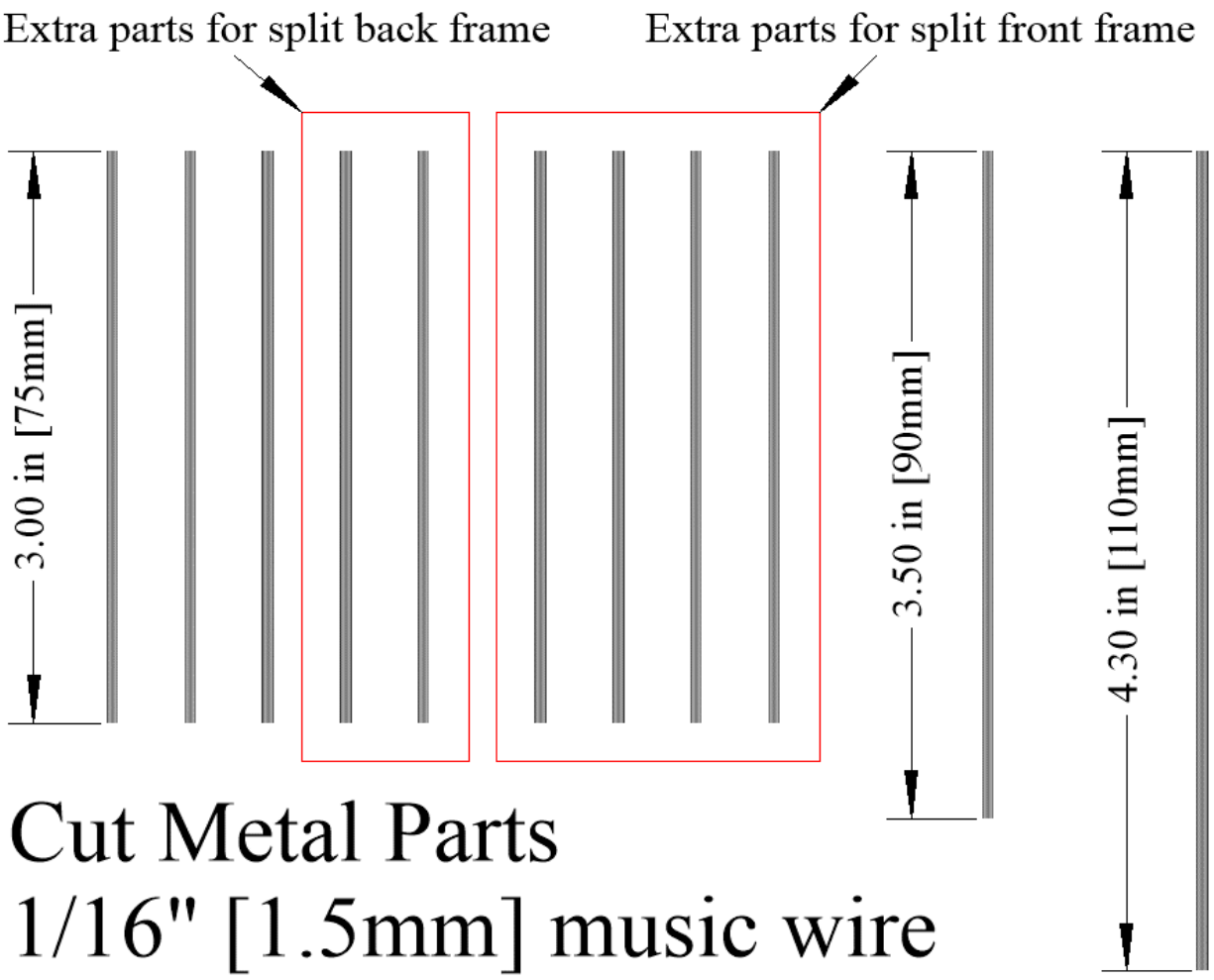


Figure 12: Cut metal parts list

Tools Required:

A few simple tools are required for building the clock.

1) Phillips screwdriver
2) Hex wrench for M3x10mm screws
3) Hacksaw or Dremel cut-off disks for cutting music wire
4) Soldering iron for assembling stepper motor driver
5) Fine tooth hand files or sandpaper may be needed to clean up some of the printed parts
6) 1.5mm or 1/16” drill bit for cleaning up printed holes

Printing the Parts

Print one of each clock part from the following table. Some parts are provided with multiple configurations, so select the one that works best for you.

I print everything using PLA with a 0.4mm nozzle, 0.2mm layer heights, 3-5 perimeters, 6 bottom layers, 7 top layers, 20% cubic infill, combine infill every 2 layers, random seams, and 0.08mm elephants foot compensation. The default orientation is usually optimal with the largest surface already facing down. Supports are never needed. Most parts print great with the new Arachne slicing engine except the gears that print better using the classic slicer.

A few parts have options depending on your printer size or other requirements. The base can be printed with the power plug at the back or either side if you want to place the clock on a narrow shelf. The base is optimized for either mini-USB or USB-C power plug. The holddown clips used to keep the CNC Shield V4 positioned properly will also be different depending on the position of the power plug.

The frame is provided as large pieces or segmented into components that can be printed on smaller printers. The one-piece back frame is 218x182mm or 175x155mm when printed as “split” components. The one-piece front frame is 218x182mm or 174x170mm when split. The one-piece versions are preferred if your printer is large enough. They will have fewer assembly steps and will have less visible screws.

Part Name	Color	Print	Time	Filament	Notes
base_back_mini	tan	1	9h 34m	149.82g	Print one of any style depending on where you want the power plug (back, left, or right) and the type of power plug in your Arduino Nano (mini USB or USB-C)
base_back_usbc	tan		9h 34m	149.77g	
base_left_mini	tan		9h 37m	148.31g	
base_left_usbc	tan		9h 38m	148.29g	
base_right_mini	tan		9h 37m	148.31g	
base_right_usbc	tan		9h 38m	148.29g	
frame_back	tan, purple	1	6h 21m	75.71g	4 perimeters, add a color change at 10.40mm
frame_back_split_lower	tan, purple	0	4h 1m	44.75g	Optional split frame, 4 perimeters, add a color change at 10.40mm
frame_back_split_upper	tan, purple	0	2h 58m	36.23g	
frame_dial_roman	tan, ivory, black	1	7h 1m	132.14g	4 perimeters, add color changes at 10.40mm and 12.20mm
frame_dial_roman_split	tan, ivory, black	0	6h 16m	122.88g	Optional split frame, 4 perimeters, add color changes at 10.40mm and 12.20mm
frame_front_split_left	tan	0	0h 43m	7.71g	
frame_front_split_right	tan	0	0h 43m	7.71g	
gear1_18	purple	1	0h 59m	7.33g	Classic slicer, 5 perimeters
gear2_84_20	purple	1	3h 43m	33.66g	Classic slicer, 5 perimeters
gear3_80_32	purple	1	3h 40m	30.89g	Classic slicer, 5 perimeters
gear4_80_40	purple	1	3h 49m	33.52g	Classic slicer, 5 perimeters
gear5_25	purple	1	1h 7m	8.01g	Classic slicer, 5 perimeters
gear5_80	purple	1	1h 39m	16.95g	Classic slicer, 5 perimeters
gear5_collar	purple	0	0h 7m	0.53g	Optional printed shaft collar
gear5_knob	purple	1	0h 17m	2.98g	
gear6_75_20	purple	1	2h 57m	23.73g	Classic slicer, 5 perimeters
gear7_80_25	purple	1	2h 32m	23.62g	Classic slicer, 5 perimeters
gear8_75	purple	1	1h 50m	16.86g	Classic slicer, 5 perimeters
hand_hour_gothic	copper	1	0h 20m	2.31g	
hand_minute_gothic	copper	1	0h 17m	1.93g	
hand_second	copper	1	0h 24m	3.35g	Print with 14 perimeters
holddown_back	any	1	1h 38m	17.00g	Print one of these to match base power plug direction
holddown_left	any		1h 28m	15.17g	
holddown_right	any		1h 28m	15.18g	
motor_mount_33mm	tan	1	2h 54m	41.31g	Print one of these to match your motor body length
motor_mount_39mm	tan		2h 53m	41.15g	
Total		**18**	**52h 29m**	**636.13g**	

Table 3: Component names and print times

The frame and base can be printed using somewhat generic settings. I like to use at 4 perimeters for the frame to provide extra strength. The gear teeth only need 3 perimeters, but the center hub in each gear is designed to print best using 5 perimeters, so use 5 perimeters for the gears.

The only other part with special print requirements is the counter weighted second hand that needs either 14 perimeters or 100% infill so the weighted end is completely solid to keep it balanced.

This is what the gear teeth should look like in the slicer.

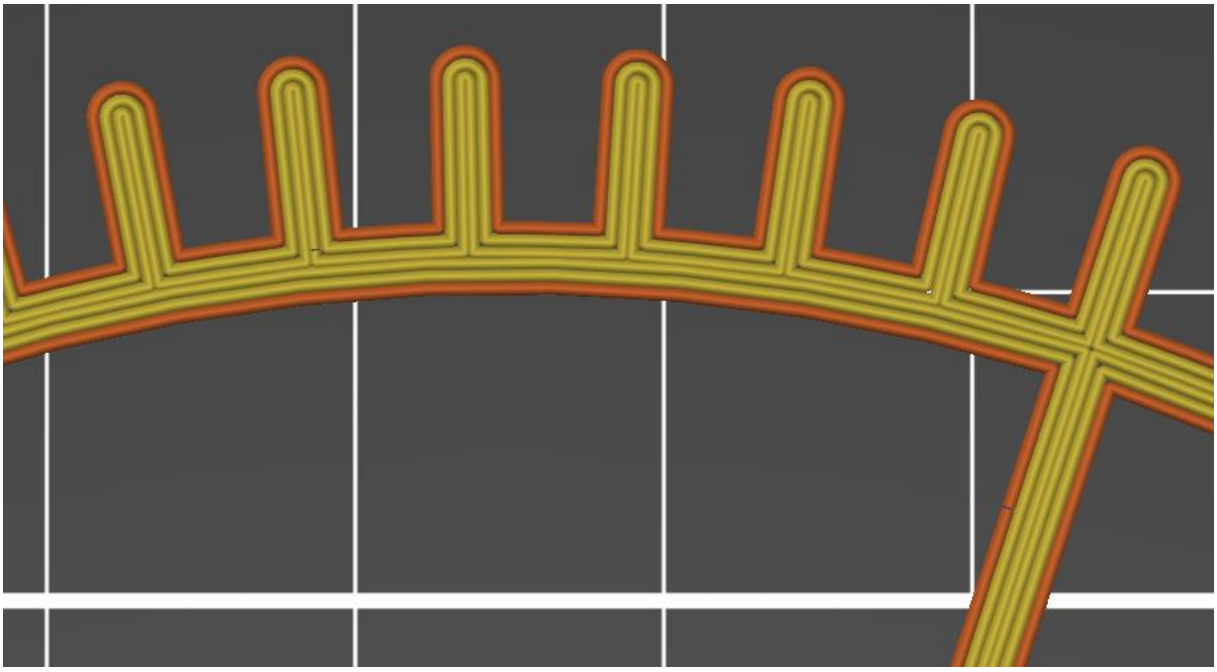


Figure 13: Gear slicer detail with five perimeters

And the center hub that prints solid with 5 perimeters.

