CRAZY GEAR DESK CLOCKS

SP10 and SP11 Assembly Notes

Instructions for building a 3D printed desk clock
with large gears and extra dynamic motion
Two different size clocks are available

Steve Peterson
07-May-2023

Contents

Tables

Figures

Revision History

18-Apr-23	Original version
22-Apr-23	Added gear placement figures on pages 51 and 52
07-May-23	Fixed typo on page 54 (gear7b_15 and gear0b_25_10)

Description

These are my newest and most dynamic clocks. The design has evolved over several years from a tiny noisy clock into a large elegant smooth flowing clock. The gears are a modified involute shape that I call "crazy gears" that are continuously in motion. The motor control circuit is ultra-quiet and maintains a typical accuracy of about a second per month. Lots of things came together to make a clock that grabs your attention and looks great on any desk.

The clock is available in two sizes, a large size at 14.75"x15" (375x380mm), and a medium size at 11.75"x12.75" (300x325mm). The gears extend past the dial for great visibility of the moving gears, especially in the large clock. The medium clock is an 80% scale with a similar size dial. Both designs are similar enough that they share the assembly guide.

There are several options to customize the look of your clock. The dial can have Roman numerals or simple numbers and can be printed with a solid background or opened up for better visibility of the moving gears. The clocks on the cover are printed in dual color Quantum PLA from MatterHackers that changes color as the gears rotate.

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) Uses readily available 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.

I consider this to be my best looking clock. The precision real time clock reference holds incredible accuracy. The TMC2208 stepper motor driver provides smooth silent motion. The components are readily available at reasonable prices. A more elegant motor control circuit is available on my Etsy shop. I may even have a few pre-built control modules that are ready to plug in and run.

Both clocks can be printed on a Prusa mini or similar machine with at least a 175x175mm print area. A Prusa MK3S (250x210mm) or Ender3 (220x220mm) size printer works best for the large clock, but is not required. The total print time for the large clock is about 80 hours on a Prusa MK3S using about 0.9kg of filament. The medium clock takes 74 hours using 0.8kg of filament.

Design Evolution

The very first prototype was designed about 4 years ago. It was a small design with great symmetry and noisy gears. A stepper motor was driven directly by a 5VAC transformer using a capacitor to delay the signal to the second coil. The motor jumped 120 times per second resulting in a loud ticking sound. The design was placed on a shelf while I went in search of a better drive circuit.

I tested synchronous motors that are often sold as microwave turntable motors. They are commonly available for low prices. One issue is their rotational direction is random and a clock obviously needs a consistent direction. I added a circuit to force the direction, then noticed that the exact speed was hard to identify. The internal gearing uses prime numbers of teeth such as a 17 tooth pinion driving a 43 tooth gear. This helps equalize gear wear, but it was difficult to get the minute hand to rotate exactly once per hour. Another issue is the variation between brands. Motors labeled as 5-6RPM were expected to rotate at 6RPM on 60Hz AC power, but one turned at 5.824RPM and the other turned at 6.228RPM. If there are variations within the first two motors tested, it would be difficult to design a consistent clock around these motors.

Figure 1: Synchronous motors

Figure 2: 28BJY-48 motors

The next test used the common 28BYJ-48 stepper motors with a ULN2003 stepper motor driver. An Arduino Nano sent pulses at a fixed rate. They are listed as having 16:1 internal gearing, but are often documented as approximately 16.128:1. The microcontroller could adjust to the rate if all motors had the same gearing. Again, I found variations in a very small sample size. One motor had a 16.128:1 ratio and the second one had exactly 16:1. How many other gear ratios are out there? Another risk is that the internal gears are incredibly tiny. How long would it hold up under continual rotational loads? My very first test started ticking within a few days. It continued to run, but I question the long term reliability of these motors.

The next step used a simple non-geared stepper motor. I tried an A4988 stepper motor driver without much success. It ran better than the first clock, but still made noise, so I designed a micro-stepping driver for smoother rotation. A binary weighted resistor driver circuit with 64 step resolution was used. The currents were small enough that the Arduino Nano can drive the stepper motor directly through the series resistors. This motor control circuit was quiet enough to release as the SP6 Silent Desk Clock. It worked great in the small size, but a larger version of the same design had too much gear noise to release.

This is the custom motor control circuit used in SP6.

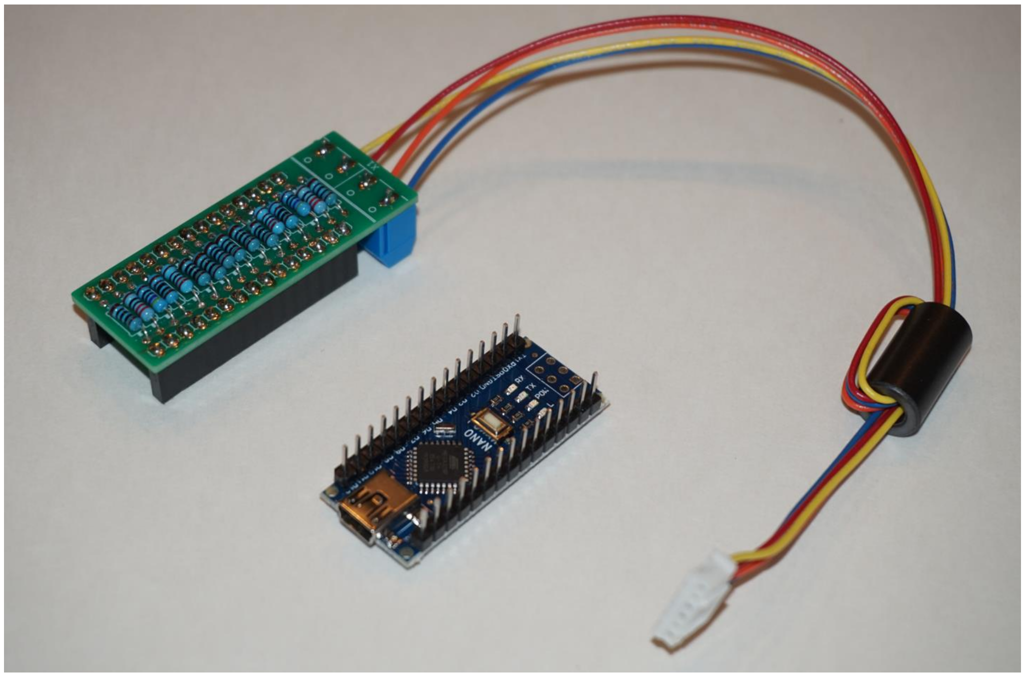


Figure 3: First successful motor controller

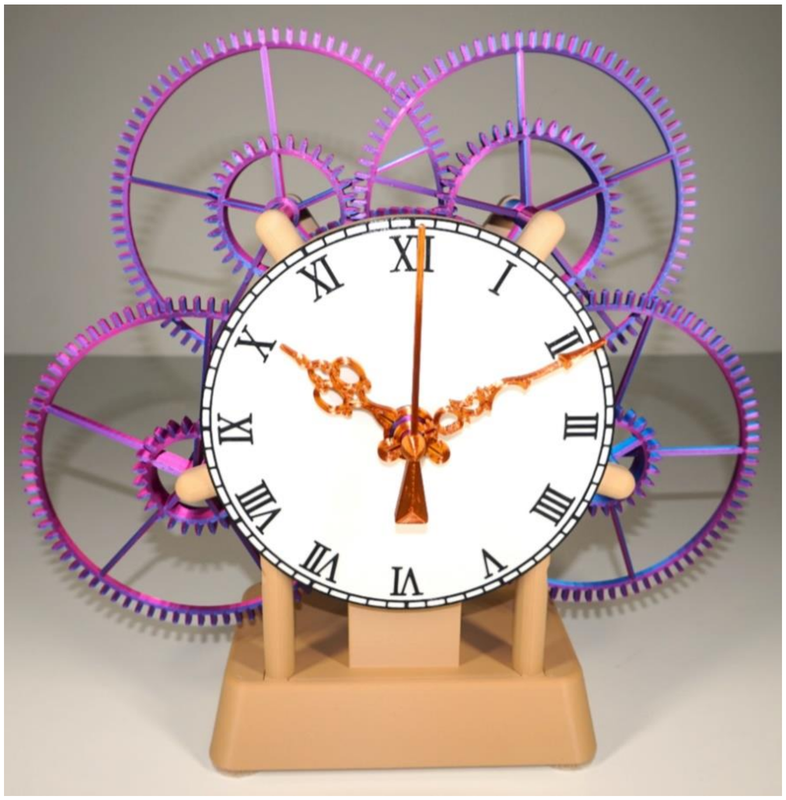


Figure 4: SP9 Large silent desk clock

I sold the small circuit board for slightly more than the cost of the parts. Shipping to regions outside the US would force additional fees that would often triple or quadruple the cost to the builder. I kept searching for a solution that would avoid the outrageous customs fees. The breakthrough came when I found a product called CNC Shield V4 that runs incredibly smooth when coupled with a TMC2208 silent stepper motor driver. This enabled a large format of the clock to be released as the SP9 Large Silent Desk Clock. A precision real time clock was added to provide an accuracy of better than a minute per year without calibration. This controller has become my new favorite motor drive circuit.

The gears in SP9 were enlarged and extended beyond the dial. The time was often hard to read with all the gears in the background, so a solid dial was added to make the hands more visible. The new motor controller is incredibly quiet and it uses readily available parts. This clock is getting closer to the ultimate design.

While designing these clocks, I kept staring at a desk clock that has been in my office for about 15 years. It is a great looking clock with continually rotating gearing. The large center gear rotates once a minute along with the gears on the side. There are redundant gears providing dynamic motion. I wanted to design something similar to this clock.

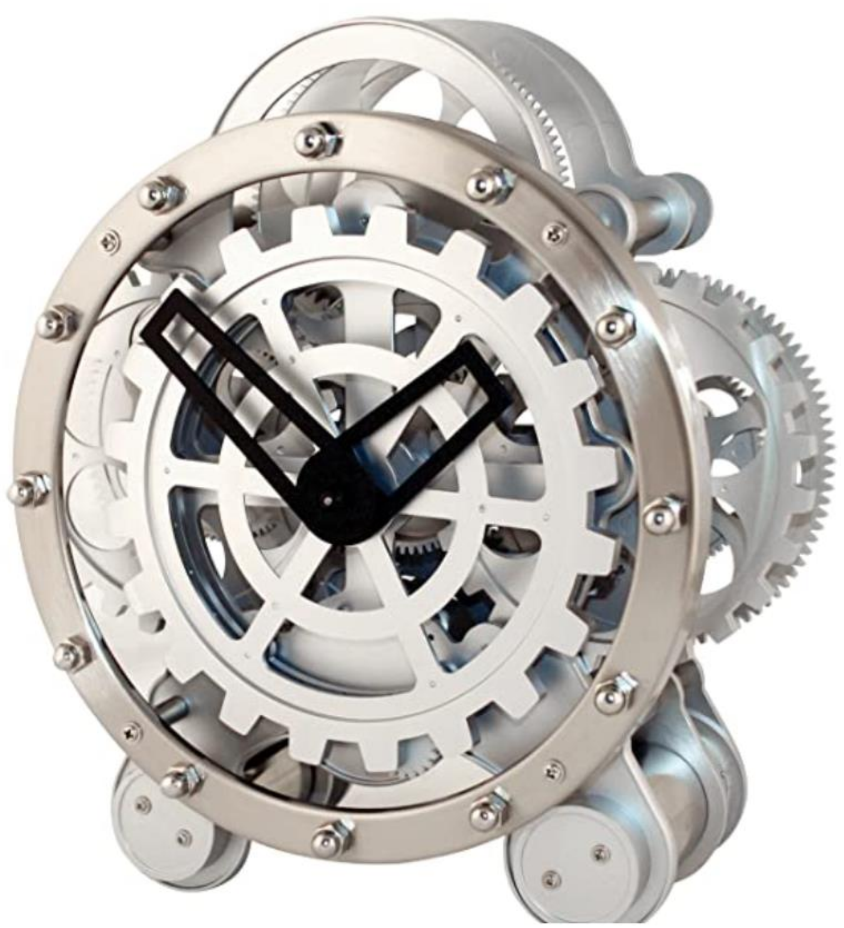


Figure 5: Inspiration for SP10 and SP11

I went back to the original small concept with six gears spaced symmetrically around the dial. Involute gears have large teeth merged with the rim, giving a unique look. The clock was designed using a diametral pitch of 5 (module 5.08). The largest 32 tooth gear has a pitch diameter of 6.4” (163mm) and an overall size of 6.75” (172mm). I call them crazy gears.

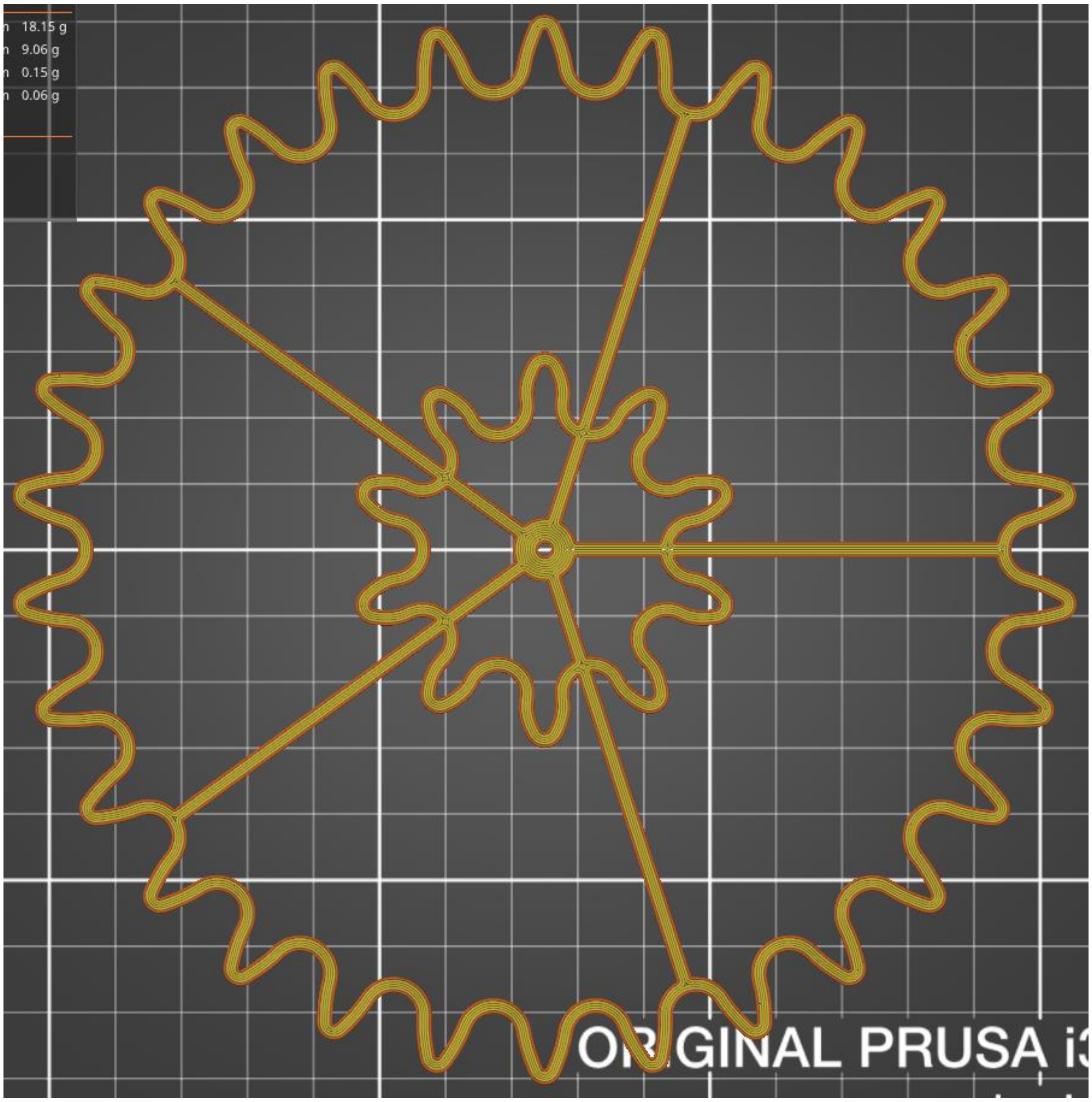


Figure 6: Crazy gear profiles

Here is a snapshot of the evolution of the desk clock series. The original prototype on the far left was too noisy to release. The remaining clocks are SP6, SP9, SP10, and SP11. Each clock has grown in size and intensity. SP9 was released with a solid dial for easy readability. I can't decide if the new clocks look best with a solid dial or an open design to really highlight the dynamic motion. They were released with both open and solid dials so you can decide which style you prefer.