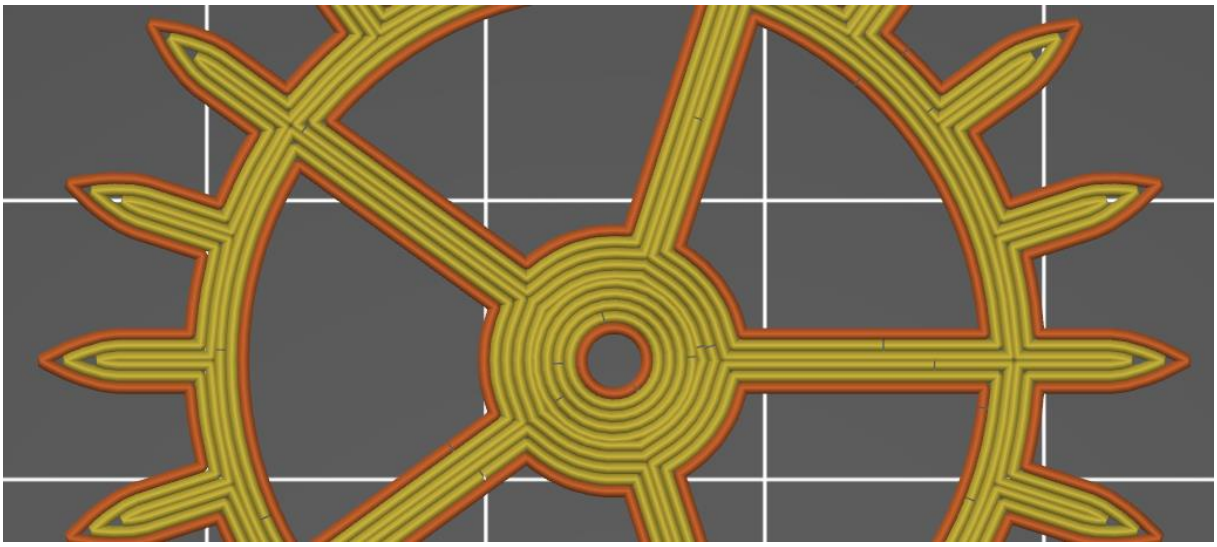


Figure 14: Gear center hub

Notes about Arachne

A recent slicer enhancement added the Arachne perimeter generation algorithm. This is the only algorithm used in the newer Cura versions and default in PrusaSlicer. Most parts print better using Arachne. However, there are a few parts that print better **without** Arachne. The gears in this clock are designed and optimized for perfect printing **without** Arachne. The rounded internal perimeters added by Arachne create extra gap fill areas that cancel out most of the benefit of the clean gear tooth profiles. The additional gap fill increases print time and increases stringing resulting in rougher gear surfaces.

This is how the gears will print using the Arachne slicing engine. Turning off Arachne will result in the profiles shown on the previous page.

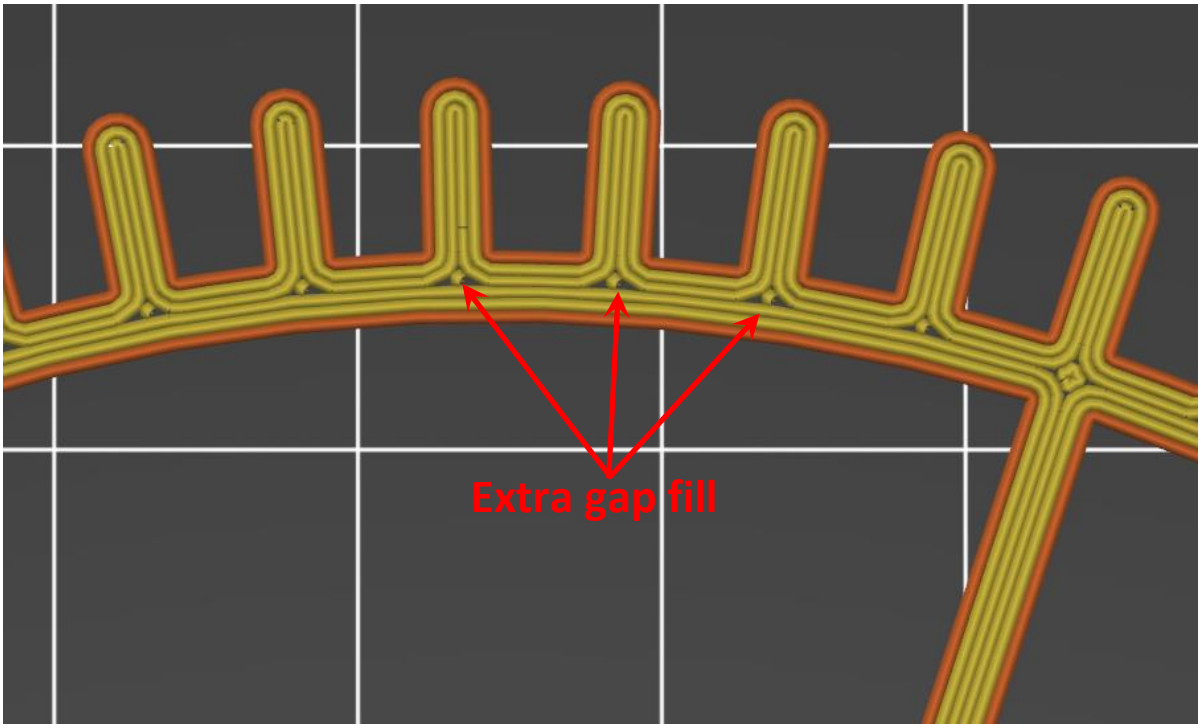


Figure 15: Arachne side effects for gears

PrusaSlicer has the ability to select between Arachne or the classic perimeter generator. The gears will be much cleaner with the classic perimeter generator and five perimeters. Unfortunately, the latest version of Cura seems to have switched exclusively to using Arachne for perimeter generation. PrusaSlicer is preferred for slicing the gears. If you use Cura, then a version before 5.0 will produce better results for the gears. The non-gear parts can use any slicer.

Layer Changes

The front frame needs a color change at 12.20mm to add highlights for the numbers. A color change at 10.40mm to change the dial to a light color is optional depending on the base color.

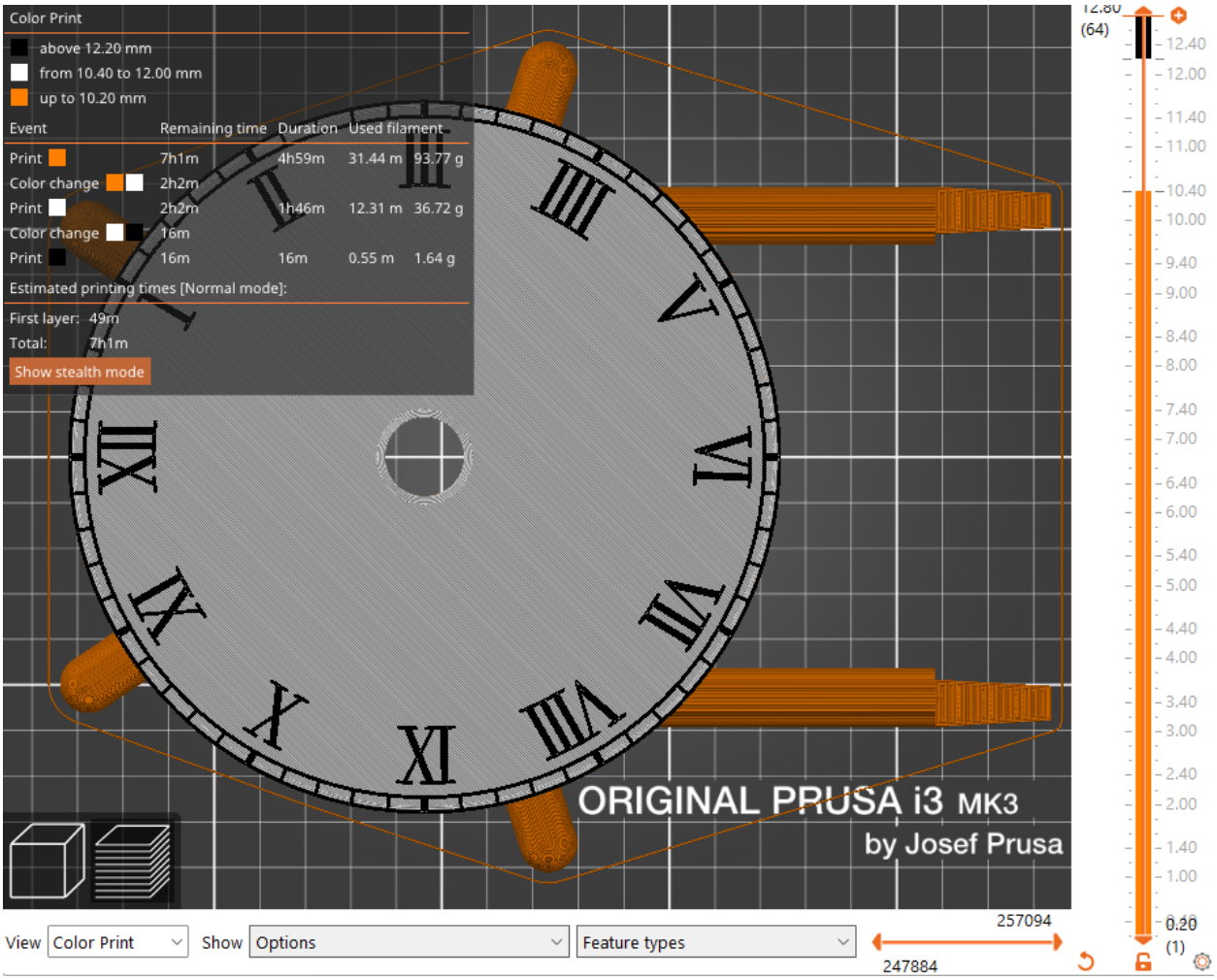


Figure 16: Front frame layer changes

The back frame has integrated standoff columns to position the gears at the proper heights. The clock looks best with a color change at 10.40mm to match the gear color.