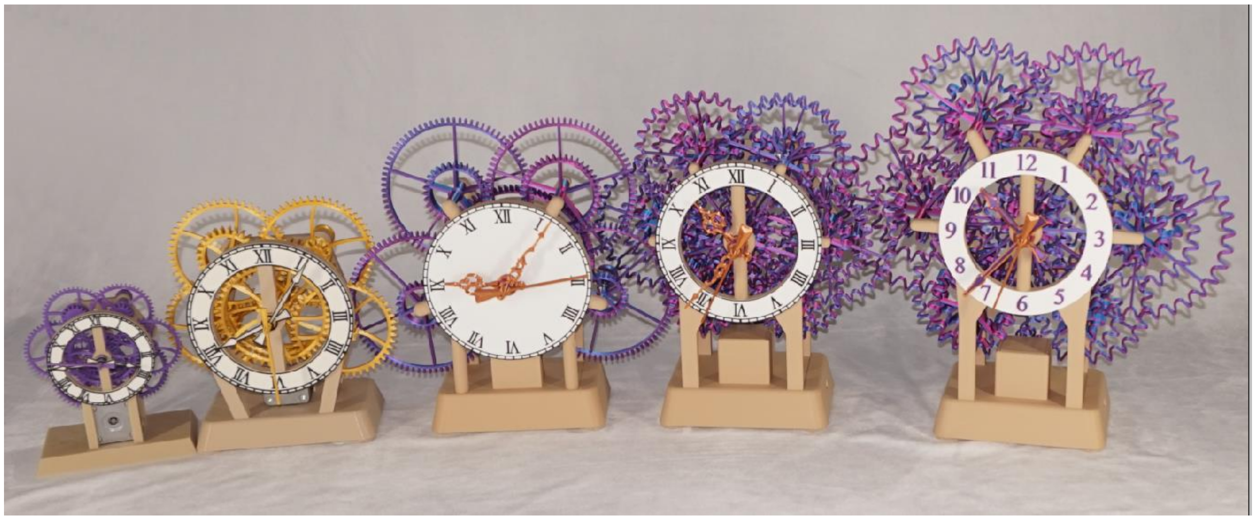


Figure 7: Evolution of the clock design

These two clocks might be last of the 3D printed desk clocks in this series. A line of wooden gear desk clocks could happen in the future.

Mechanical Components

Here are the non-printed mechanical parts required to build the clock. Many parts should be available locally. Ace Hardware Stores in the US or any hobby store that carries K&S metals should have the music wire.

Component	Notes
8X #6x3/4" flat head wood screws (metric M3x20mm or M3.5x20mm)	Frame is assembled using small flat head wood screws to hold the legs in position and hold the base.
51" X 1/16" music wire (metric 1.3m X 1.5mm)	Arbors are made using hardened music wire rod cut into short segments (described later). 1/16" or 1.5mm both work equally well.
4" X 3mm (10cm X 3mm) threaded rod or dowel	Optional dowels needed for SP11 split frame with small printers. Threaded rod is best, but any type of 3mm dowel should work.
1X small pen 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" (25mm) felt pads	Optional pads for under the base of the clock.

Table 1: Mechanical components

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. SP10 and SP11 both use the same length 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.

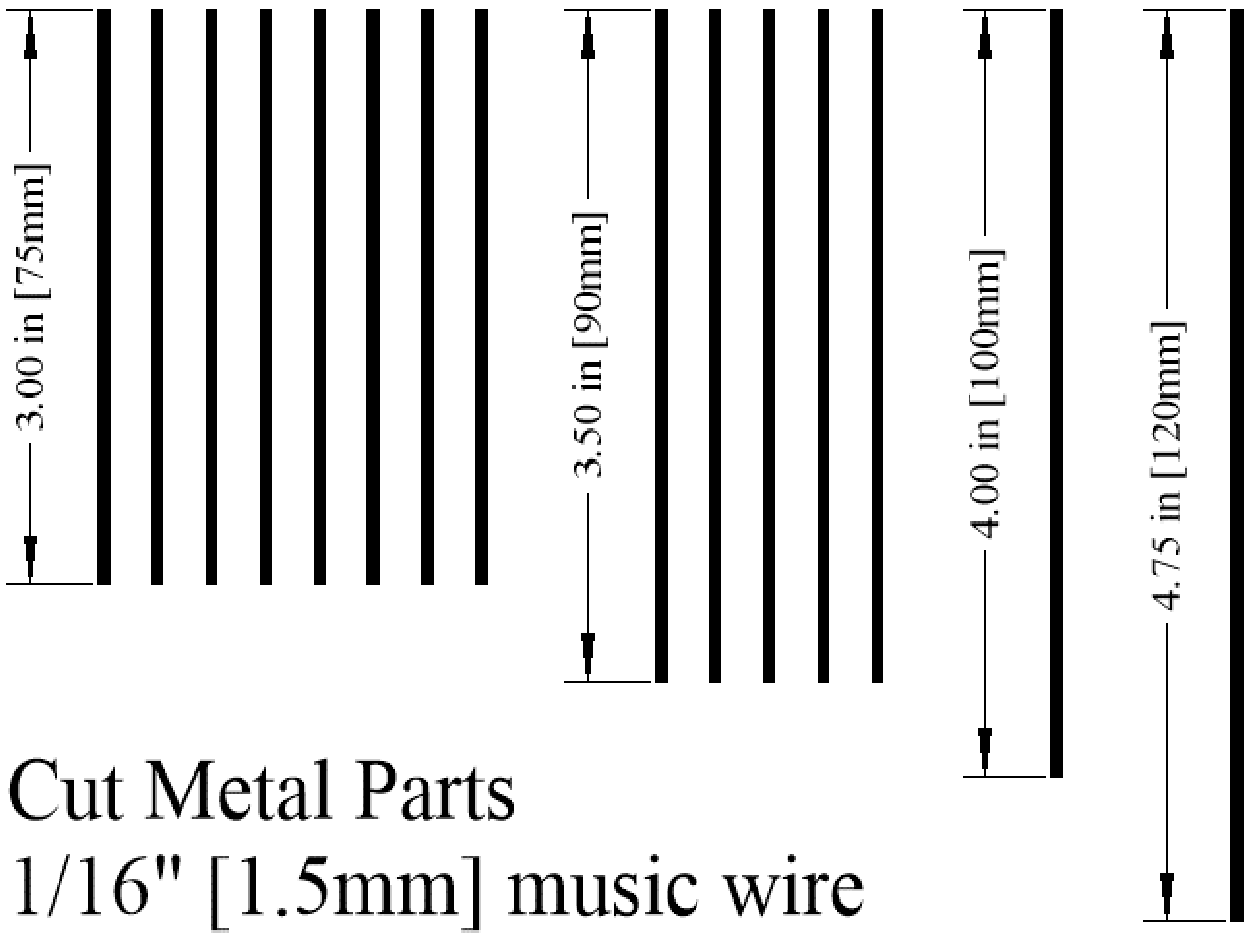


Figure 8: Cut metal parts list

(SP11 large clock option) Smaller printers using the split frame option will need an additional four short pieces of 3mm rod. Solid rod should be fine, but 3mm threaded rod gives the glue better grip. #4-40 threaded rod can also be used. Very little strength is needed, so nearly any type rod close to 3mm in diameter should work.

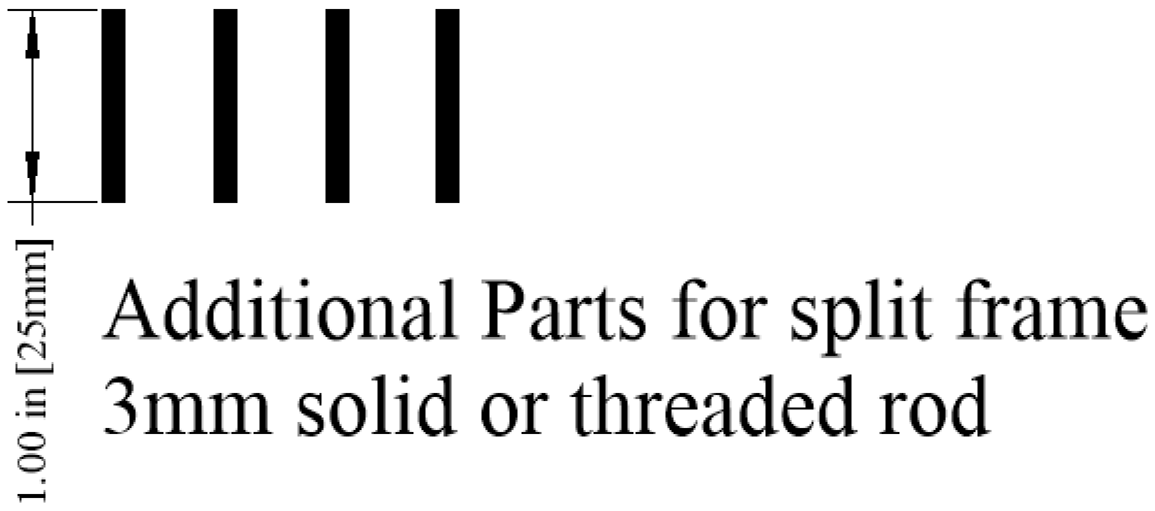


Figure 9: Additional parts needed for large clock split frame

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 printed parts
6) 1.5mm or 1/16" drill bit for cleaning up printed holes
7) Slightly upsized drill bits (1.6mm or 1.8mm) are helpful to enlarge pivot holes if needed

Electrical Components

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

Component	Cost (USD)	Notes
CNC Shield V4[1] or Silent Shield Clock Controller[2]	$10.96 for 3 or ~$10-20 for Etsy board	Main circuit board that all components plug into. The cheapest option is the CNC Shield V4 board. The Silent Shield Clock Controller option is available on Etsy.
Arduino Nano[3]	$16.99 for 3	Processor that controls everything. The design supports either mini-USB or USB-C connectors. USB-C is preferred.
TMC2208[4]	$21.99 for 6	Ultra-quiet Trinamic stepper motor driver.
DS3231 real time clock[5]	$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[6,7]	$16.99 for 0.9 degrees (400 steps) or $10.99 for 1.8 degrees (200 steps)	This design supports almost any NEMA17 bipolar stepper motor. 2-3 ohm motors rated for 1-2A with body lengths up to 40mm. The cable should have a standard 4 pin header with 0.1" (2.54mm) spacing. 0.9 degree (400 step per revolution) motors are best since they run smoother so the clock will be quieter, but 200 step motors can also be used.
Mini USB cable or USB-C cable		For powering and programming the Arduino Nano. Select mini USB or USB-C to match your Arduino Nano.
USB power source		To power the clock. Current is around 100mA.

Table 2: Electrical components

Notes regarding a few items:

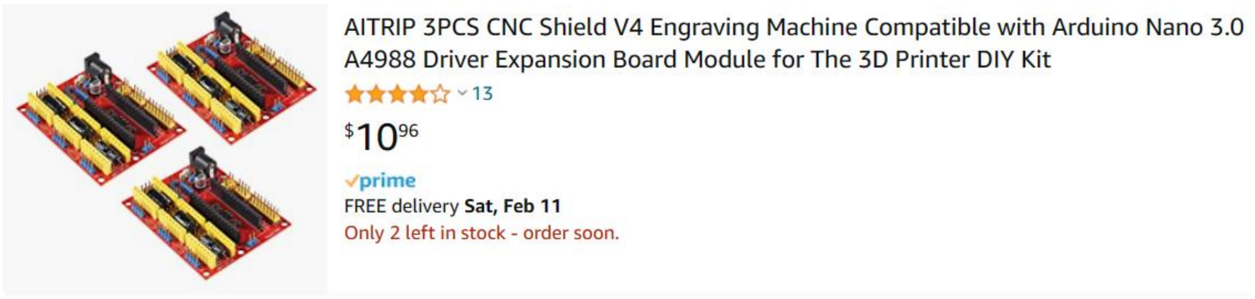
1) The CNC Shield V4 was designed to power a small engraving machine using an Arduino Nano and three A4988 stepper motor drivers. We only need a single stepper motor driver and use a TMC2208 instead of the A4988. The CNC Shield V4 has a bug that needs to be fixed. It is described in the next section. 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) An alternative custom circuit board on Etsy called "Silent Shield Clock Controller" is functionally identical to the features used in the CNC Shield V4 board. It cannot compete with high volume pricing of the CNC Shield V4 and is slightly more expensive. The soldering is easier and it may be available as a pre-built kit if you want to avoid soldering. Either board will operate equally well in the clock.
3) The Arduino Nano is now available with a USB-C connector instead of the old style mini USB connector. The clock will support either style. USB-C cables have become more common, so the Arduino Nano with USB-C connectors is slightly preferrable.
4) The TMC2208 stepper motor driver will be configured for 16X position micro-stepping. It will be wired to bypass the internal regulator and run directly on 5V from the USB port. Vref will be set to the lowest possible motor current.
5) 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?). The spec is 2ppm between 0C and 40C which would correlate to a maximum error of 63 seconds per year. Normal variation at room

temperature should be less than 30 seconds of drift per year. My first two long term test clocks both seem to be tracking better than 1 second per month.

6) Nearly any NEMA17 stepper motor can be used in this clock. The biggest limitation is the physical size. They should have a single shaft and a body length of 40mm or less. The clock works best with 400 step per revolution (0.9 degree) stepper motors, but will also work with cheaper 200 step (1.8 degree motors). My favorite motors are StepperOnline 17ME15-1504S (400 steps, 1.5A, 2.0 ohms). Many other 1.5A motors with 2-4 ohm winding resistance will also work great. I tested StepperOnline part number 17HS15-1504S (200 steps, 1.5A, 2.0 ohms) and a short body 17HE08-1004S (200 steps, 1.0A, 3.6 ohms). I have not found a NEMA17 stepper motor that would not work. The 34 ohm stepper motors used in the original SP6 design can be used, but the more common 2-4 ohm stepper motors are preferred.
7) My first stepper motor desk clock (SP6 Silent Desk Clock) was designed using 1.8 degree stepper motors with 200 steps per revolution. The extra moving gears in this clock have more potential for gear noise that can be reduced by using 0.9 degree stepper motors with 400 steps per revolution. They are a bit more expensive, but I believe it is worth it.

Here are some sample pictures of the electrical components from Amazon. Many are sold in quantities of 3 for a tiny bit more than the single prices. For example, the CNC Shield V4 is available at $6.99 for one, $9.49 for two, or $10.96 for three. Many builders seem to make more than one clock, so 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 bundled with A4988 drivers. Order the bare versions since we will be populating them with a TMC2208. Make sure to order CNC Shield V4 and not CNC Shield V3.


AITRIP 3PCS CNC Shield V4 Engraving Machine Compatible with Arduino Nano 3.0 A4988 Driver Expansion Board Module for The 3D Printer DIY Kit

★★★★☆ 13

$10.96

prime

FREE delivery **Sat, Feb 11**

Only 2 left in stock - order soon.