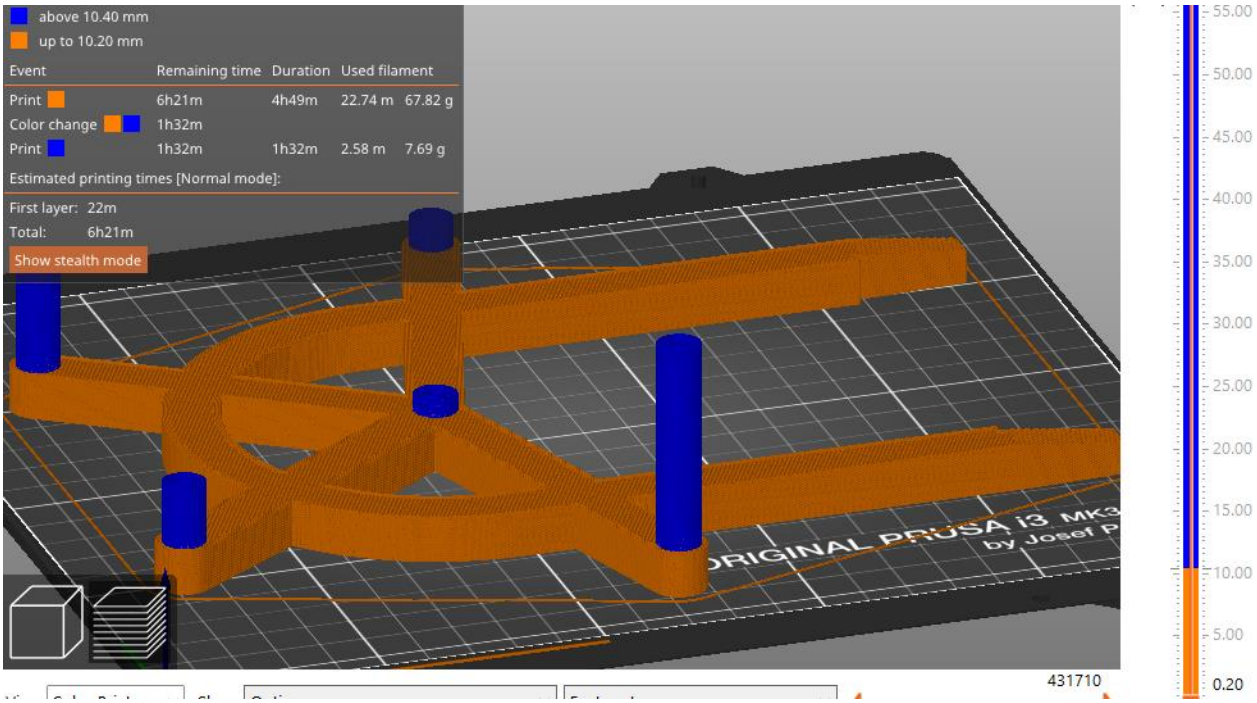


Figure 17: Optional back frame layer changes

Here is a diagram showing the various gears used in the clock.

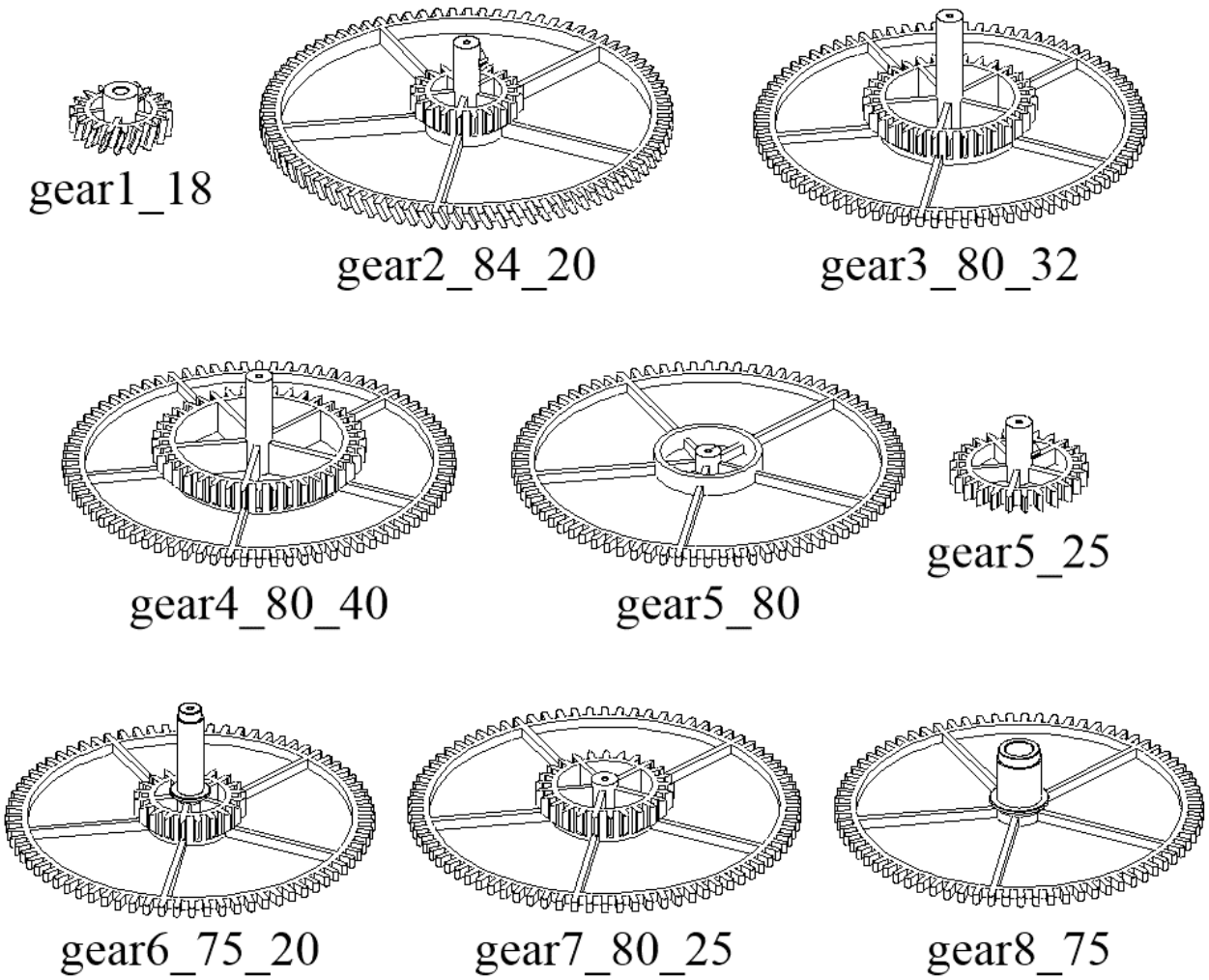


Figure 18: Gear reference diagram

Building the Clock

Start by printing all the parts as described earlier. A few components can be pre-assembled before being placed into the clock.

Component Pre-Assembly

Gear 2 is the second hand gear driven once per minute by the motor pinion. The arbor needs to be tight on the gear so the second hand will rotate when gear 2 rotates. It doesn't need to be very tight, just enough to hold the position. The shaft should extend 0.4' (10mm) below the bottom of the gear. The ideal screw size is M3x8mm, but M3x6mm or M3x10mm screws can also be used.

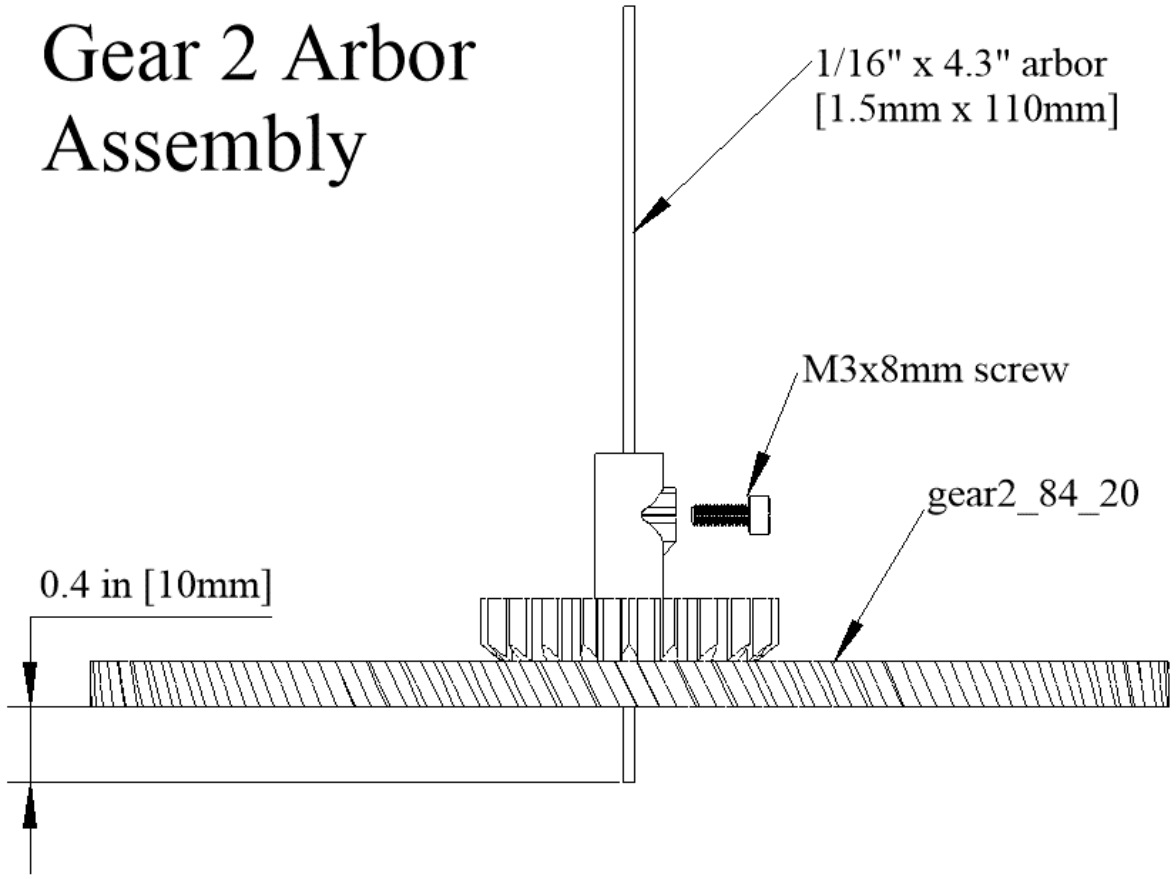


Figure 19: Gear 2 shaft assembly

The gear 5 assembly incorporates a friction clutch to hold when the clock is running and slip when setting the time. Gear5_25 and the shaft collar are tight on the 3.5” arbor and gear5_80 is allowed to rotate. The pen spring provides a slight amount of pressure to hold gear5_80 steady when the clock is running.

Add the 1/16” (1.6mm) shaft collar to the arbor with 1” (25mm) extended out the bottom end. There is an optional printed version of the shaft collar that can be used to avoid having to buy a single shaft collar. The printed gear5_collar is a press fit it onto the shaft. If the printed shaft collar is too tight, drill it by turning a drill bit by hand. Make a single pass most of the way through. The arbor can then be pressed into the printed collar and it should hold tight enough. A small washer could be added to prevent the pen spring from digging into the printed part.