Figure 10: CNC Shield V4

The CNC Shield V4 board requires edits on the back side, so I decided to create a small custom circuit board with the bare minimum functionality needed in the clock. It is available on Etsy as the “Silent Shield Clock Controller”. I hope to sell it in kit form and as fully assembled modules. Shipping costs within the US are reasonable. Unfortunately, international shipping and customs charges will be high, so the CNC Shield V4 option may be the best option for international orders.

These are the parts that will be available in the Silent Shield Clock Controller kit on Etsy. It includes many of the hardware components including the screws and shaft collar. Fully assembled boards may also be available. Links to the board are in the product description on MyMiniFactory.

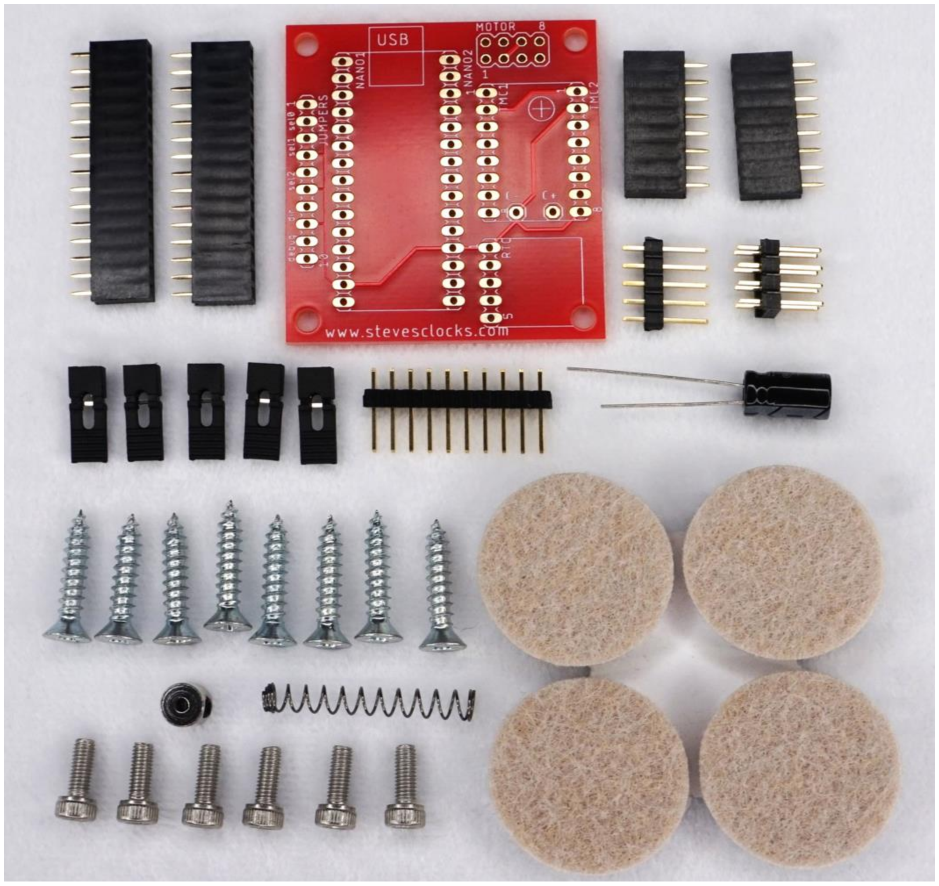
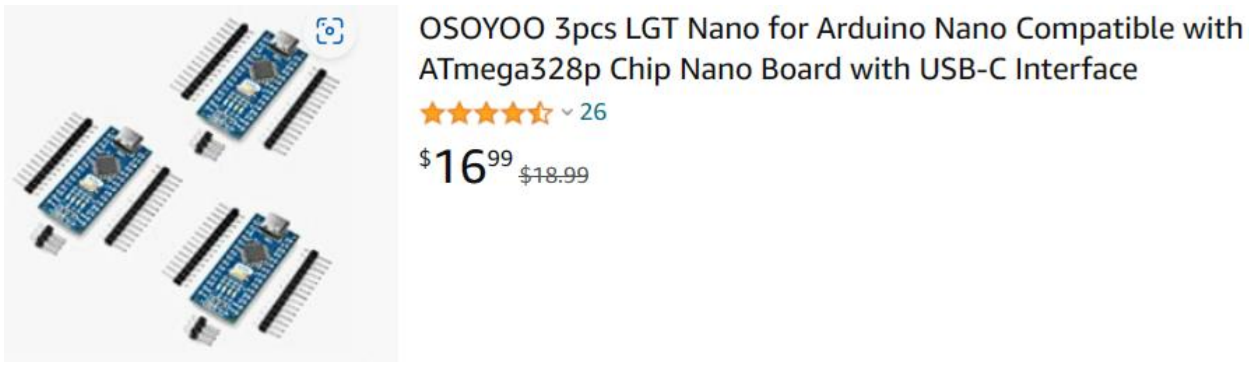


Figure 11: Silent Shield Clock Controller in kit form

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 if you order multiples.

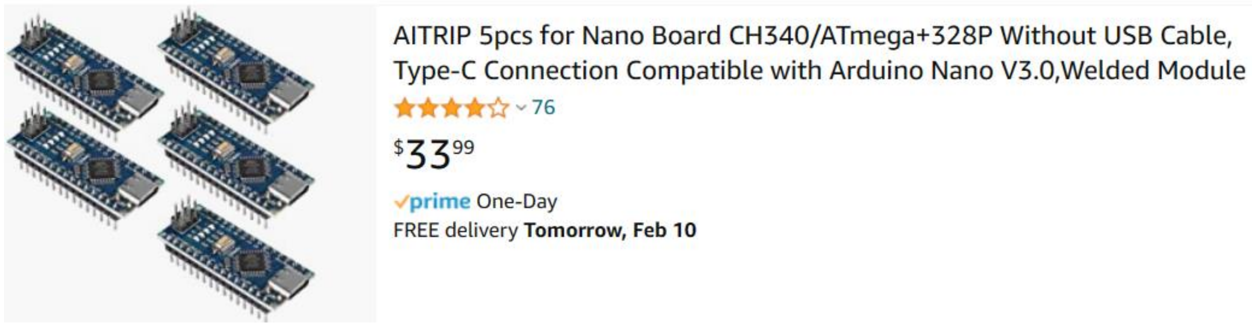


OSOYOO 3pcs LGT Nano for Arduino Nano Compatible with ATmega328p Chip Nano Board with USB-C Interface

26

$16.99 ~~$18.99~~

Figure 12: Arduino Nano



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

76

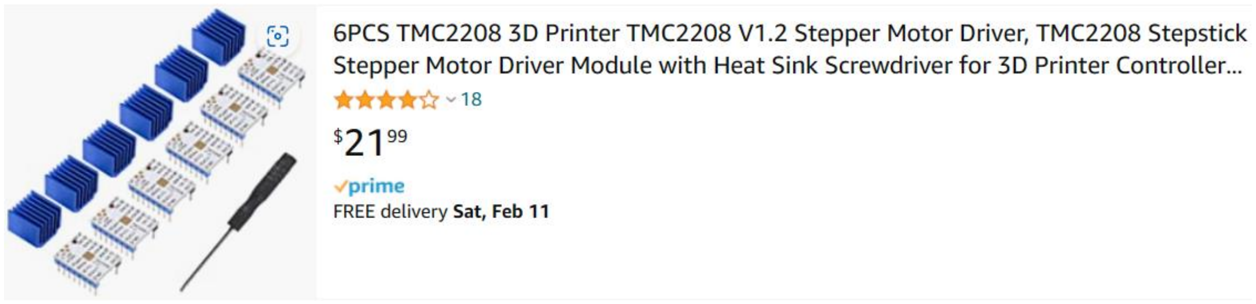
$33.99

prime One-Day

FREE delivery **Tomorrow, Feb 10**

Figure 13: Arduino Nano with pre-installed headers (preferred)

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...

18

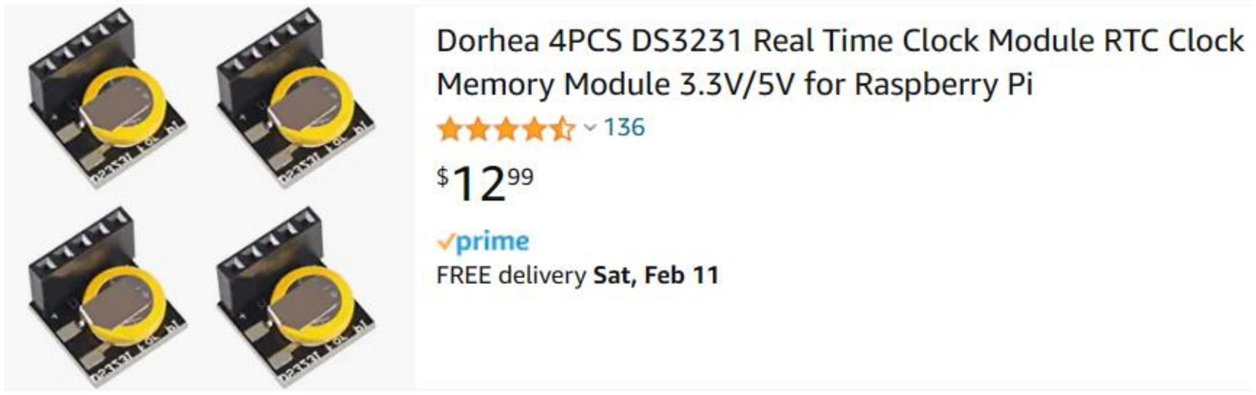
$21.99

prime

FREE delivery **Sat, Feb 11**

Figure 14: TMC2208 stepper motor drivers

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 and a yellow battery or capacitor.



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

136

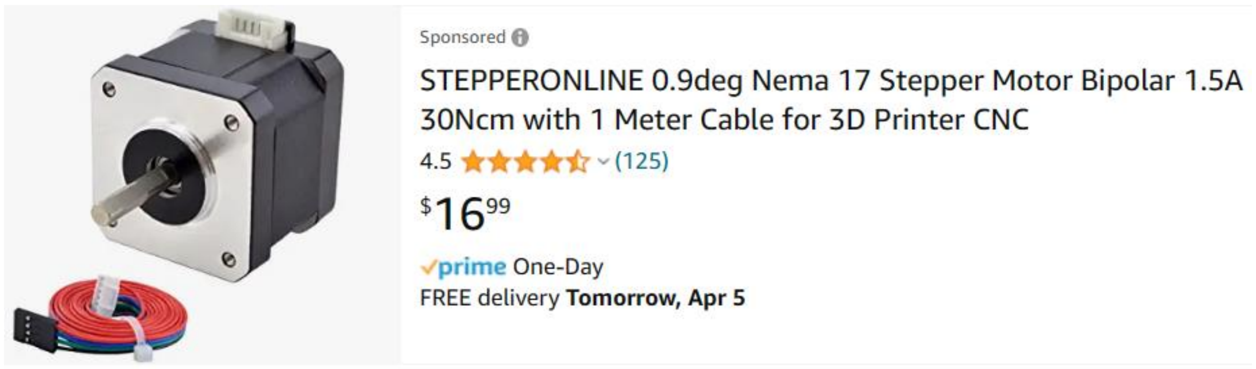
$12.99

prime

FREE delivery **Sat, Feb 11**

Figure 15: DS3231 real time clock module

Any of the following style stepper motors should work. Many 3D printers use similar motors. Make sure to get cables. The controller end needs to have a 4 pin socket to plug directly into the header. A connector at the motor is helpful, but not absolutely required. The first motor shown with 400 steps per revolution (0.9 degree step angle) will be the quietest and is highly desirable. The upcoming Prusa MK4 will use 400 step motors, so maybe they will become more common and the price will drop.


Sponsored

STEPPERONLINE 0.9deg Nema 17 Stepper Motor Bipolar 1.5A 30Ncm with 1 Meter Cable for 3D Printer CNC

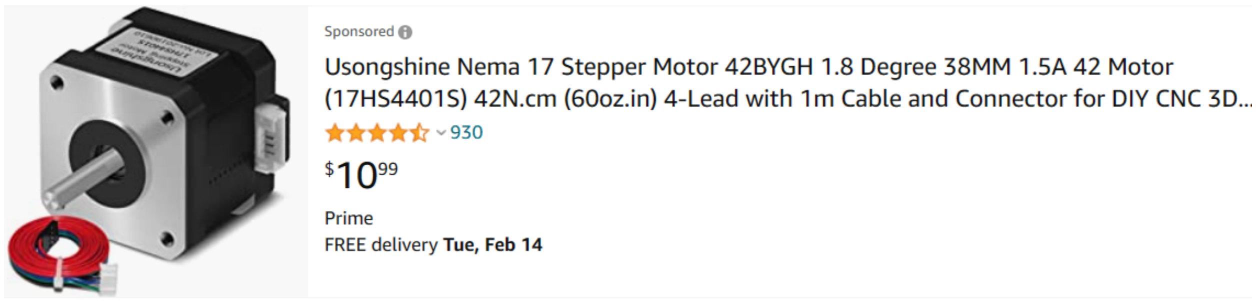
4.5 ★★★★☆ (125)

$16.99

prime One-Day

FREE delivery **Tomorrow, Apr 5**

Figure 16: Preferred stepper motor with 0.9 degree steps


Sponsored

Usongshine Nema 17 Stepper Motor 42BYGH 1.8 Degree 38MM 1.5A 42 Motor (17HS4401S) 42N.cm (60oz.in) 4-Lead with 1m Cable and Connector for DIY CNC 3D...

★★★★☆ 930

$10.99

Prime

FREE delivery **Tue, Feb 14**

Figure 17: Optional stepper motor with 1.8 degree steps

CNC Shield V4 Wiring

This section describes the wiring modifications to be made to the CNC Shield V4 and the DS3231 real time clock. Skip to the next section if you are using the Silent Shield Clock Controller available on Etsy.

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 and slightly more expensive, so I will describe how to modify the red board. Both boards need the edits to connect the RTC and to power the TMC2208 with 5V. There is little reason to recommend the more expensive black KeyeStudio board over the red board.

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

1) Remove the three 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 three jumper pins to the nearest Vcc connection. We always use 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. The black KeyeStudio version of the CNC Shield V4 does not need this edit, but it still needs all the other edits including #3.

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 motor 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.

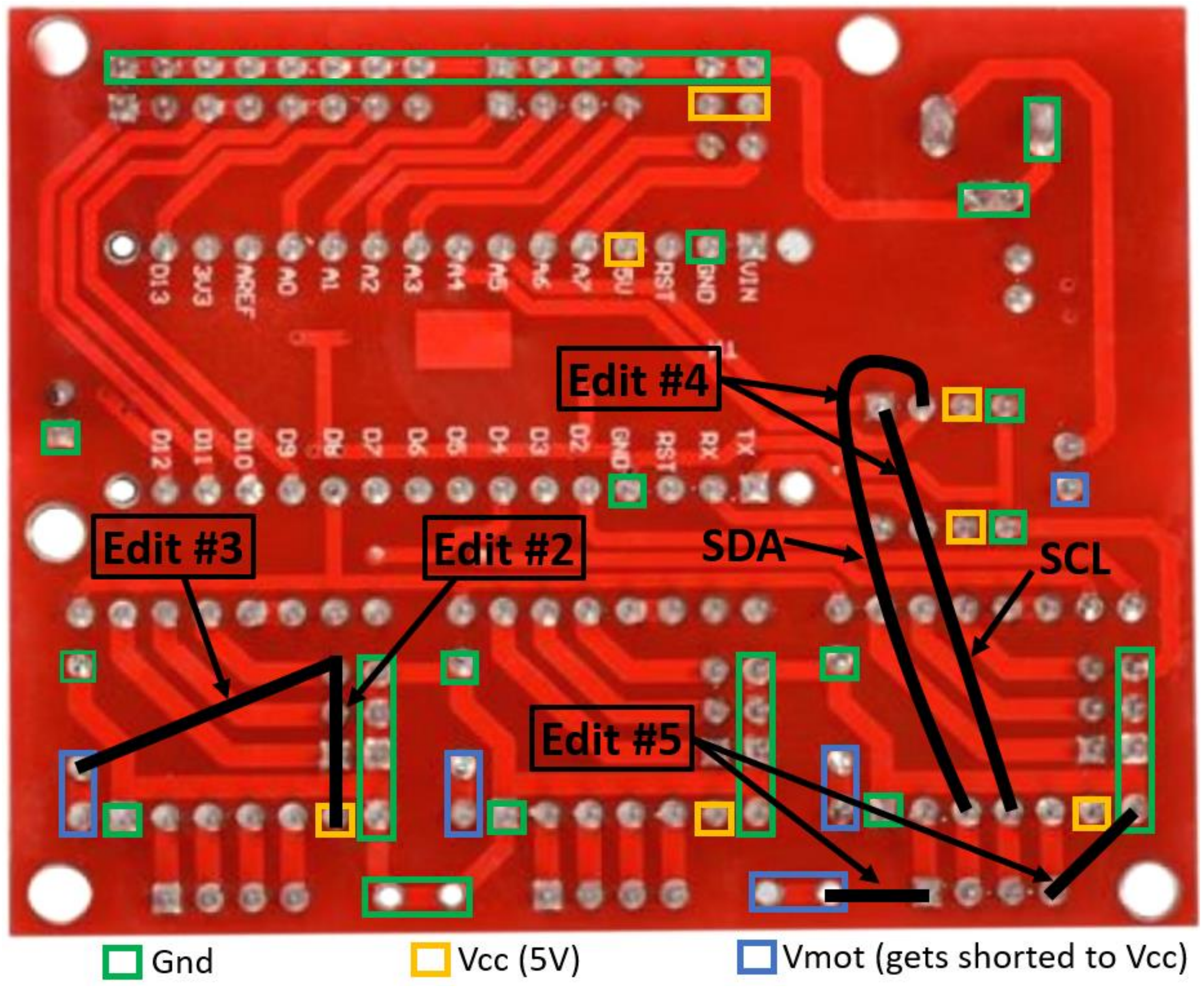


Figure 18: CNC Shield V4 wiring modifications

The top side of the board 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 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 will be used for debug modes and speed selection jumpers.

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

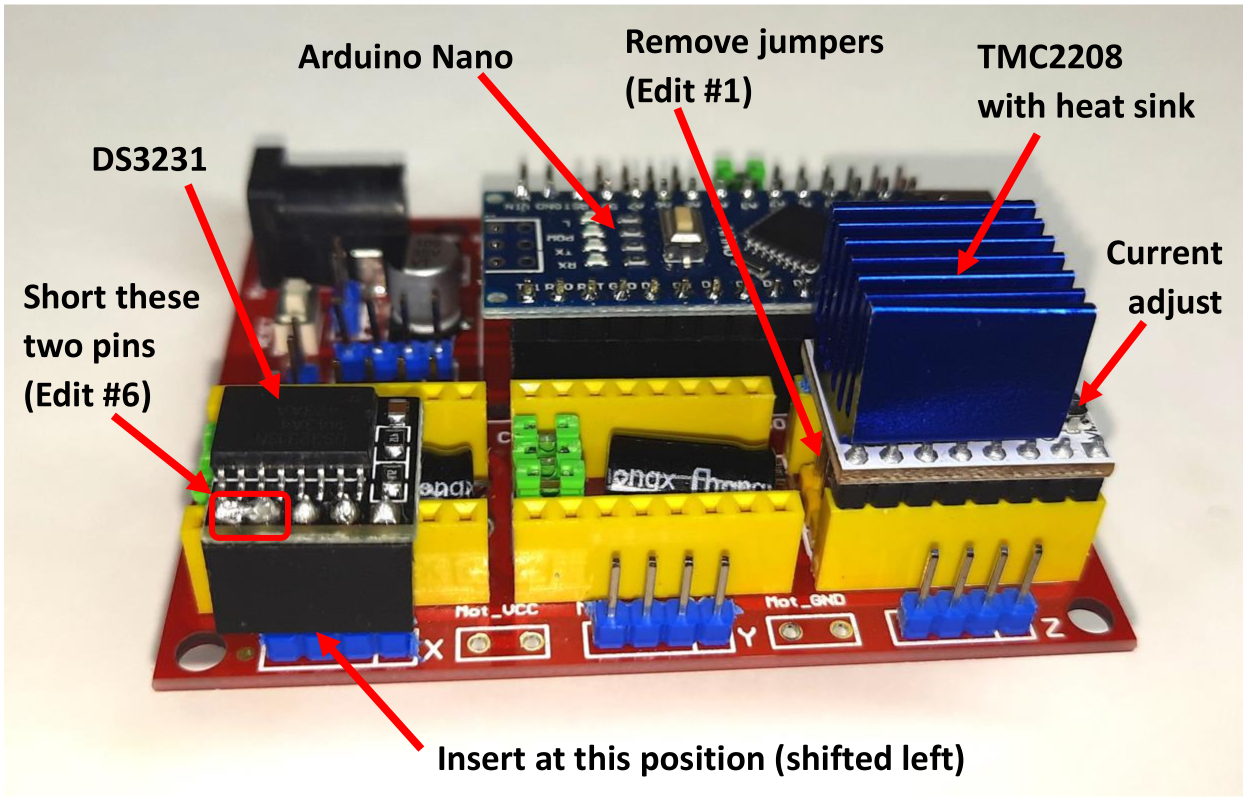


Figure 19: Top side close-up of components

This is the complete board with a stepper motor attached and ready to test. The TMC2208 heat sink might be optional with the low currents used in this clock. Every TMC2208 comes with it, so it makes sense to use it. Make sure not to block the current adjustment potentiometer.

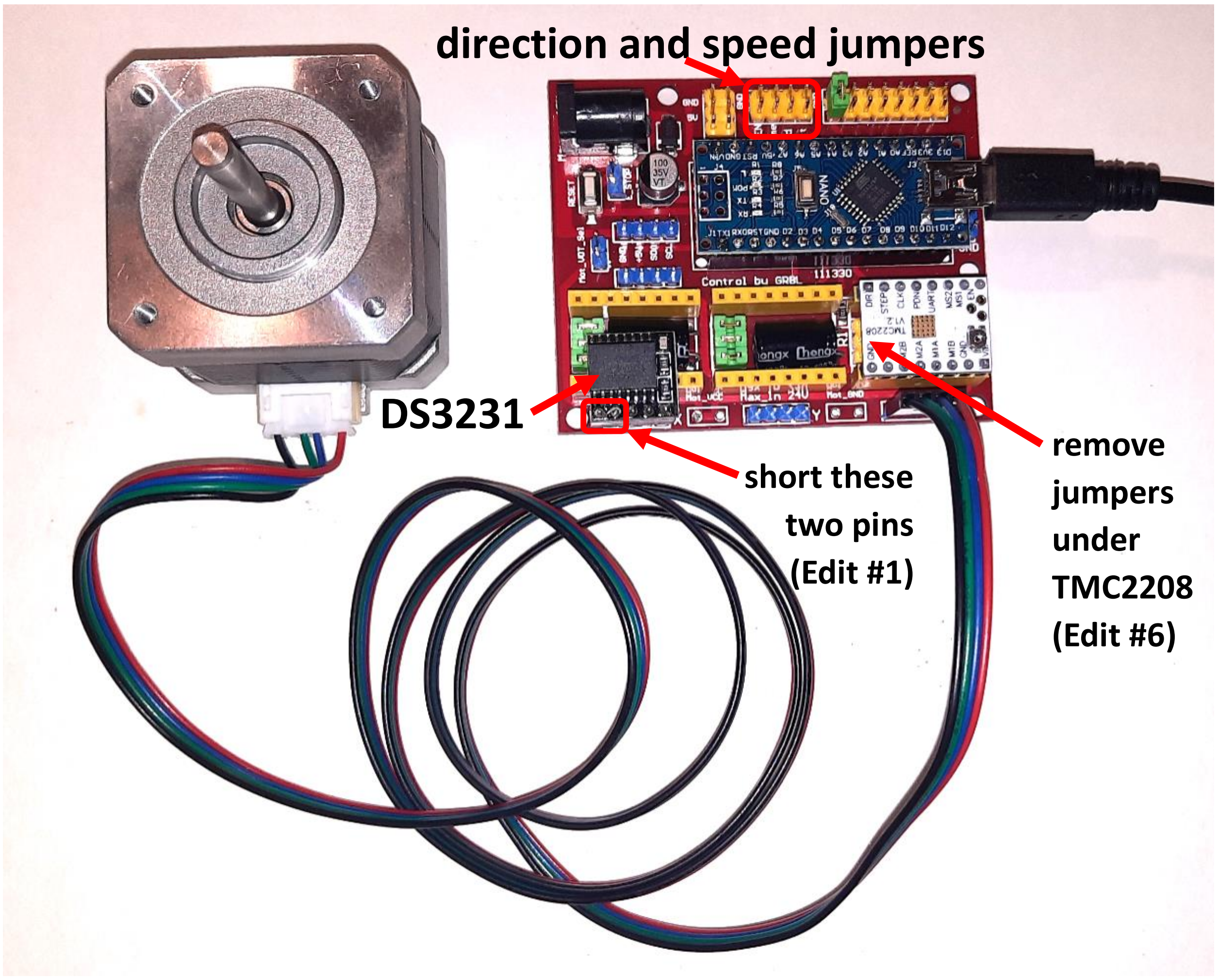

direction and speed jumpers
DS3231
short these two pins (Edit #1)
remove jumpers under TMC2208 (Edit #6)

Figure 20: Assembled driver ready to test

Silent Shield Clock Controller

This section describes the assembly of the Silent Shield Clock Controller kit available on Etsy. Skip to the next section if you built the CNC Shield V4 controller.

Links to the custom board will be on MyMiniFactory or my web site (https://www.stevesclocks.com). It is a slightly more elegant solution with simple soldering. It may even be available as a pre-assembled and programmed module that is ready to run. It is a low volume board that will be more expensive than the CNC Shield V4. It is completely hidden inside the base of the clock, so it is up to you to decide which option you want to use to control your clock.

Assembly of the custom board is straightforward. Insert the components into the top of the board and solder them. The component positions should be obvious. The 10 pin header goes into the row with 10 holes, 15 pin sockets go into the rows with 15 holes, etc. I like to solder a single pin on each component, then hold the board on its side and straighten the component while re-melting the first soldered connection. Solder the remaining pins after each component has been straightened.

The capacitor needs to be inserted with the "-" lead in the mounting hole marked "C-" and enough lead length to bend it to fit under the RTC location.

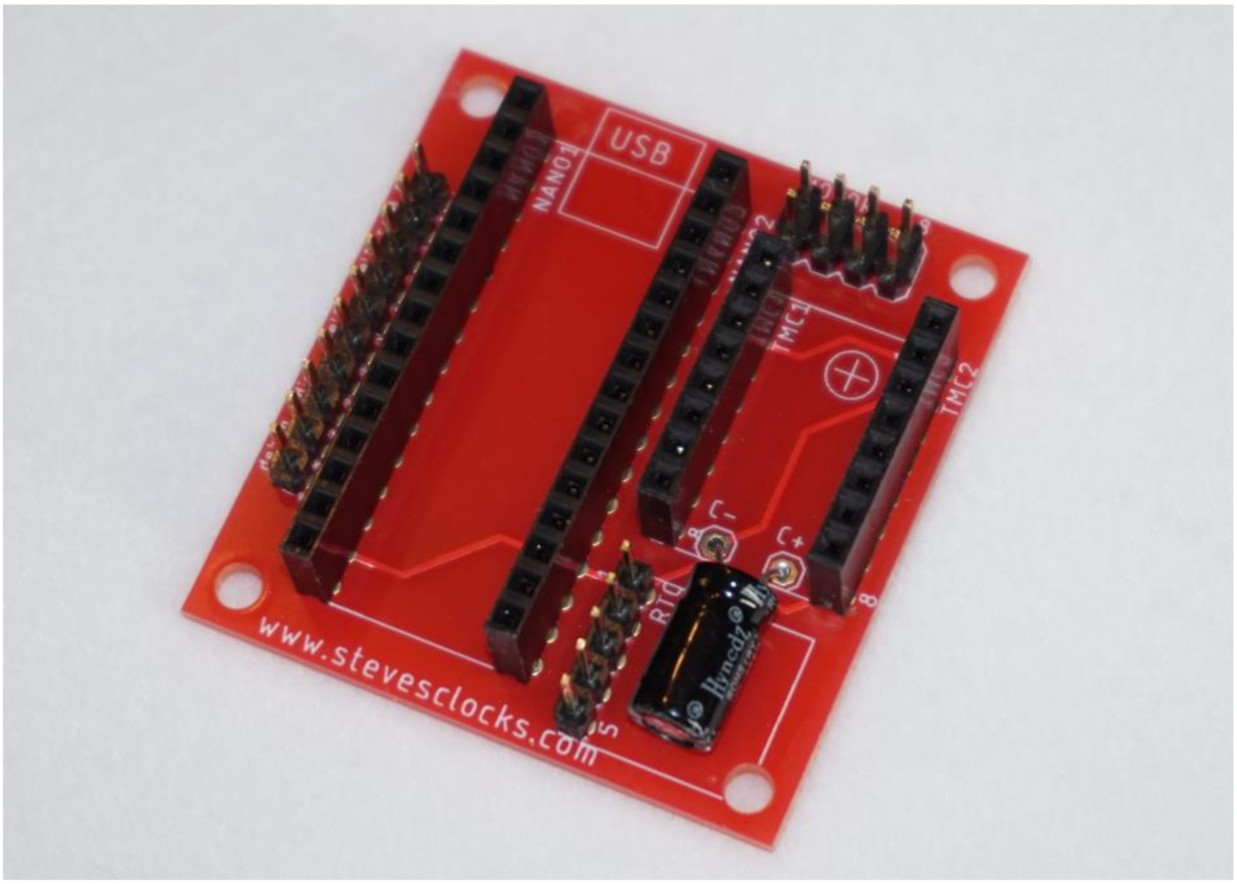


Figure 21: Assembled Silent Shield Clock Controller

Insert the components into the soldered board to match the diagram below. The Arduino Nano is oriented with the USB port at the end marked "USB". The TMC2208 is oriented with the Vref potentiometer positioned above the cross marker on the board. Add the heat sink leaving room for the Vref potentiometer adjustment.

The RTC inserts into the 5 pin header. It doesn't need the CNC Shield V4 edit #6 to short the two top side pins, but will work if you have done that edit for a CNC Shield V4 controller.

The custom Silent Shield Clock Controller is much smaller than the CNC Shield V4 board.

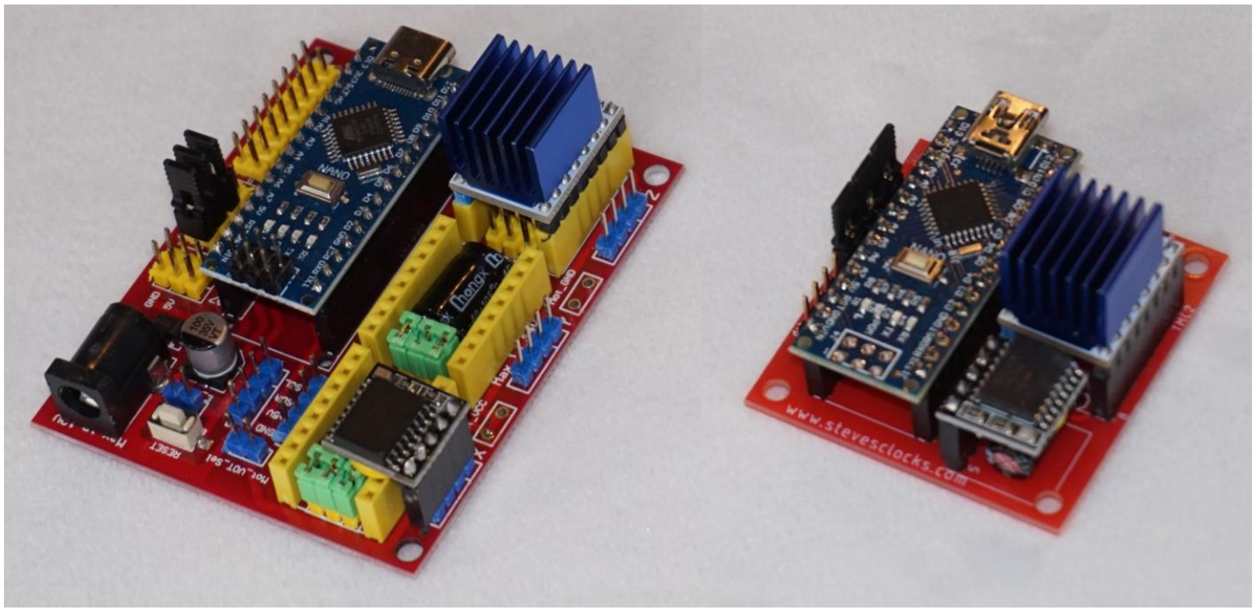


Figure 22: Comparison between CNC Shield V4 and Silent Shield

It becomes a similar size after the printed retainer is added. This allows either controller to fit into the base.