Add a pen spring, gear5_80, and gear5_25 spring to the arbor. Position gear5_25 with 0.3” (8mm) extended. Secure it with an M3x8mm screw. The total thickness of the stack should be around 2.2” (58mm). Gear5_80 should be able to rotate independently from gear5_25.

Here is the completed assembly.

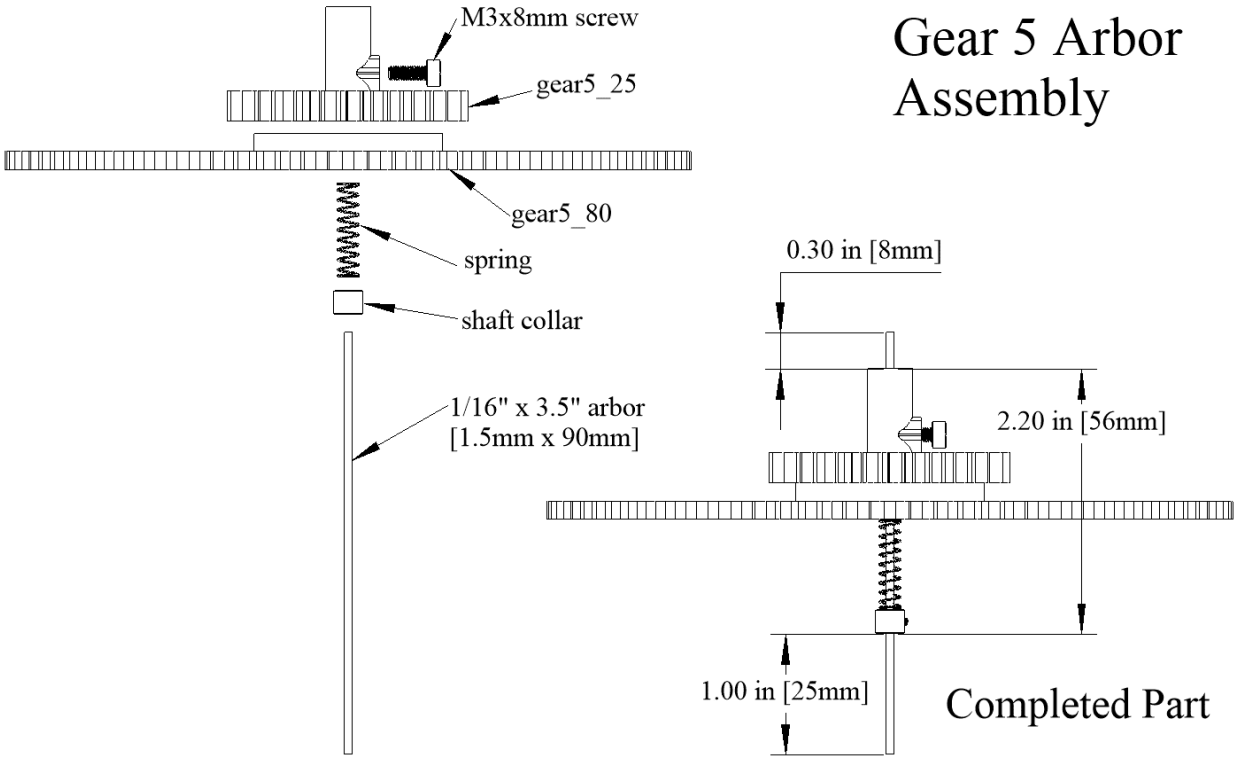


Figure 20: Gear 5 shaft assembly

The front dial is the largest component in the clock. Print frame_dial_roman as a single piece if your printer has a 220x185mm or larger print area. Smaller printers with a 180x180mm print area can use the split dial option and assemble the components using alignment rods and screws. The default orientation should fit, or you may need to rotate parts slightly. You may also need to adjust the skirt distance or eliminate the skirt for some parts.

The lower legs of the split dial option will be aligned using four 1/16” by 3” (1.5mm by 75mm) music wire rods and attached using two #6x3/4” (M3x20mm or M3.5x20mm) wood screws. The alignment rods are expected to be a tight fit that may need pre-drilling. The front of the legs have beveled edges as shown in the diagram.

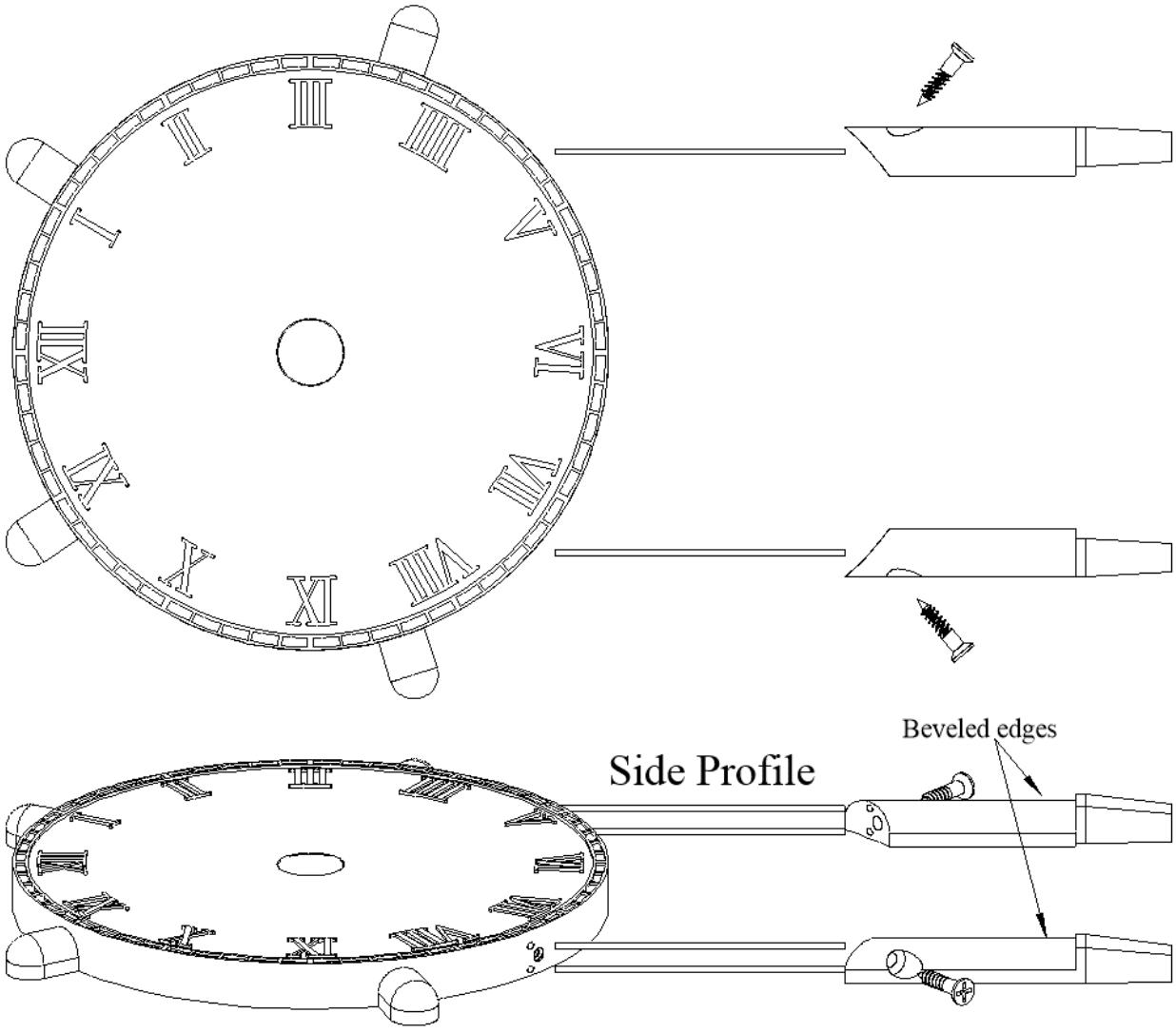


Figure 21: Segmented front frame assembly

The back frame is the same size as the front dial. The single piece component needs a 280x185mm print area and the segmented print only needs a 180x160mm printer. The single piece print will be slightly stronger and skips the extra assembly steps.

The segmented back frame uses two 1/16” by 3” (1.5mm by 75mm) alignment rods and a #6x3/4” (M3x20mm or M3.5x20mm) wood screw. The alignment rods are expected to be a tight fit that may need pre-drilling.

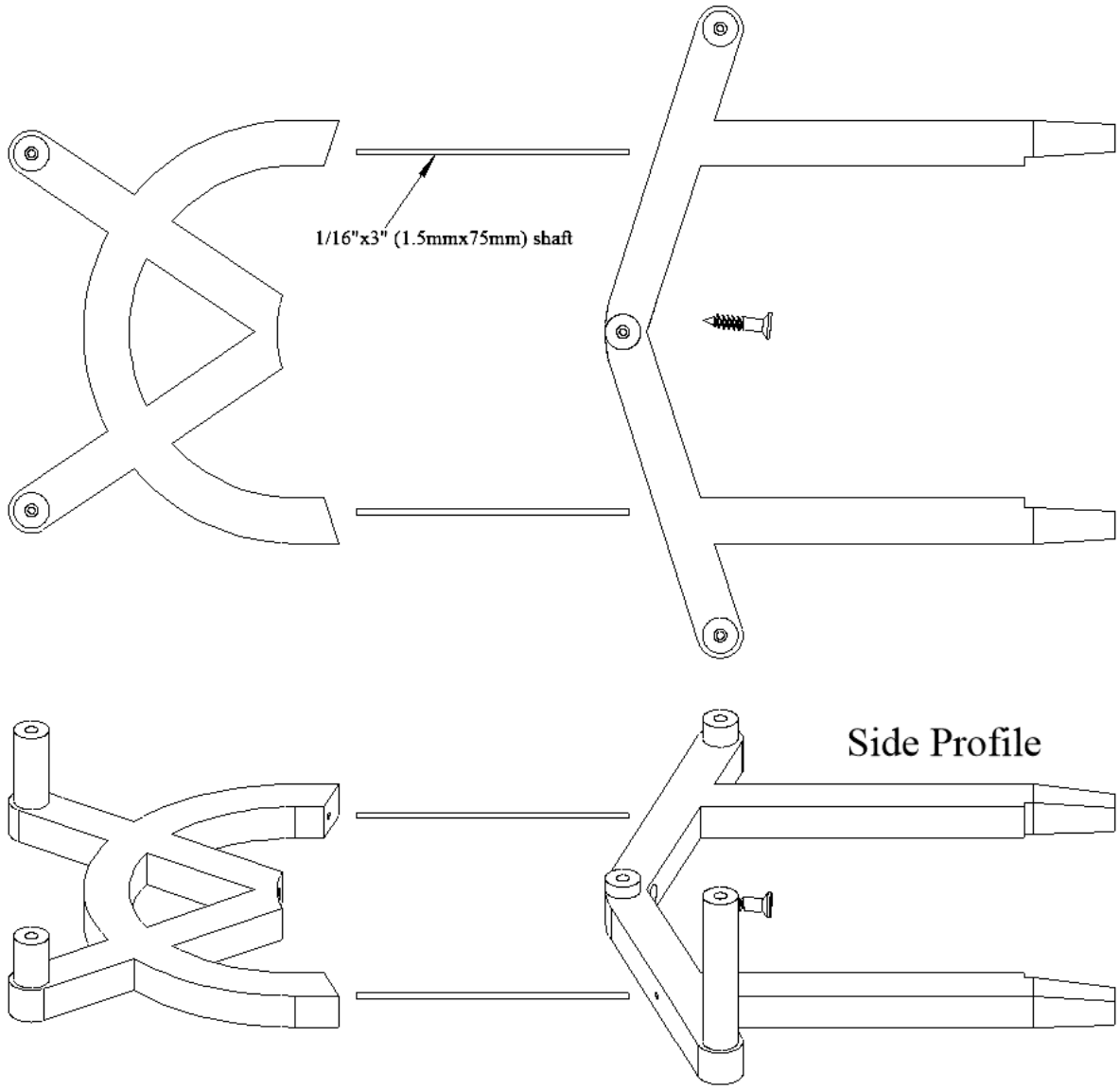


Figure 22: Segmented back frame assembly

Checking Component Fit

It is a good idea to check the fit of the remaining components before assembling the clock.

1) Check that arbors fit in their respective holes. The 1/16” or 1.5mm arbors need to fit into the holes in the frame. The arbors should also pass through the gears. Drill them out if needed. It is desirable for the holes to be just large enough for the arbor to pass through and spin freely. Don’t enlarge the holes too much. The back frame has several standoffs that position gears at the proper depths. The tall portion of each standoff is a loose fit and the arbor gets accurately positioned at the bottom of each hole. Use a long drill bit if needed to provide clearance at the bottom of each standoff. See the following picture for details.

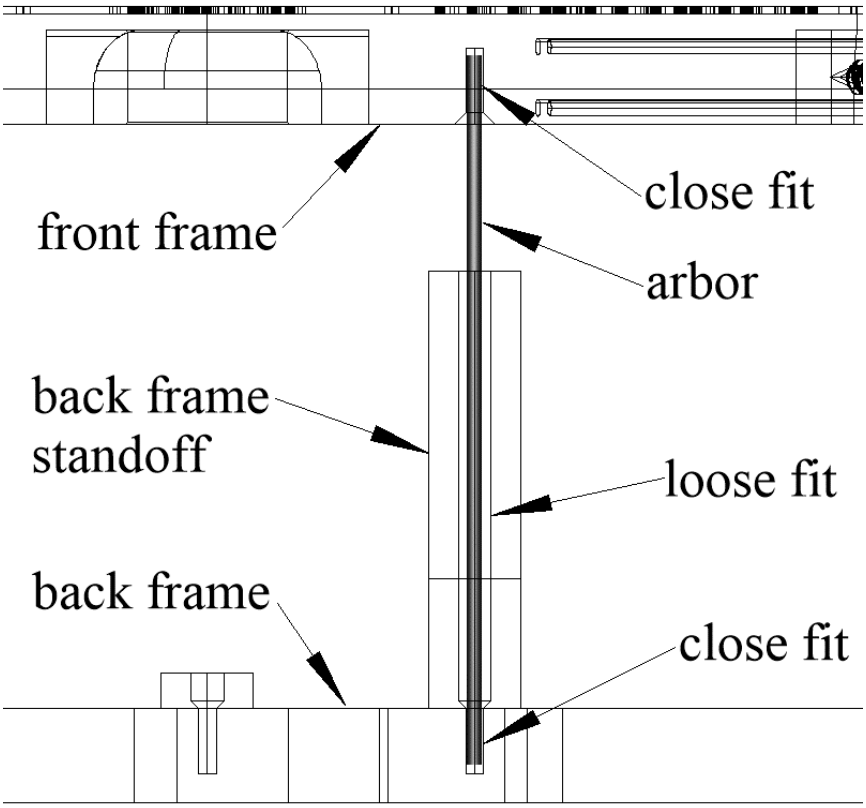


Figure 23: Arbor detail and back frame standoffs

2) Check that the minute hand (gear6_75_20) fits into the hour hand (gear8_75) gear without binding. Also check that the hour hand gear fits into the front frame. Gently enlarge the holes using rolled up sandpaper or a round file if needed. Or wrap sandpaper around the outer shafts to reduce the diameters slightly.
3) Check that the completed Arduino and motor controller fits into the base. Check that the USB port passed into the base and into the Arduino Nano USB port.
4) Check that the frame fits into the base. The pegs are tapered so they should go in easily at the start and get tighter when they bottom out. Sand the pegs slightly if needed. The bottom screws should pull everything together. Use the screws to help push the frame out if you need to take the clock apart.

Adding the Gears

The gears are added to the clock sequentially, starting with the gear 2 arbor assembly and working numerically up to gear 8. Most arbors are 3” (75mm) long except the 4.3” (110mm) gear 2 arbor that passes through the front of the clock and the 3.5” (90mm) gear 5 arbor that passes through the back.

Start by adding the previously completed gear 2 assembly into the center hole of the back frame.

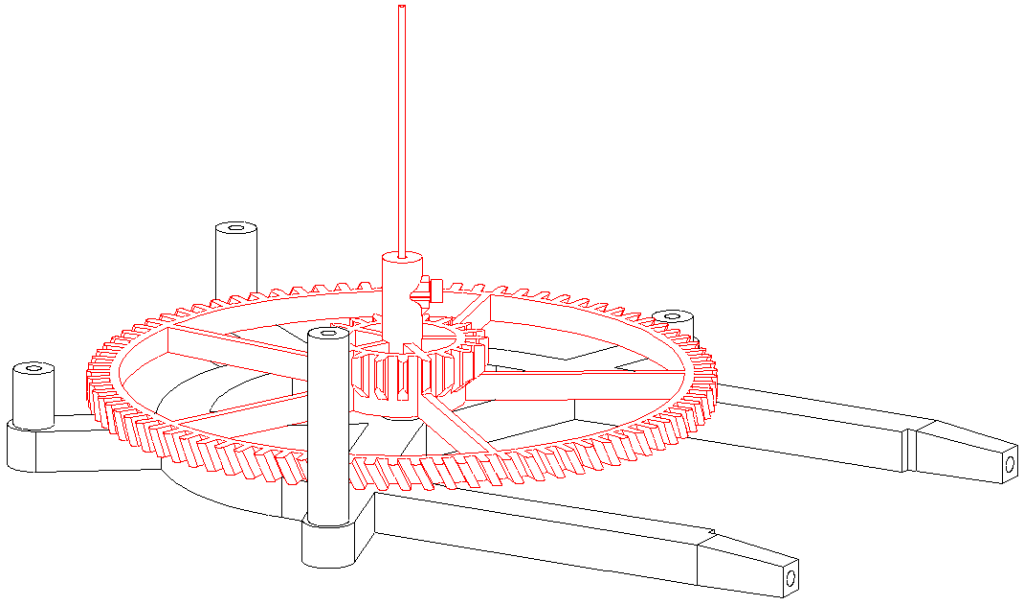


Figure 24: Start with the gear 2 assembly

Add a 1/16” by 3” arbor and gear3_80_32 into the upper left frame position. It should mesh with gear 2.

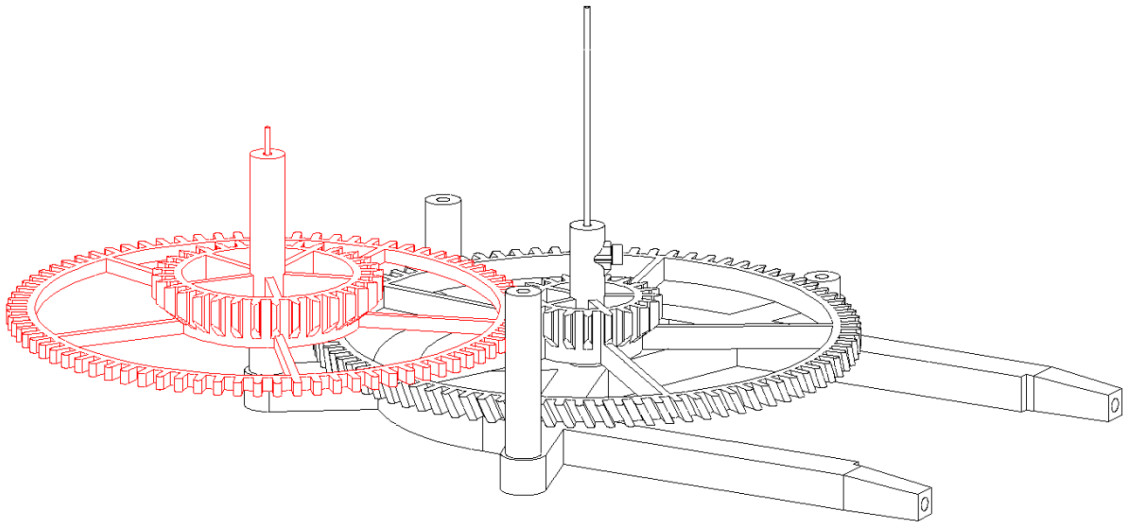


Figure 25: Add gear 3

Add a 1/16” by 3” arbor and gear4_80_40 into the upper right position. It meshes with gear 3.

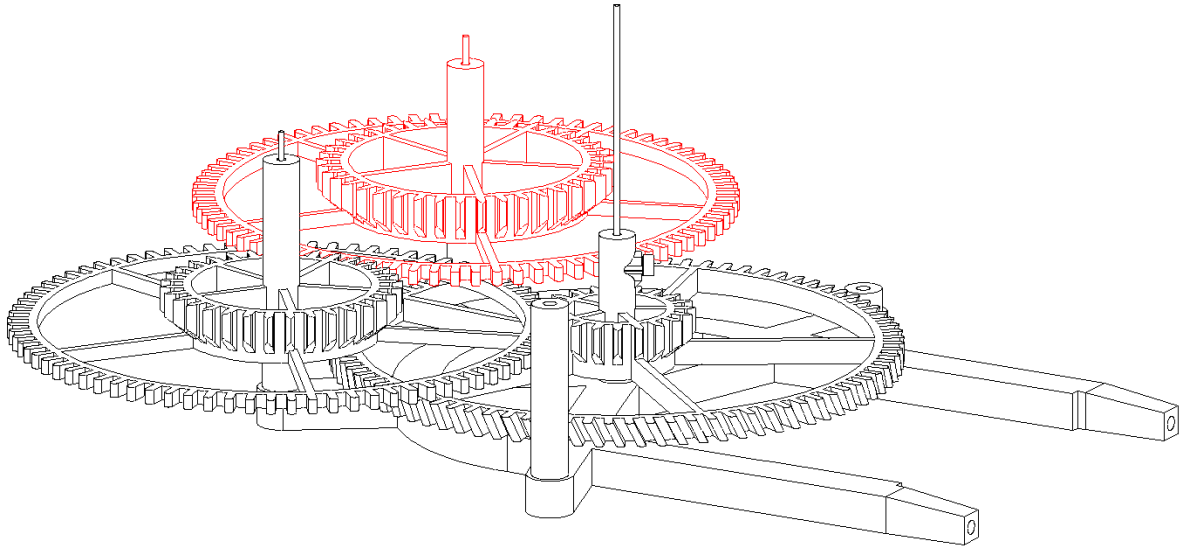


Figure 26: Add gear 4

Add the previously completed gear 5 assembly in the lower right position. The long end of the arbor sticks through the back of the frame.