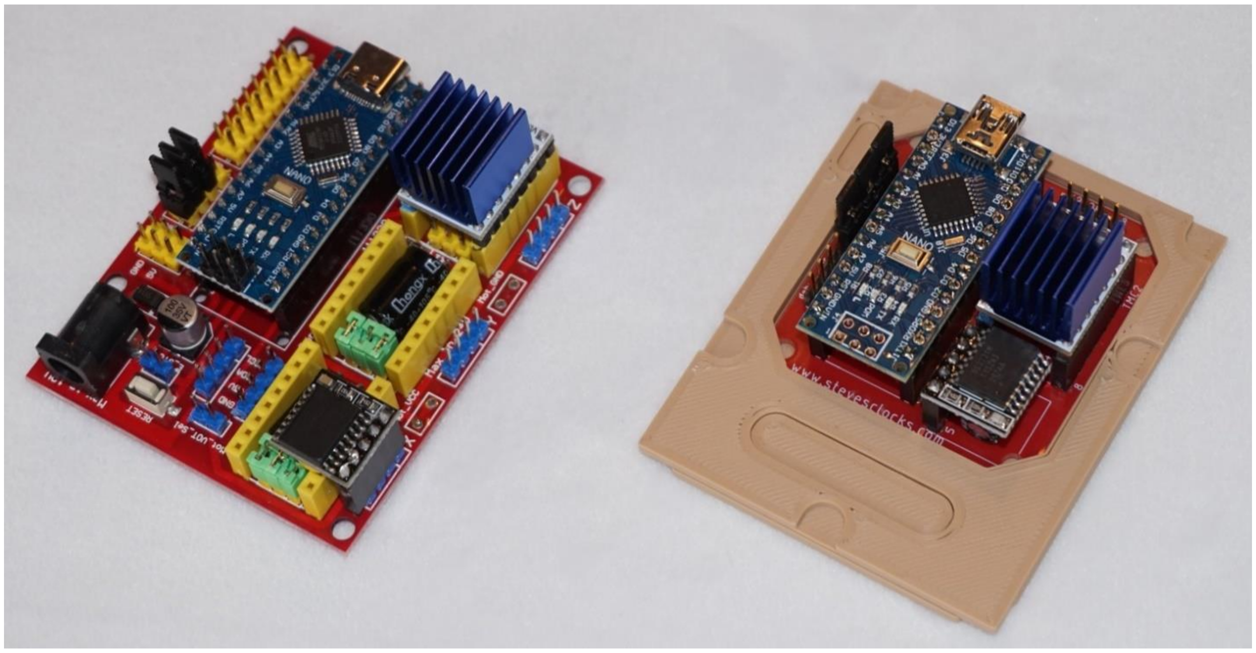


Figure 23: Silent Shield with retainer is similar to CNC Shield V4

The Silent Shield Clock Controller has two header rows for the 4 pin stepper motor cable. The motor can plug into either row depending on the style of motor you have. One row supports straight stepper motor cables and the other row supports crossover wiring. Select the position that works properly for your motor. This allows you to skip the section further down about crossover cables. Simply move the stepper motor cable to the second position if your motor needs it.

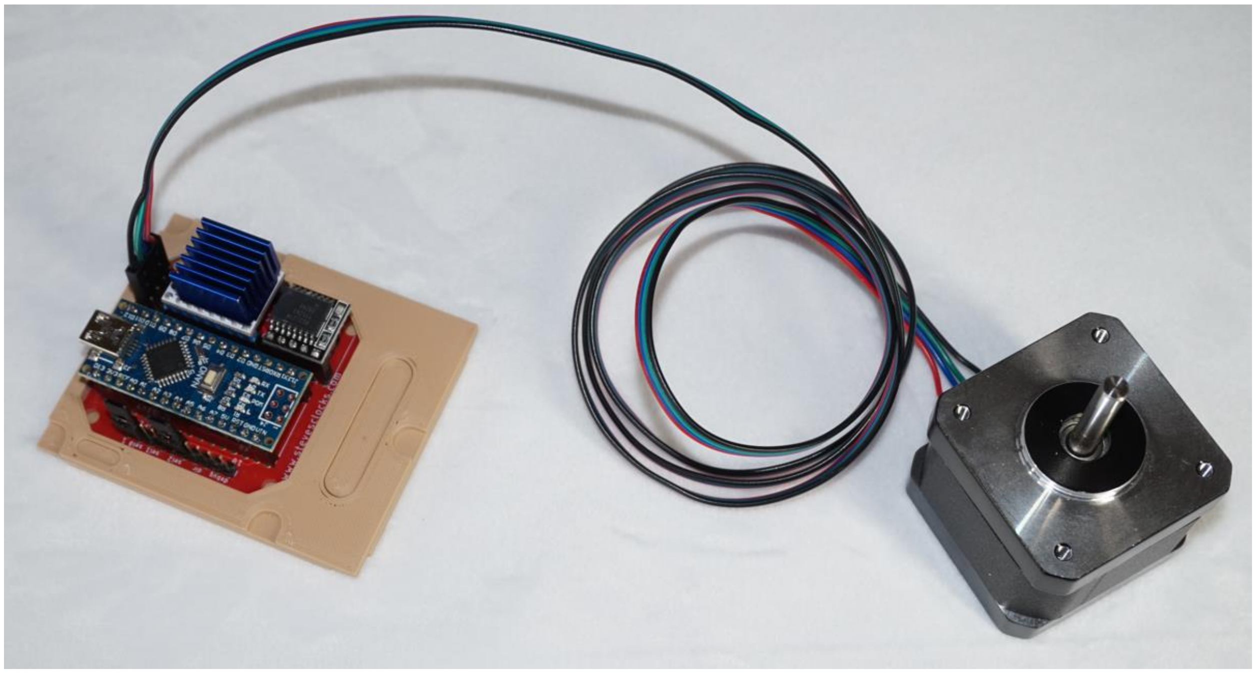


Figure 24: Fully assembled Silent Shield Clock Controller

Motor Test

It is a good idea to program and test the circuit before assembling everything into the completed clock. This allows you to adjust the TMC2208 current levels and determine if the cable modification will be required.

Here are the steps to program the Arduino Nano:

1) Assemble a motor control board by following either the CNC Shield V4 section or the Silent Shield Clock Controller section. Most of the remaining steps are identical for both controllers.
2) Insert components into the controller board (CNC Shield V4 or Silent Shield Clock Controller). Follow the pictures seen in the previous sections.
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 Arduino 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 Arduino Nano sketch from https://www.stevesclocks.com/sp10 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) Verify (compile) the code by clicking the "check mark" icon in the upper left corner of the IDE.
8) 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.
9) 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 jumps 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 or near the lowest possible 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. 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. This modification is only needed for the CNC Shield V4 board. Some stepper motors are wired as 1B, 1A, 2A, and 2B. Other stepper motors are wired as 1B, 2A, 1A, and 2B. The CNC Shield V4 board may need a cable modification to swap the middle two wires near 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.

Note that the Silent Shield Clock Controller option will never need this cable wiring edit. Simply move the motor cable to the other header position and it will work.

Here is the original unmodified stepper motor cable

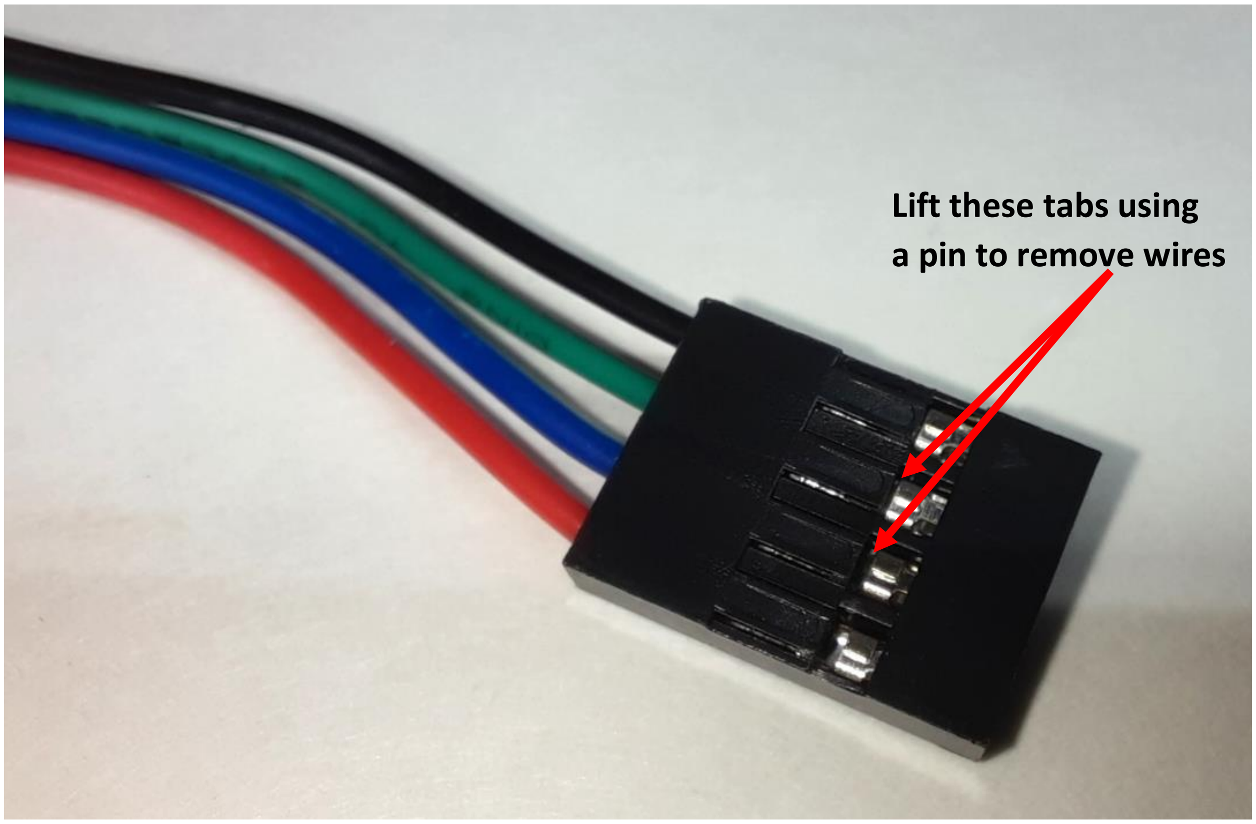


Figure 25: Standard stepper motor cable

Cross over the two wires and insert them back into the connector. This is the modified cable with the middle two pins swapped.

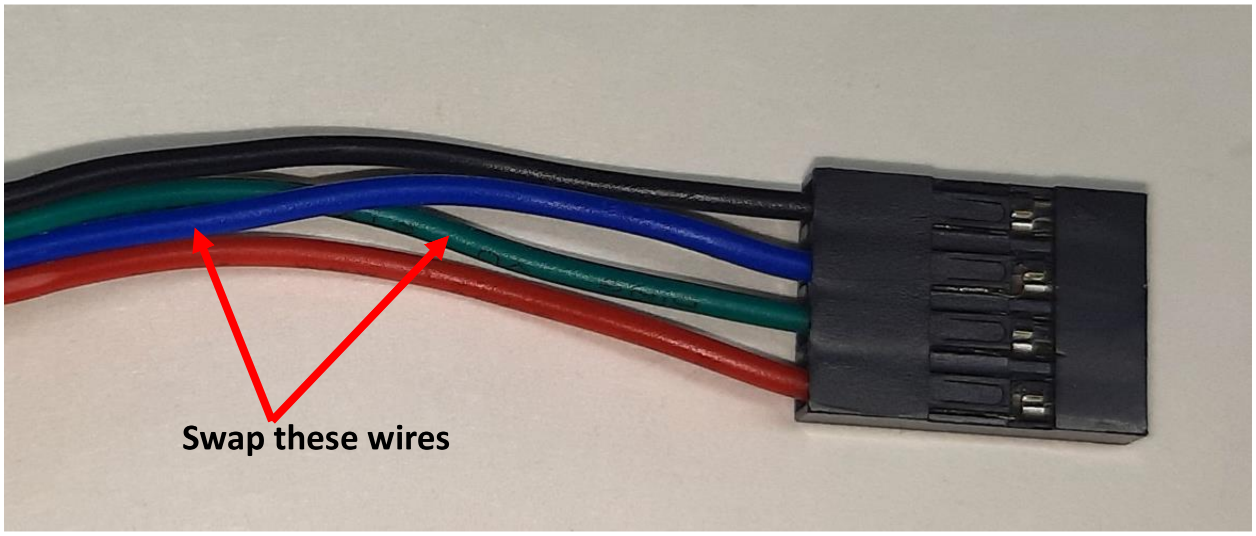


Figure 26: Modified stepper motor wiring

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 algorithm uses jumper blocks to select different motor speeds. This allows one programmed Arduino Nano to be moved between clocks without needing to be re-programmed. These clock are my 3rd and 4th stepper motor desk clocks. The gear ratios on each design are different, therefore each one needs different motor speeds. The speeds also need to be modified when switching between 200 step motors and 400 step motors.

The motor speed is determined by three jumper blocks in the 8-pin header near the top of the CNC Shield V4. The Silent Shield Clock Controller uses 8 pins of the 10 pin header for this function. The left jumper will be used for debugging. Previous algorithms used the left jumper to change motor direction, but this will now be done by reversing the stepper motor cable at the 4 pin or 8 pin header. The left jumper is now used to enter debug mode.

The table below shows the latest motor speed selections. The table may change over time as new clocks are designed. Older versions of the algorithm may have had different speeds assigned. Check the code comments and download the latest version that includes settings for SP10 and SP11.

The medium size clock SP10 will be assigned settings 1 and 2 depending on the stepper motor used. The gear ratios are identical for different motors, but the 400 step motor needs twice as many pulses per second. The large size clock SP11 will be assigned settings 5 and 6.

Setting	Jumpers	RPM	Gear Ratios	Notes
0	0000	1.000	-	reserved
1	0001	4.630	25:15 + 25:9	SP10 (medium) with 200 step motors
2	0010	4.630	25:15 + 25:9	SP10 (medium) with 400 step motors
3	0011	3.600	36:10 or 54:15	SP6 desk clock uses this setting
4	0100	4.667	56:12 or 84:18	SP9 large printed clock uses this setting
5	0101	6.944	25:15 + 25:6	SP11 (large) with 200 step motors
6	0110	6.944	25:15 + 25:6	SP11 (large) with 400 step motors
7	0111	20.000	-	Fast mode for debug

Table 3: Speed selection jumpers

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

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 degree.