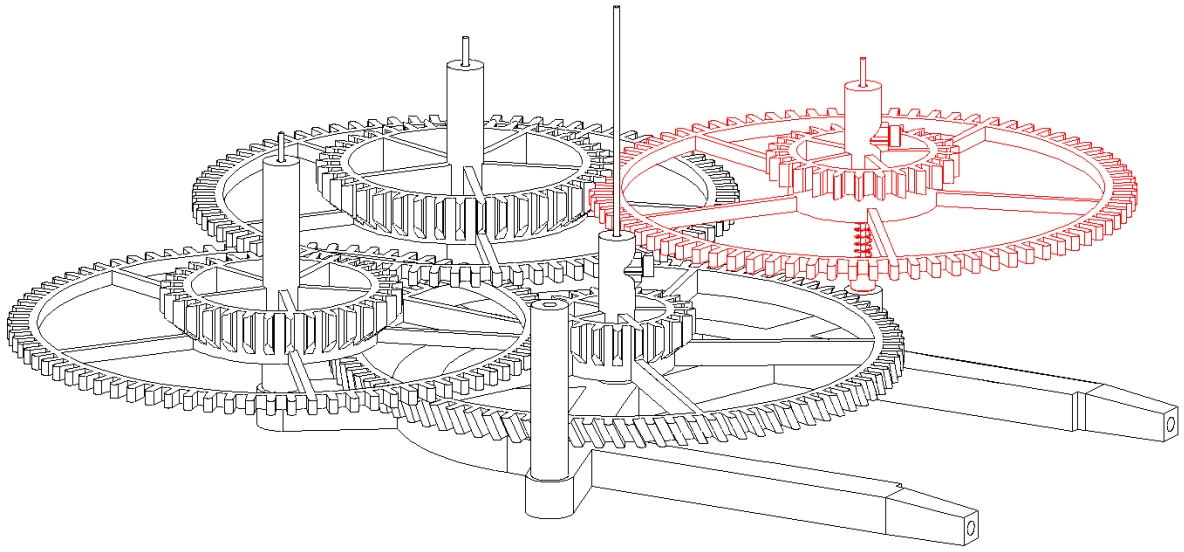


Figure 27: Add gear 5 assembly

Add gear5_knob to the arbor using a M3x8mm screw where the arbor sticks through the back of the frame. Gear5_25 should rotate with the knob when holding gear5_80.

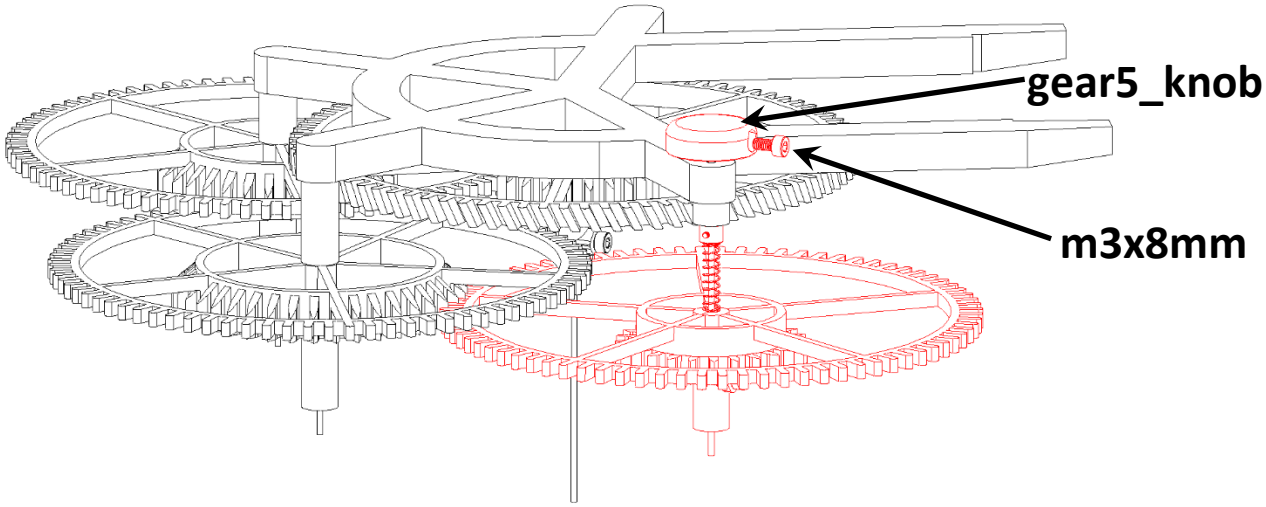


Figure 28: Knob added to gear 5 behind the clock

The gear 5 knob is now projecting behind the clock so it will not sit flat anymore. Set the back frame onto an empty filament spool for the remaining assembly steps.

Add gear6_75_20 to the gear 2 arbor in the center of the clock. This gear drives minute hand and will pass through the front frame. It meshes with gear 5.

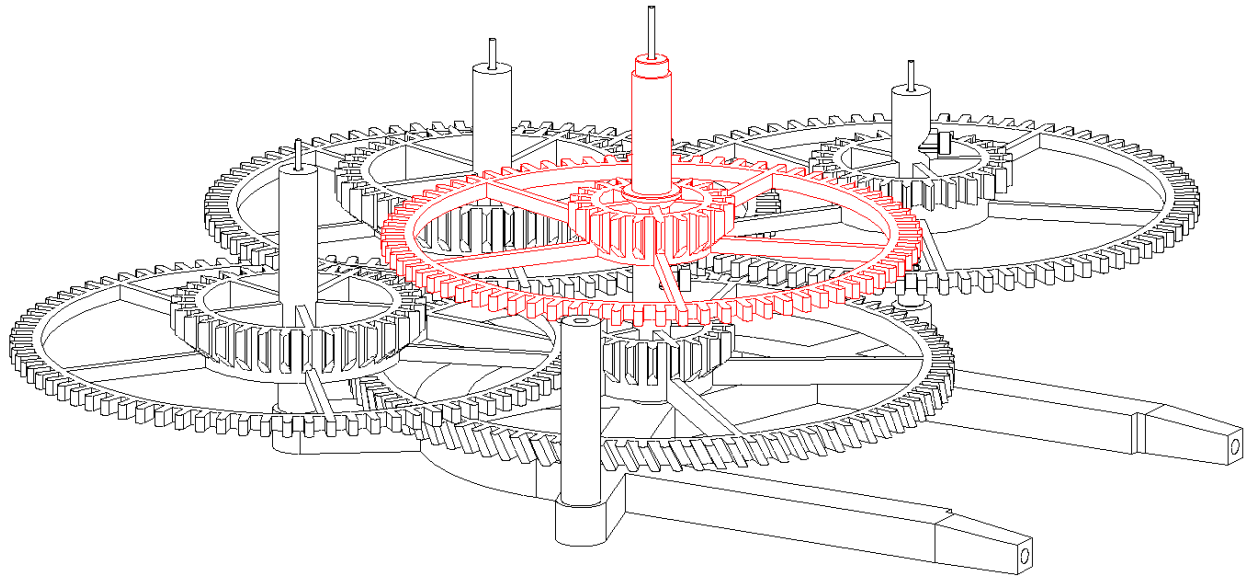


Figure 29: Add gear 6

Add a 1/16” by 3” arbor and gear7_80_25 in the lower left position. It meshes with gear 6.

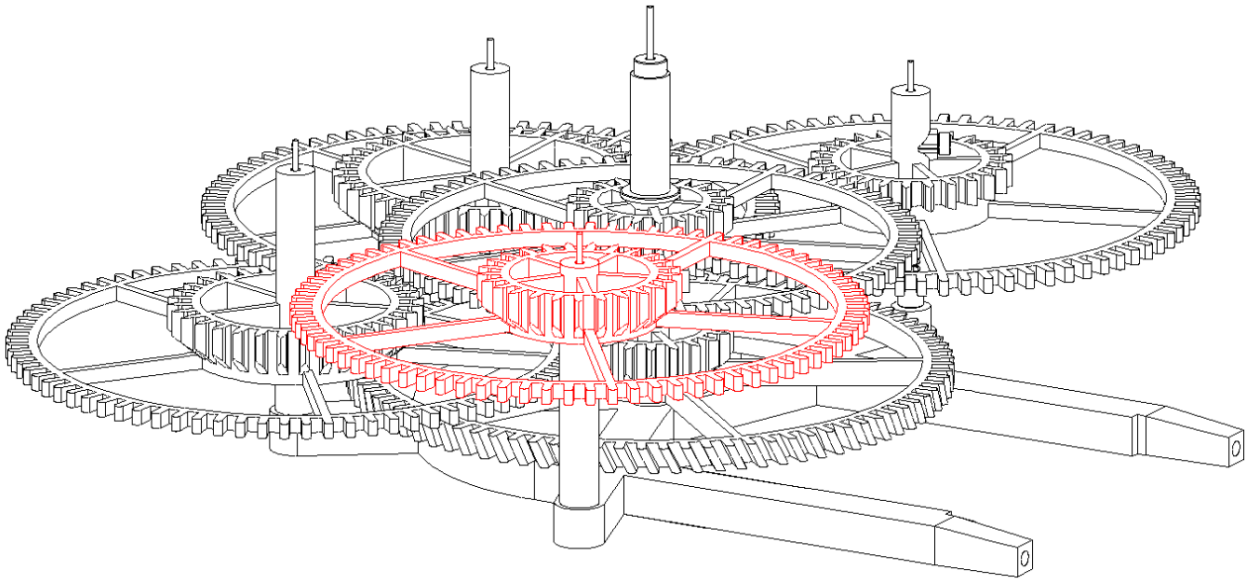


Figure 30: Add gear 7

The last remaining large gear is gear8_75 that goes on the gear 2 arbor in the center of the clock. It meshes with gear 7 and passes through the front frame to hold the hour hand.