The algorithm can be compared 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:

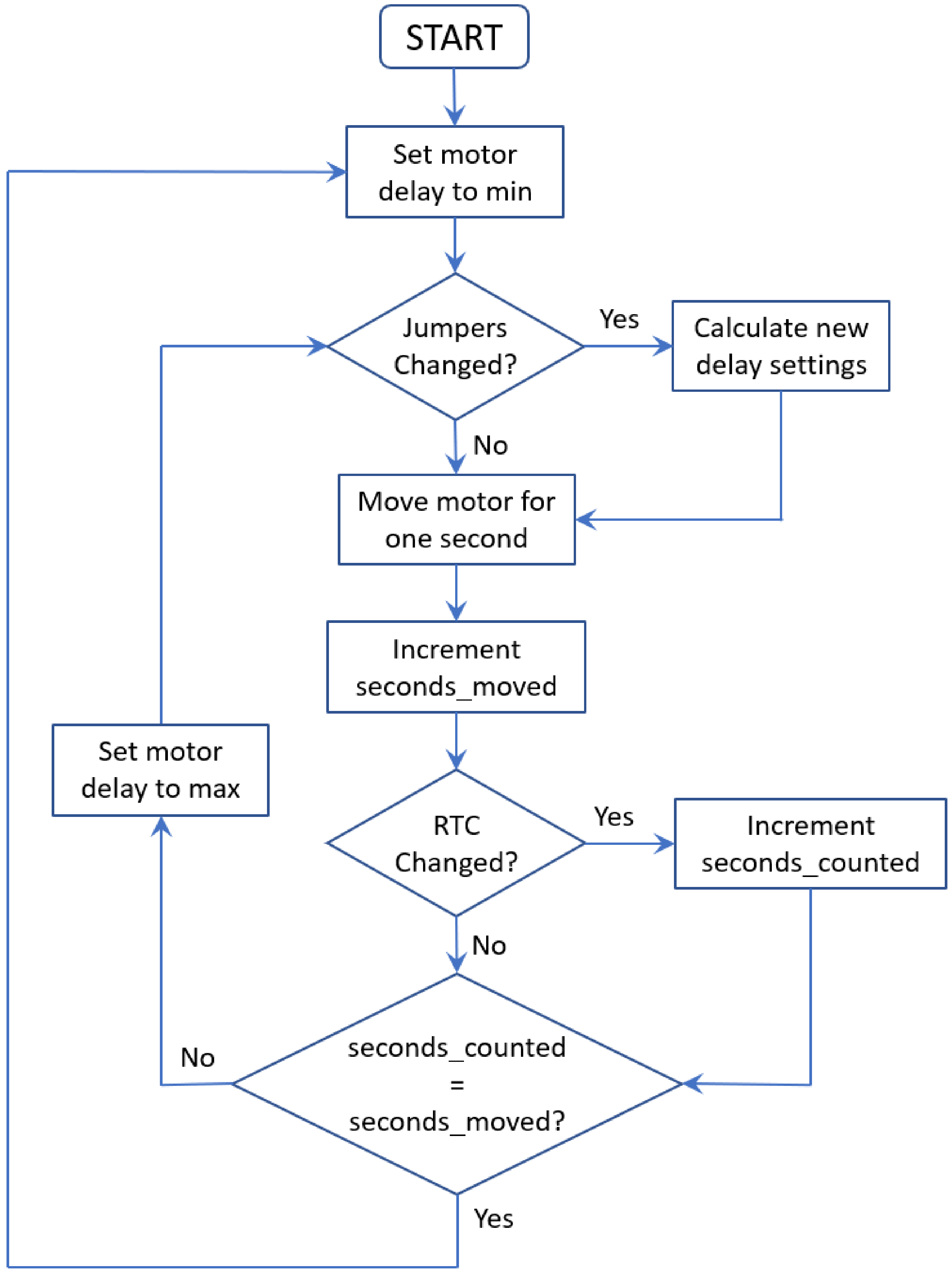


Figure 27: Algorithm flowchart

Algorithm Timing Description

Skip this section if you are not interested in the math behind the stepper motor delay settings.

Non-geared stepper motors make great motors for driving a clock. Pulse them a fixed number of times and they will rotate an exact number of degrees. The Arduino Nano algorithm changes the number of steps depending on the clock gear ratios and stepper motor type.

The calculations would be easy if the stepper motor drove the second hand directly. It would simply need to provide 200 or 400 steps per minute times 16X microstepping. The algorithm for 200 step motors would need (200 * 16) / 60 = 53.3333 steps per second. The easiest way to achieve this would be 53 steps every second plus one additional step every 3 seconds. A 400 step motor would need twice as many steps.

These clocks are more complicated since the motor is two gear stages away from the second hand gear. The motor must spin faster for the second hand to rotate once per minute. The large SP11 clock has easier calculations, so it will be described first. Assume the 200 step per revolution motor with 16X microstepping. The motor has a 6 tooth pinion driving a 25 tooth gear followed by a 15 tooth pinion driving a 25 tooth gear. The motor needs 3200 steps * 25/6 * 25/15 = 22222.2222… steps per minute or 370.370370370… steps per second. This works out to 370 steps per second plus 370 extra steps every 999 seconds. The extra steps can be simplified to 10 extra steps every 27 seconds. The total number of steps will be 370 * 27 + 10 = 10000 steps every 27 seconds.

The medium size clock uses similar calculations with different gear ratios. The motor has a 9 tooth pinion driving a 25 tooth gear followed by a 15 tooth pinion driving a 25 tooth gear. The motor needs 3200 steps * 25/9 * 25/15 = 14814.814814… steps per minute or 246.913580246913580… steps per second. This is a recurring decimal pattern that would need 246 steps per second plus 913580246 extra steps every 999999999 seconds. I used Google to factorize 913580246 into 2 * 37^2 * 333667 and 999999999 into 3^4 * 37 * 333667. The extra steps can be simplified to 2 * 37 extra steps every 3^4 seconds (74 extra steps every 81 seconds). The total number of steps will be 246 * 81 + 74 = 20000 steps every 81 seconds.

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 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 status LED on the Arduino Nano will also blink every time a “+” is displayed.

The serial monitor might look like this:

```
www.stevesclocks.com clock movement - Rev 2.05

speed 1
steps 246 (plus 74/81)
min_delay 1930
max_delay 2040

-------------------------++-+-+-+-++-+-+-+-++-+-+-+-++-+-+-+ (1m) (59)
++-+-++-+-++-++-+-++----------++-+-++-+-++-++-+-++-+-++-+-+- (2m) (120)
+-+-+-++-+-++-+-++-+-++-++-+-++-+-++-+-++----------++-++-+-+ (3m) (179)
```

The header shows the motor configured as speed setting of 1. The motor gets 246 full steps per second plus 74 extra steps every 81 seconds. This corresponds to 246.91358 steps per second and an overall motor speed of 4.944RPM. The motor will toggle between delay values of 1.930ms and 2.040ms to meet the target speed. The remaining lines show a history of the Arduino tracking the RTC. It usually takes a few seconds to start tracking, then a mixture if "+" and "-" indicates proper tracking. The last two numbers in parenthesis are the elapsed time and the total number of seconds counted by the RTC. The gaps showing long sequences of "-" on line 2 and 3 are from the extra 74 steps added every 81 seconds.

Debug Modes

The algorithm includes several debug modes to help identify potential hardware issues without connecting to the debug monitor. These debug modes are entered by inserting the 4th jumper. This jumper is labeled "CoolEN" on the CNC Shield V4 board and "dir" on the Silent Shield Clock Controller. The remaining three jumpers select the desired mode.

Most of the debug modes display their result using an LED on the Arduino Nano. The power LED is always lit. The second LED indicates debug mode status. Debug modes 9, 10, and 12 will light the 2nd LED to indicate that the jumper is functional. Debug mode 11 shows the Arduino speed. Debug mode 13 is one of the most useful. It indicates that the RTC is operating properly. Debug modes 14 and 15 test the TMC2208 stepper motor driver.

Mode	Jumpers	Name	Notes
8	1000	LED off	No test running, second LED will be off.
9	1001	LED on	Simple hardware test to indicate that jumper 0 is working.
10	1010	LED on	Simple hardware test to indicate that jumper 1 is working.
11	1011	Arduino delay test	LED will blink at 1Hz (0.5s on, 0.5s off). This test shows that the built in Arduino Nano delay routines are accurate.
12	1100	LED on	simple hardware test to indicate that jumper 2 is working
13	1101	RTC test	LED should blink at 0.5Hz if the RTC is working. The LED will toggle once per second if the RTC is operating properly.
14	1110	Rotate motor CW	Rotate motor one direction at low speed
15	1111	Rotate motor CCW	Rotate motor in opposite direction at a higher speed

Table 4: Debug mode jumpers

Printing the Parts

The printed parts list is broken into three sections. The first section includes the parts that are common to both clocks. The second section includes the parts that are unique to the medium size clock and the third section includes parts unique to the large clock. Many of the names in the 2nd and 3rd list are the same, but the print times are different since the medium size clock parts are smaller.

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 with additional options for mini-USB or USB-C power plugs. The holddown clips used to keep the motor controller positioned properly will need to match your power plug direction. The clock hands have options for a nominal fit on the shaft, <filename>_1.stl with a slightly tighter fit, and <filename>_2.stl that is even tighter. Select the version that fits best.

These are the parts that are identical for both clocks. Print one of each item as listed below.

Part Name (both clocks)	Color	Print	Time	Filament	Notes
base_back_mini	tan	1	11h 43m	166.46g	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		11h 43m	166.43g	
base_left_mini	tan		11h 45m	164.58g	
base_left_usbc	tan		11h 44m	164.41g	
base_right_mini	tan		11h 43m	164.42g	
base_right_usbc	tan		11h 44m	164.41g	
base_holddown_back	any	1	1h 38m	16.98g	Print one of these to match base power plug direction
base_holddown_left	any		1h 30m	15.61g	
base_holddown_right	any		1h 30m	15.60g	
base_motor_cover	tan	1	0h 48m	8.04g	
base_motor_mount	tan	1	3h 31m	49.99g	
base_retainer_lower	any	0	0h 35m	7.49g	Optional retainer used with Silent Shield Clock Controller
base_retainer_upper	any	0	0h 31m	6.04g	
hand_hour_gothic	copper	1	0h 23m	2.35g	Print an hour hand and minute hand of either style, spade option needs a color change at 2.20mm. "_1" and "_2" options have a tighter fit on the shaft if needed
hand_hour_spade	purple, copper		0h 14m	2.15g	
hand_minute_gothic	copper	1	0h 19m	1.97g	
hand_minute_spade	purple, copper		0h 16m	2.31g	
hand_second	copper	1	0h 27m	3.42g	Print with 14 perimeters
	Total	**31**	**18h 49m**	**248.58g**	

Table 5: Printed parts used in both size clocks

The medium size SP10 clock needs the following printed parts. There was no need to split any parts since the largest component still fits on a Prusa Mini sized printer (180x180mm). The dial has options for traditional roman numerals or simple numbers. It can also be printed with an open or solid face. Select any style you like.

Print these parts for the medium sized clock (SP10).

Part Name (SP10)	Color	Print	Time	Filament	Notes
frame_back_center	tan, purple	1	5h 27m	62.72g	Add a color change at 10.40mm
frame_back_left	tan	1	1h 9m	14.02g	
frame_back_right	tan	1	1h 9m	14.04g	
frame_dial_numbers_open	tan, ivory, purple	1	6h 1m	84.99g	Print one of these dial styles, select roman numerals or simple numbers, and solid or open dial. Add color changes at 10.40mm and 12.20mm
frame_dial_numbers_solid	tan, ivory, purple		6h 46m	123.32g	
frame_dial_roman_open	tan, ivory, black		6h 10m	85.95g	
frame_dial_roman_solid	tan, ivory, black		6h 55m	124.25g	
frame_front_left	tan	1	1h 11m	14.35g	
frame_front_right	tan	1	1h 12m	14.37g	
gear_motor_9	purple	1	1h 0m	7.76g	Classic slicer, 5 perimeters
gear0_15_8	purple	1	2h 16m	19.59g	Classic slicer, 5 perimeters
gear0b_25_10	purple	1	2h 23m	21.00g	Classic slicer, 5 perimeters
gear1_15	purple	1	1h 14m	10.45g	Classic slicer, 5 perimeters
gear1b_25_15	purple	1	4h 22m	41.23g	Classic slicer, 5 perimeters
gear2_25_8	purple	1	2h 13m	18.85g	Classic slicer, 5 perimeters
gear3_32_20_10	purple	1	3h 47m	31.92g	Classic slicer, 5 perimeters
gear3b_15	purple	1	1h 18m	10.77g	Classic slicer, 5 perimeters
gear4_30_10	purple	1	2h 35m	23.31g	Classic slicer, 5 perimeters
gear4b25_15	purple	1	3h 2m	28.53g	Classic slicer, 5 perimeters
gear5_15	purple	1	1h 15m	10.53g	Classic slicer, 5 perimeters
gear5_30	purple	1	1h 13m	13.01g	Classic slicer, 5 perimeters
gear5_collar	purple	0	0h 7m	0.53g	Optional printed shaft collar
gear5_knob	purple	1	0h 18m	3.12g	
gear5b_25_10	purple	1	1h 51m	18.10g	Classic slicer, 5 perimeters
gear6_25_8	purple	1	2h 30m	20.15g	Classic slicer, 5 perimeters
gear7_32_20_10	purple	1	3h 0m	28.10g	Classic slicer, 5 perimeters
gear7b_15	purple	1	2h 37m	21.62g	Classic slicer, 5 perimeters
gear8_30_15	purple	1	1h 57m	18.40g	Classic slicer, 5 perimeters
	Total	**31**	**55h 0m**	**550.93g**	

Table 6: Printed parts unique to SP10

The large size SP11 clock needs the parts listed on the following page. The frame can be printed in one piece at 214x172mm or 157x172mm when "split" for smaller printers. The one-piece versions are preferred if your printer is large enough, since they have fewer assembly steps.

Print these parts for the large sized clock (SP11).

Part Name (SP11)	Color	Print	Time	Filament	Notes
frame_back_center	tan, purple	1	5h 7m	62.98g	Add a color change at 10.40mm
frame_back_split_center	tan, purple	0	4h 44m	58.62g	Optional frame for smaller printers replaces frame_back_center, add a color change at 10.40mm
frame_back_split_left	tan, purple	0	0h 21m	3.23g	
frame_back_split_right	tan, purple	0	0h 23m	3.43g	
frame_back_left	tan, purple	1	1h 24m	17.33g	Add a color change at 10.40mm
frame_back_right	tan, purple	1	1h 34m	18.04g	Add a color change at 10.40mm
frame_dial_numbers_open	tan, ivory, purple	1	6h 8m	86.85g	Print one of these dial styles, select roman numerals or simple numbers, solid or open dial, and full size or split option for smaller printers. Add color changes at 10.40mm and 12.20mm
frame_dial_numbers_solid	tan, ivory, purple		7h 4m	127.28g	
frame_dial_roman_open	tan, ivory, black		6h 12m	87.31g	
frame_dial_roman_solid	tan, ivory, black		7h 6m	127.55g	
frame_dial_split_numbers_open	tan, ivory, purple		8h 48m	83.59g	
frame_dial_split_numbers_solid	tan, ivory, purple		6h 41m	123.78g	
frame_dial_split_roman_open	tan, ivory, black		5h 53m	84.06g	
frame_dial_split_roman_solid	tan, ivory, black		6h 47m	124.30g	
frame_dial_split_right	tan	0	0h 19m	2.75g	Side pieces used with split dial option
frame_dial_split_left	tan	0	0h 19m	2.76g	
frame_front_left	tan	1	1h 18m	16.29g	
frame_front_right	tan	1	1h 18m	16.31g	
gear_motor_6	purple	1	1h 2m	7.08g	Classic slicer, 5 perimeters
gear0_15_8	purple	1	2h 42m	24.69g	Classic slicer, 5 perimeters
gear0b_25_10	purple	1	2h 44m	26.21g	Classic slicer, 5 perimeters
gear1_15	purple	1	1h 24m	12.75g	Classic slicer, 5 perimeters
gear1b_25_15	purple	1	5h 2m	51.53g	Classic slicer, 5 perimeters
gear2_25_8	purple	1	2h 38m	23.64g	Classic slicer, 5 perimeters
gear3_32_20_10	purple	1	4h 18m	40.18g	Classic slicer, 5 perimeters
gear3b_15	purple	1	1h 29m	13.06g	Classic slicer, 5 perimeters
gear4_30_10	purple	1	3h 1m	28.88g	Classic slicer, 5 perimeters
gear4b25_15	purple	1	3h 31m	35.69g	Classic slicer, 5 perimeters
gear5_15	purple	1	1h 26m	12.85g	Classic slicer, 5 perimeters
gear5_30	purple	1	1h 30m	16.60g	Classic slicer, 5 perimeters
gear5_collar	purple	0	0h 7m	0.53g	Optional printed shaft collar
gear5_knob	purple	1	0h 18m	3.12g	
gear5b_25_10	purple	1	2h 10m	23.04g	Classic slicer, 5 perimeters
gear6_25_8	purple	1	2h 55m	25.04g	Classic slicer, 5 perimeters
gear7_32_20_10	purple	1	3h 31m	36.03g	Classic slicer, 5 perimeters
gear7b_15	purple	1	2h 58m	26.20g	Classic slicer, 5 perimeters
gear8_30_15	purple	1	2h 16m	22.43g	Classic slicer, 5 perimeters
	Total	**31**	**61h 43m**	**646.82g**	

Table 7: Printed parts unique to SP11

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 needs 5 perimeters, so print all gears using 5 perimeters.

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

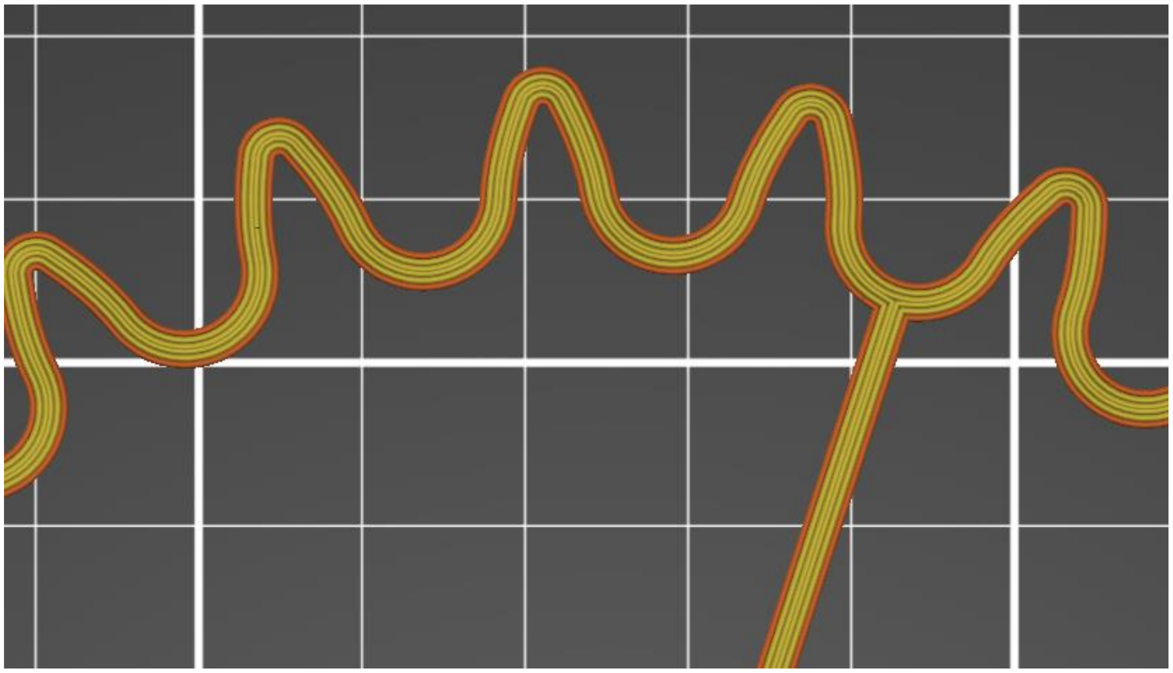


Figure 28: Gear slicer detail with five perimeters

And the center hub that prints solid with 5 perimeters.

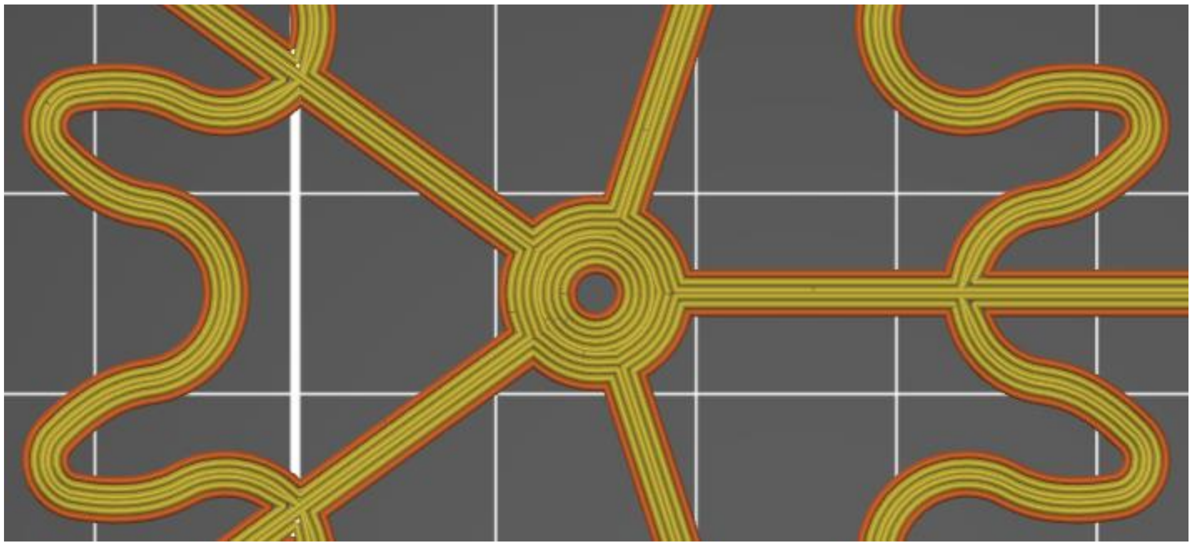


Figure 29: Gear center hub

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.

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 many of the benefit of the clean gear tooth profiles. The additional gap fill increases print time and increases stringing resulting in rougher gear surfaces. The gear profiles used in these clocks have less potential for extra gap fill compared to my other clocks, however they still print cleaner **without** Arachne.

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.

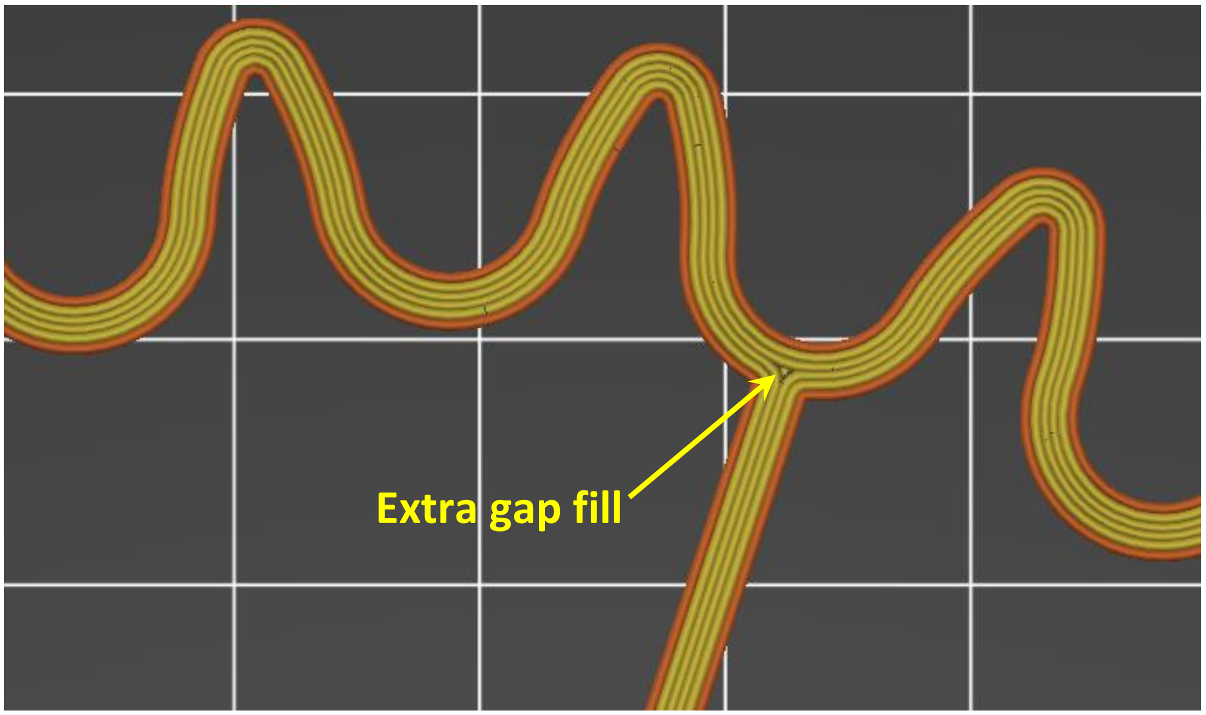
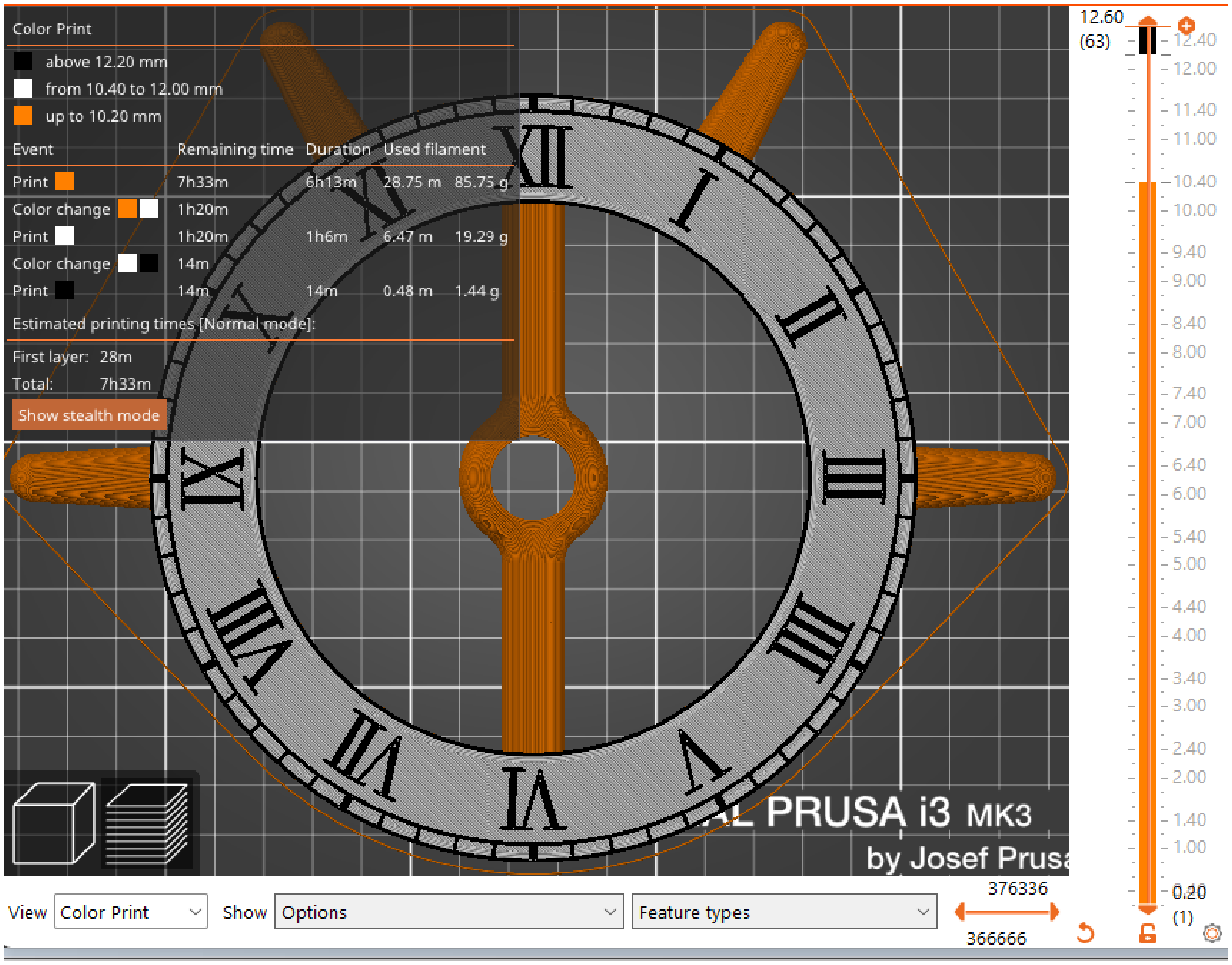


Figure 30: Arachne side effects for gears

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, 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.



Color Print

⬛	above 12.20 mm
⬜	from 10.40 to 12.00 mm
🟧	up to 10.20 mm

Event	Remaining time	Duration	Used filament	
Print 🟧	7h33m	6h13m	28.75 m	85.75 g
Color change 🟧⬜	1h20m			
Print ⬜	1h20m	1h6m	6.47 m	19.29 g
Color change ⬜⬛	14m			
Print ⬛	14m	14m	0.48 m	1.44 g

Estimated printing times [Normal mode]:

First layer: 28m

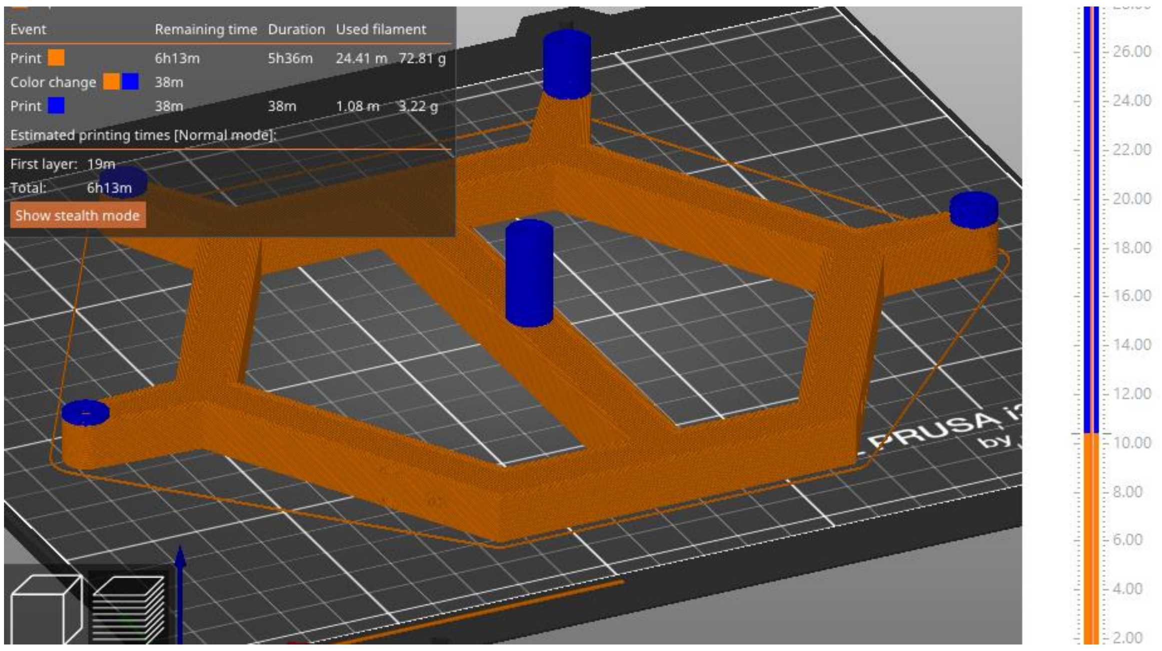
Total: 7h33m

Show stealth mode

View: Color Print   Show: Options   Feature types

Figure 31: 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.