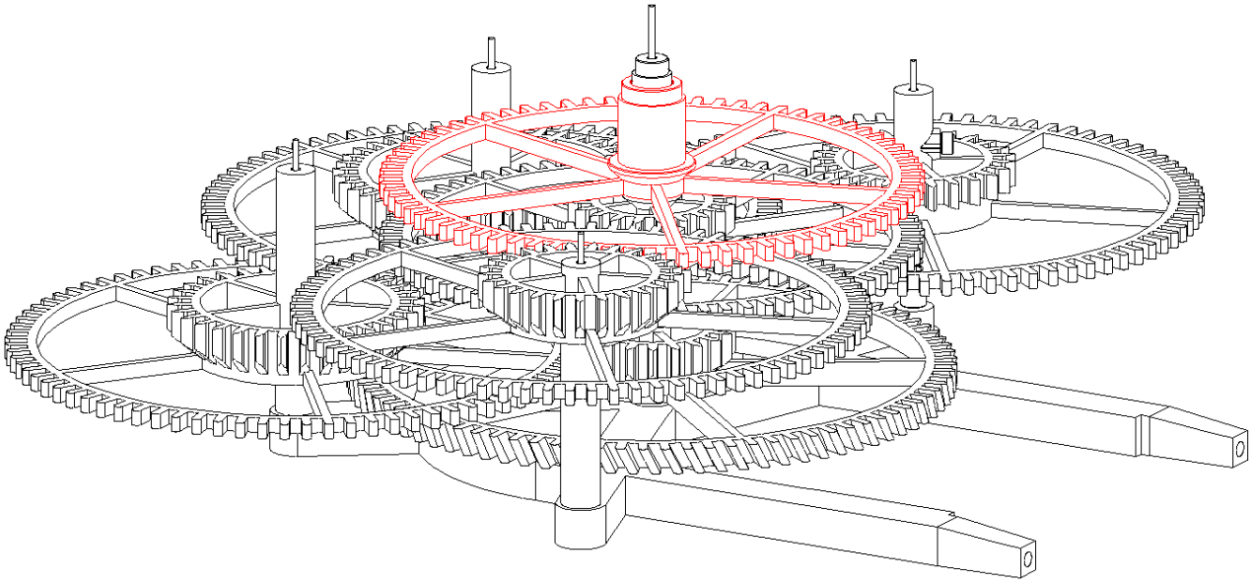


Figure 31: Add gear 8

Add the front frame onto the clock. Start by placing the dial over the central arbor. Align the remaining arbors with their respective holes. This clock should be easy to assemble since the holes for the arbors are easily visible around the dial and the holes are slightly chamfered. The frame will drop into place when all the arbors are positioned properly. The hands can be added now or later.

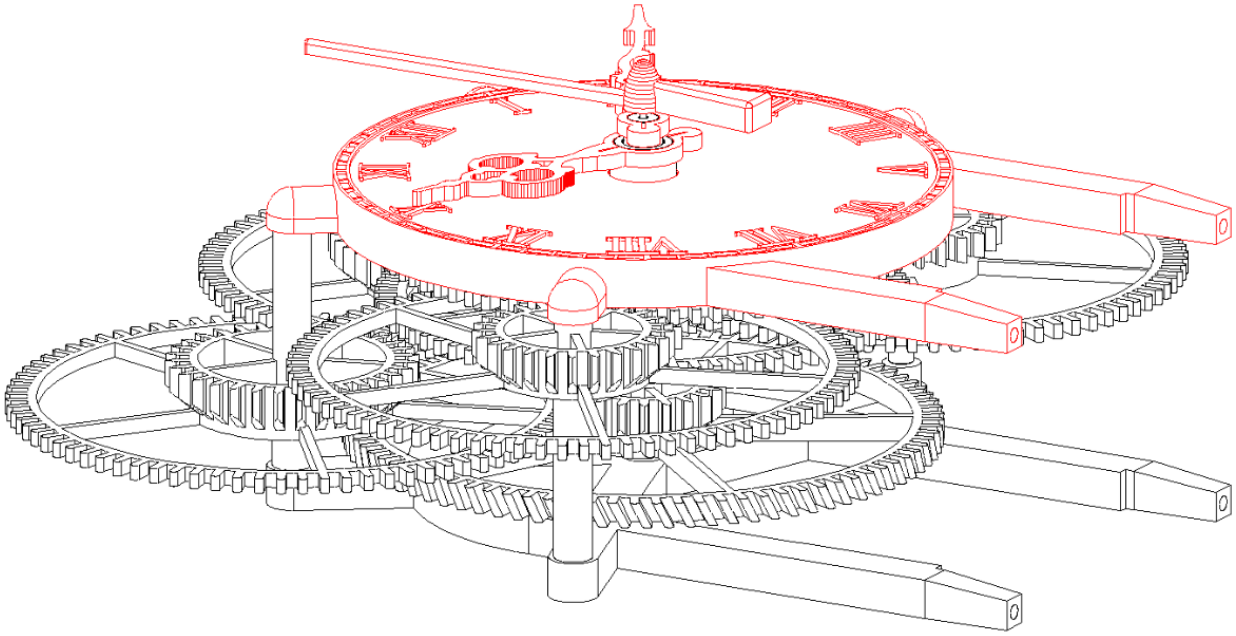


Figure 32: Add the front frame

Motor Tuning

The final assembly step is to add the motor and base onto the clock. It is easier to adjust the controller if you run the stepper motor cable through the USB port so the CNC Shield V4 sits outside the base. It can be placed back into the base after debug has completed.

Print the 33mm or 39mm version of the motor mount that matches closest to your motor size. Enlarge the bottom opening using a chisel or Dremel tool if needed so the wires will be able to pass through later.

Attach the motor to the motor mount using two M3x8mm screws. Another M3x8mm screw holds gear1_12 onto the motor shaft. The top of gear 1 should be positioned 0.78” (20mm) above the top of the motor. It is OK for the stepper motor shaft to extend through gear 1.

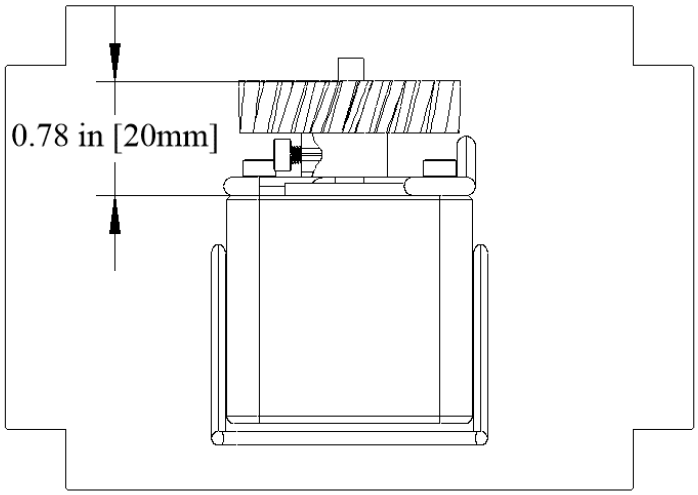


Figure 33 Motor and motor mount

Print the style of base that matches your Arduino Nano port (mini-USB or USB-C) and the desired power plug location (back, left, or right). Add the top portion of the clock into the base. The screws to hold the base are optional at this point since you will need to take it apart to add the CNC Shield V4 into the base in a later step.

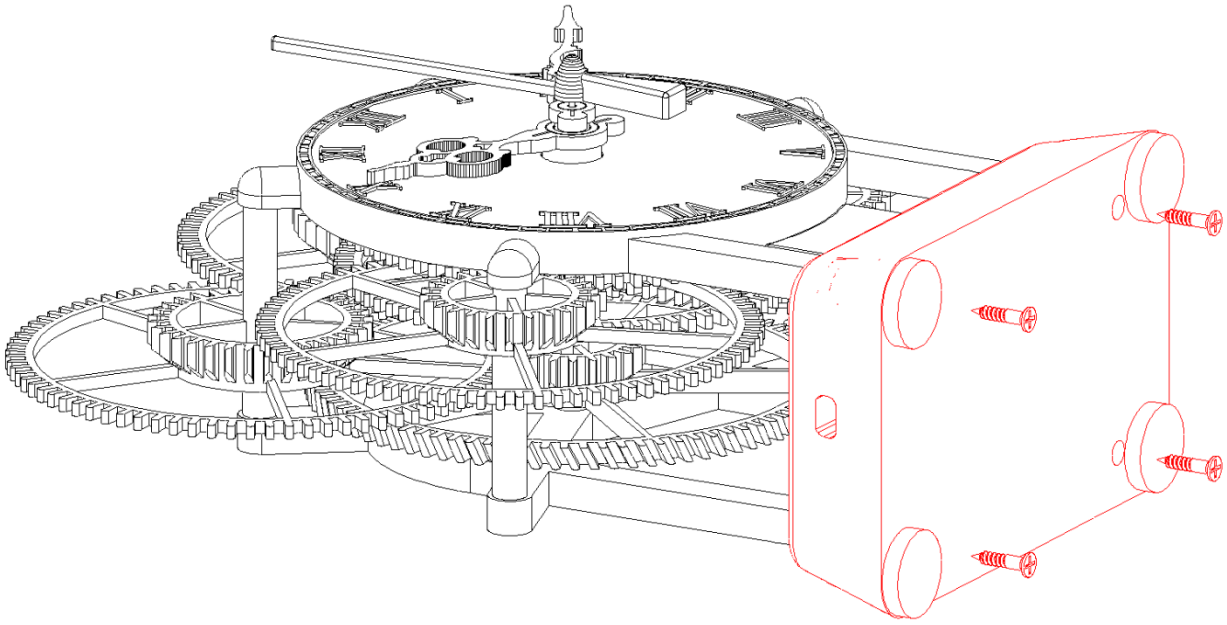


Figure 34: Add the base

Stand the clock upright and connect the previously edited and programmed CNC Shield V4 to the stepper motor. Plug in the USB cable and the clock should start moving.

The jumper labeled “CoolEN” can be added or removed to change the motor direction. The next three jumpers should be set to 100 (“in”, “out”, and “out”) for the gear ratios used in this clock.

The next step is to adjust the stepper motor current to be as low as possible for the clock to operate. The clock only requires a tiny amount of energy. Start with the TMC2208 Vref potentiometer rotated all the way to the left. If the clock runs, leave it at the low setting. Or your clock may need a tiny bit of additional current. Turn the potentiometer a fraction of a turn until the clock operates properly.

It should only require a small amount of force to stall the second hand gear. The stepper motor will jump between positions with minimal pressure on the second hand gear. This can be useful to set the second hand to an exact position, since this clock has an accuracy of about a second per week. The lowest possibly motor power makes this easier.

Final Assembly

After the driver is tested, the clock is ready for final assembly. It is a good idea to unplug the USB cable before disconnecting the stepper motor cable. Remove the base from the clock and move all the components into the base.

Insert the CNC Shield V4 module into the base so the USB plug aligns with the cable opening. Check that the USB cable can fit through the hole and into the Arduino Nano. Add the appropriate holddown (left, right, or back) to keep the CNC Shield V4 held in place. Coil up the stepper motor cable so the motor mount can be added without pinching anything. Pay attention to the direction that the stepper motor cable plugs into the TMC2208 header, since it is not keyed. Reversing the cable may reverse the motor direction and the direction jumper is not easily accessible after the clock is assembled.

Here is what the base and motor assembly would look like with the power on the left side. The first debug test can have the stepper motor cable passing through the USB power hole to access the CNC Shield V4 outside the base. The final assembly will have the CNC Shield V4 inside the base and secured using the holddown. Felt pads up to 1” or 25mm diameter can be added to the bottom of the base if desired.

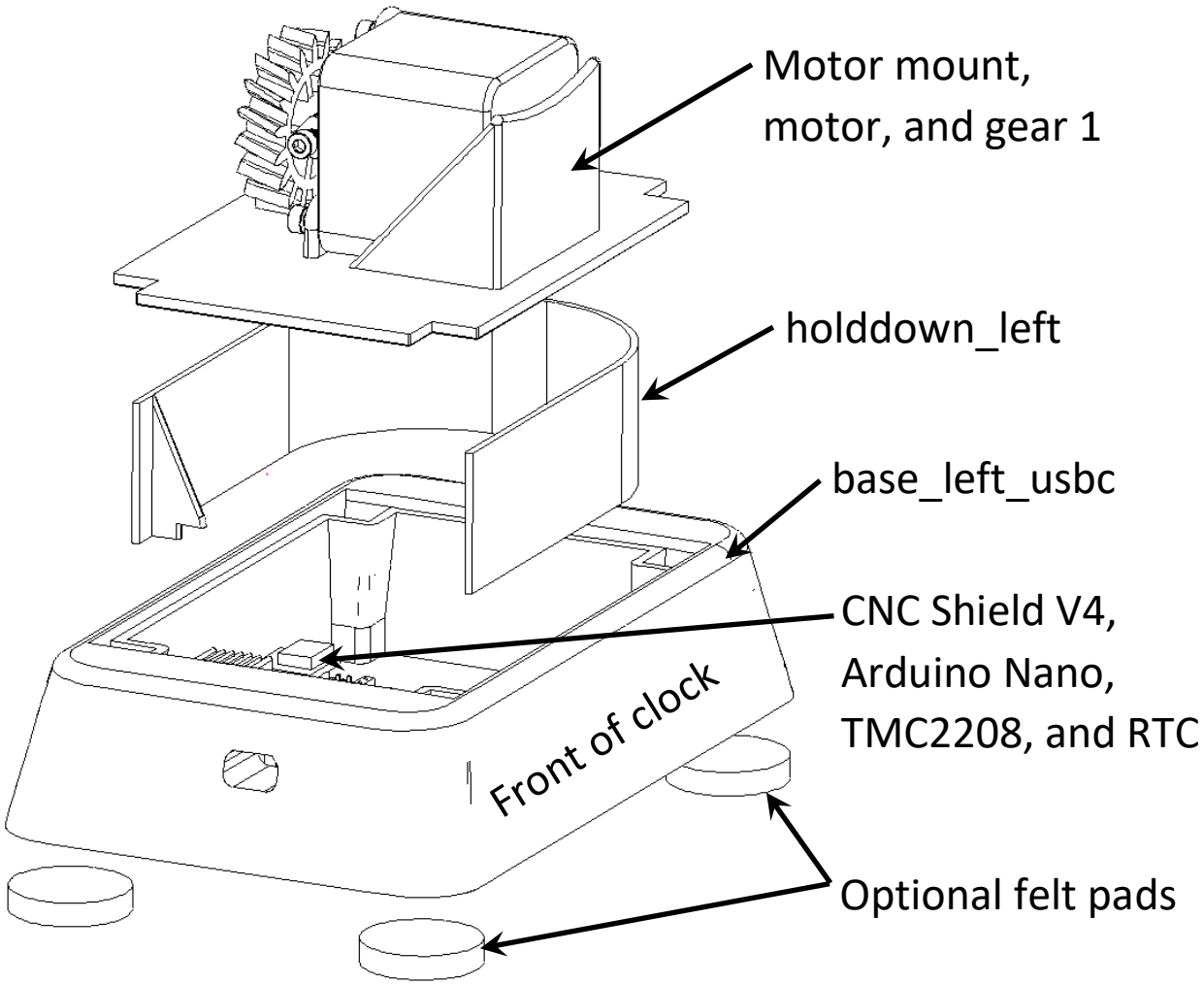


Figure 35: Motor and electronics added to base

Debug

There are a few things to observe after the clock is re-assembled with the electronics in the base.

1) Is the motor rotating before it is placed into the clock? This should be the very first test. Observe the debug monitor window when you are programming the Arduino Nano.
2) Does the motor rotate by itself, but not when assembled into the complete clock? This may be caused by binding within the gear train. The clock only requires a tiny amount of power. If the second hand moves, all other gears should also move. Everything should spin freely. Check for friction where gear 6 passes through gear 8 and where gear 8 passed through the front dial. Also check for too little pressure on the friction clutch. Stretch the spring slightly if needed. You can also increase the motor current by rotating the Vref potentiometer.
3) The debug monitor can be checked for a mixture of "+" and "-" if the clock is tracking against the RTC. You can also watch the status LED. It should be blinking every few seconds. It will be on when a "+" is displayed on the monitor. The status LED is easiest to see when the electronics are outside the clock.
4) A missing or dead RTC will stall the algorithm. The debug monitor will show the first header line and stop the first time it tries to read the RTC.
5) Does the clock keep accurate time? The clock should track within a small fraction of a second compared to the real time clock if it is operating properly.

Final Notes

I hope you enjoy building this clock as much as I enjoyed designing it. Many features make it an extremely functional clock. The new drive electronics make the clock completely silent with incredible accuracy. The solid dial makes it easy to read the time, but I still wanted to see the gears so they were extended well beyond the frame. This has become my new favorite clock.

Feel free to message me with questions during the build or just to say hi. You can reach me at MyMiniFactory, YouTube, or the forum on my web site at https://www.stevesclocks.com/forum
My Patreon page is https://www.patreon.com/user/about?u=30981480 if you want to support me.

Happy clock building,
Steve

Appendix: Some of my Other Clock Designs

Here are a few of the other clocks I have built. Many of them will eventually be released for others to build. The first is a grasshopper escapement to replace the deadbeat escapement in my one of my earlier clocks. It needs a bit of fine tuning before it can be released. The second image is a rendering of one of my designs as it may look after porting to use wooden gears.

Figure 36: Grasshopper clock and wood clock rendering

These are some sample wooden gears cut from solid wood using a new method to prevent expansion from humidity changes. They will eventually be used to create the rendered clock on the previous page.

Figure 37: Wooden gear experiments

Here are some of the prototypes before settling on the final design in this release. The clock on the left was a proof of concept. The middle clock was released in 2021 ago as SP6. This design is similar to the one on the right. A future wooden gear version may be slightly larger.

Figure 38: Desk clocks

Here is the clock that started it all. It is posted to https://www.thingiverse.com/thing:3524448

Figure 39: Original Thingiverse design

This is my second clock posted to https://www.myminifactory.com/object/3d-print-137009

Figure 40: Large pendulum clock

A clock posted to https://www.myminifactory.com/object/3d-print-32-day-clock-easy-build-156759 with a runtime up to 32 days between winding. It is one of my easiest to build. Some of the features making it easy to build also make it more efficient so the runtime can be increased to 32 days.

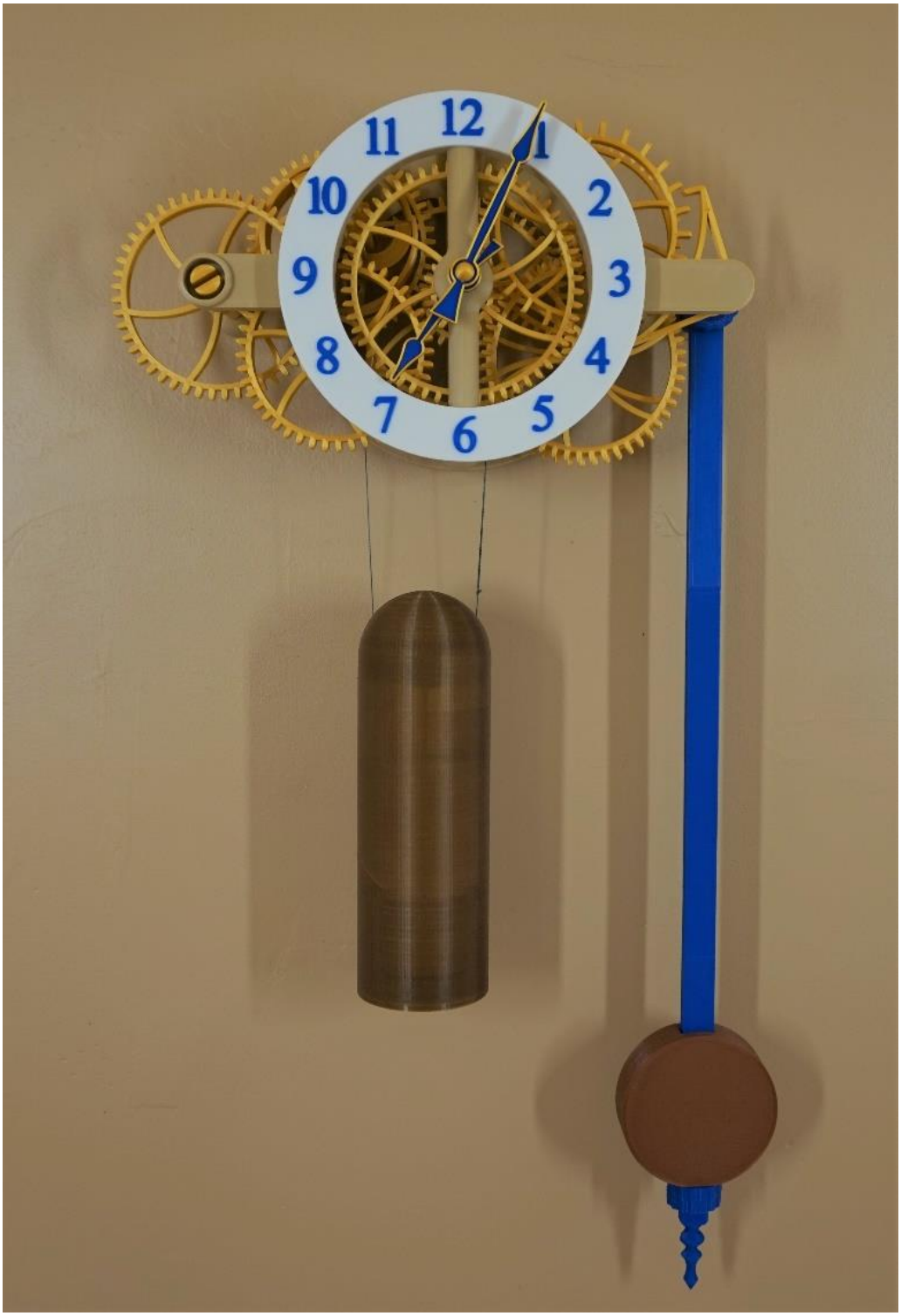


Figure 41: Easy build clock with 32 day runtime