Event	Remaining time	Duration	Used filament
Print 🟧	6h13m	5h36m	24.41 m 72.81 g
Color change 🟧🟦	38m		
Print 🟦	38m	38m	1.08 m 3.22 g

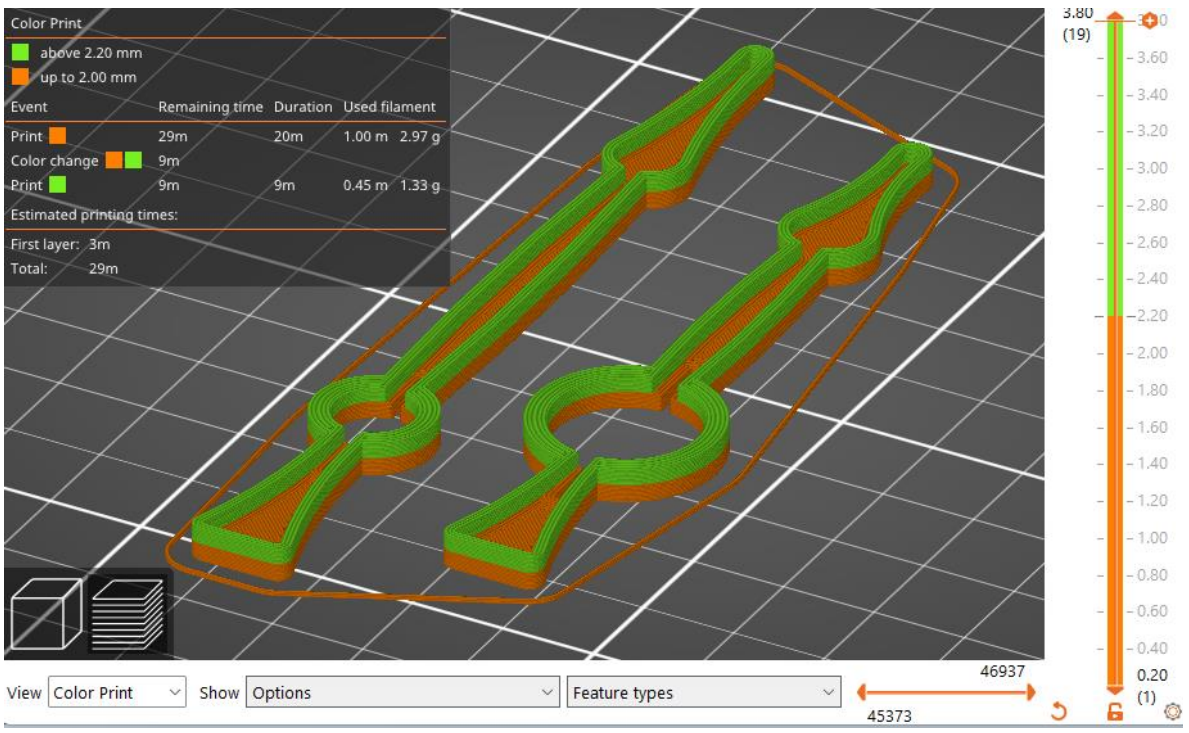
Estimated printing times [Normal mode]:

First layer: 19m
Total: 6h13m

Show stealth mode

Figure 32: Optional back frame layer changes

The spade style hands need a color change at 2.20mm to add a contrasting color.



Color Print

🟩 above 2.20 mm
🟧 up to 2.00 mm

Event	Remaining time	Duration	Used filament
Print 🟧	29m	20m	1.00 m 2.97 g
Color change 🟧🟩	9m		
Print 🟩	9m	9m	0.45 m 1.33 g

Estimated printing times:

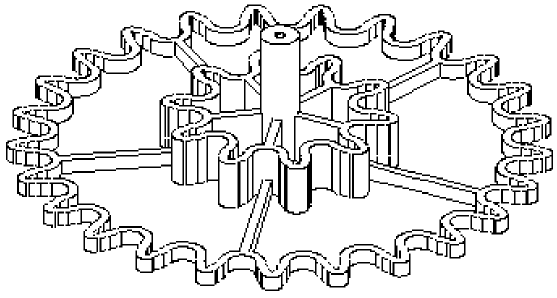
First layer: 3m
Total: 29m

View: Color Print
Show: Options
Feature types

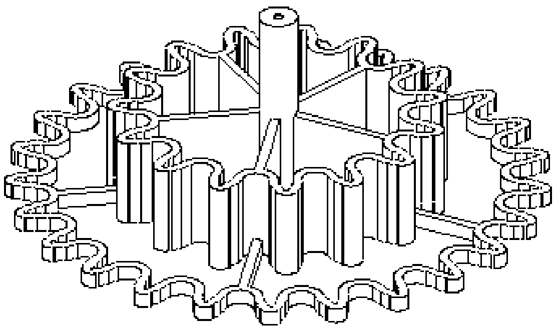
Table 8: Layer changes for spade hands

Here are some diagrams showing the various gears used in the clock. They are spread across several pages to show better detail. The diagrams show gears from the large SP11 clock. The medium SP10 clock gears look nearly identical except for the motor pinion called gear_motor_9 with 9 teeth instead of gear_motor_6 with 6 teeth.

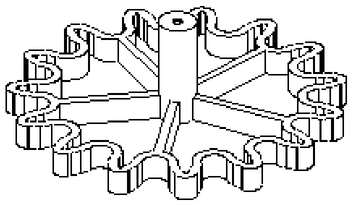
The first page shows the back layer of gears identified with a "b" in the file names.



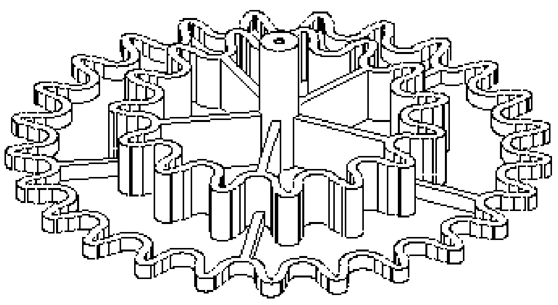
gear0b_25_10



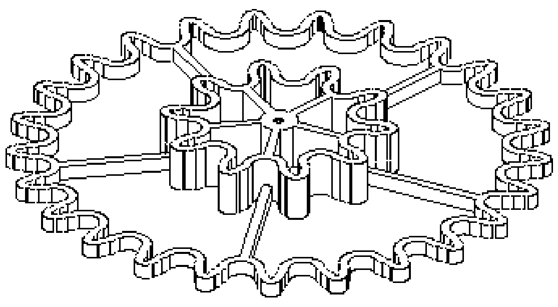
gear1b_25_15



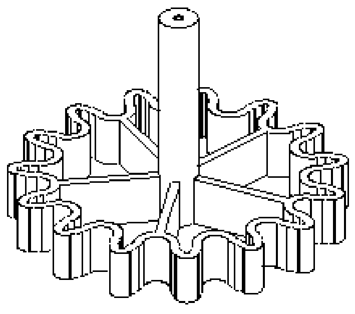
gear3b_15



gear4b_25_15



gear5b_25_10



gear7b_15

Figure 33: Back layer of gears

This page shows half of the primary gears driving the clock hands. The number of teeth in each gear are indicated even if they are cosmetic. For example, gear3_32_20_10 only uses the 32 and 10 tooth part of the gear to drive other gears. The 20 tooth portion is cosmetic and does not actually drive anything. It is still listed in the gear name to help with identification.

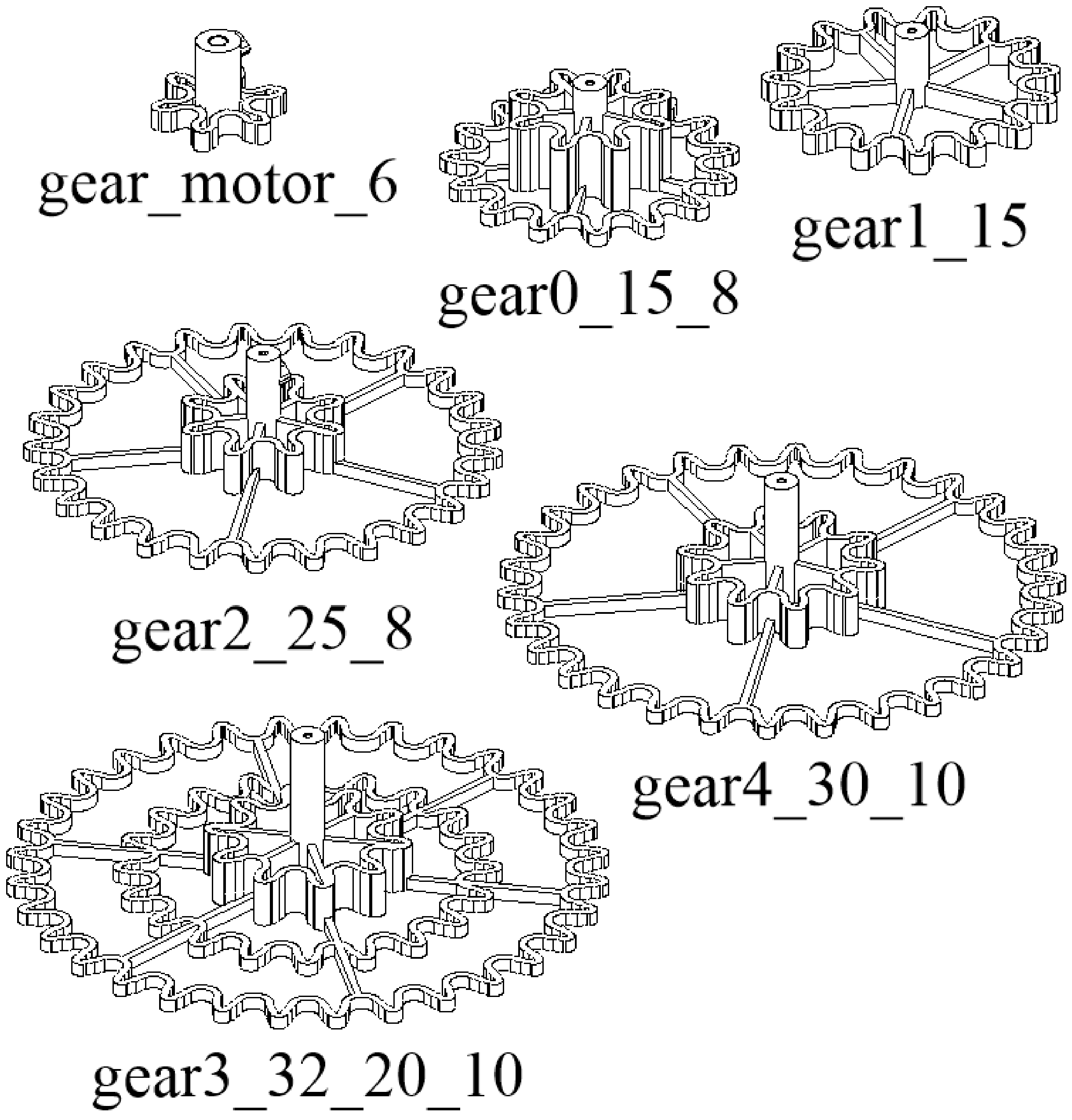


Figure 34: Gears 0 to 4

The page shows the remaining gears between gear 5 and gear 8.

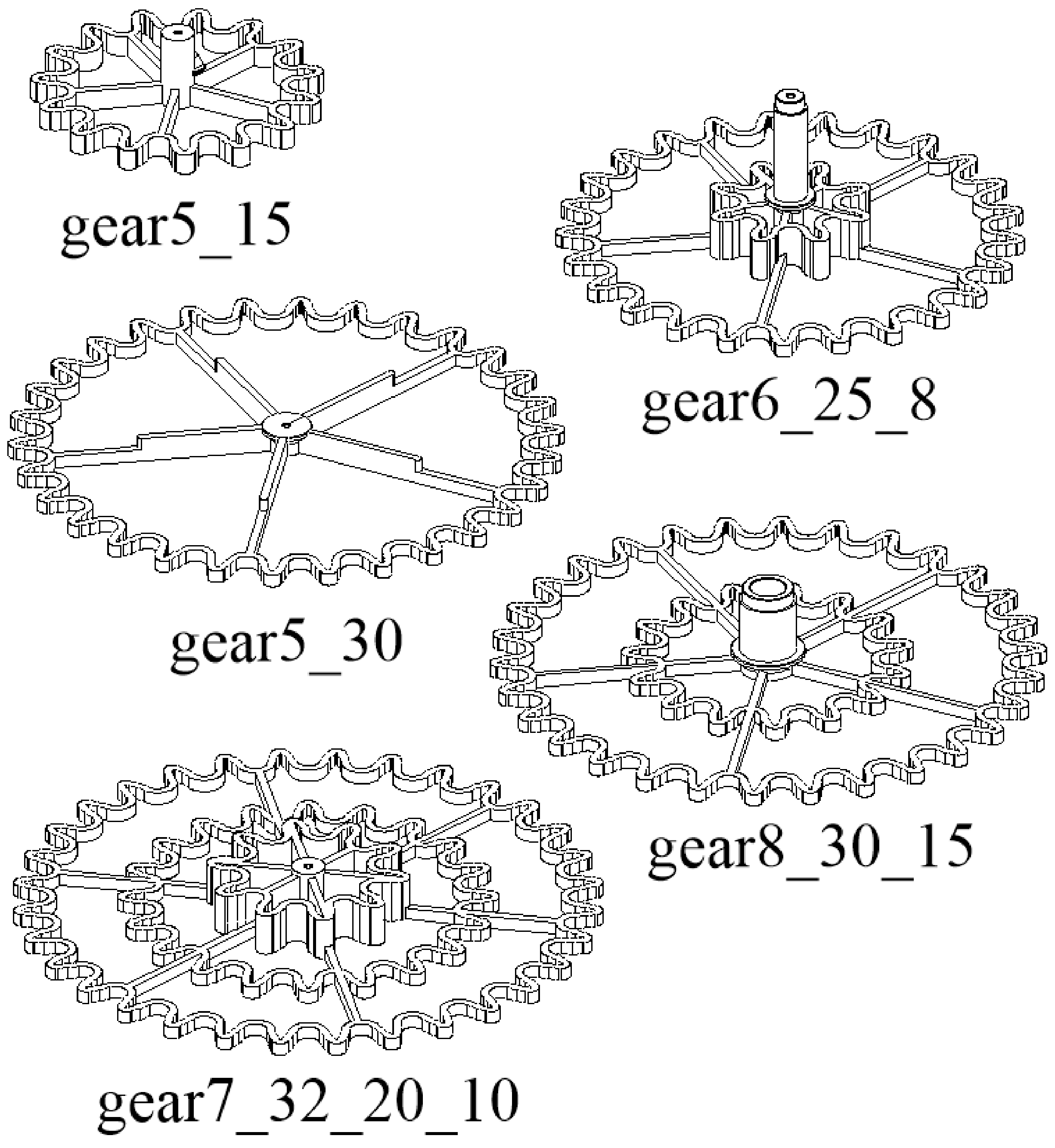


Figure 35: Gears 5 to 8

Building the Clock

Start by printing all the parts as described earlier. A few components can be pre-assembled before placing them in the clock.

Component Pre-Assembly

Gear 2 is the second hand gear rotating once per minute. An M3 set screw holds the arbor tight on the gear so the second hand will rotate when gear2_25_8 rotates. It doesn't need to be very tight, just enough to hold the position. The shaft should extend 1.05" (27mm) below the bottom of the gear. The ideal screw size is M3x8mm, but M3x6mm or M3x10mm screws can also be used.

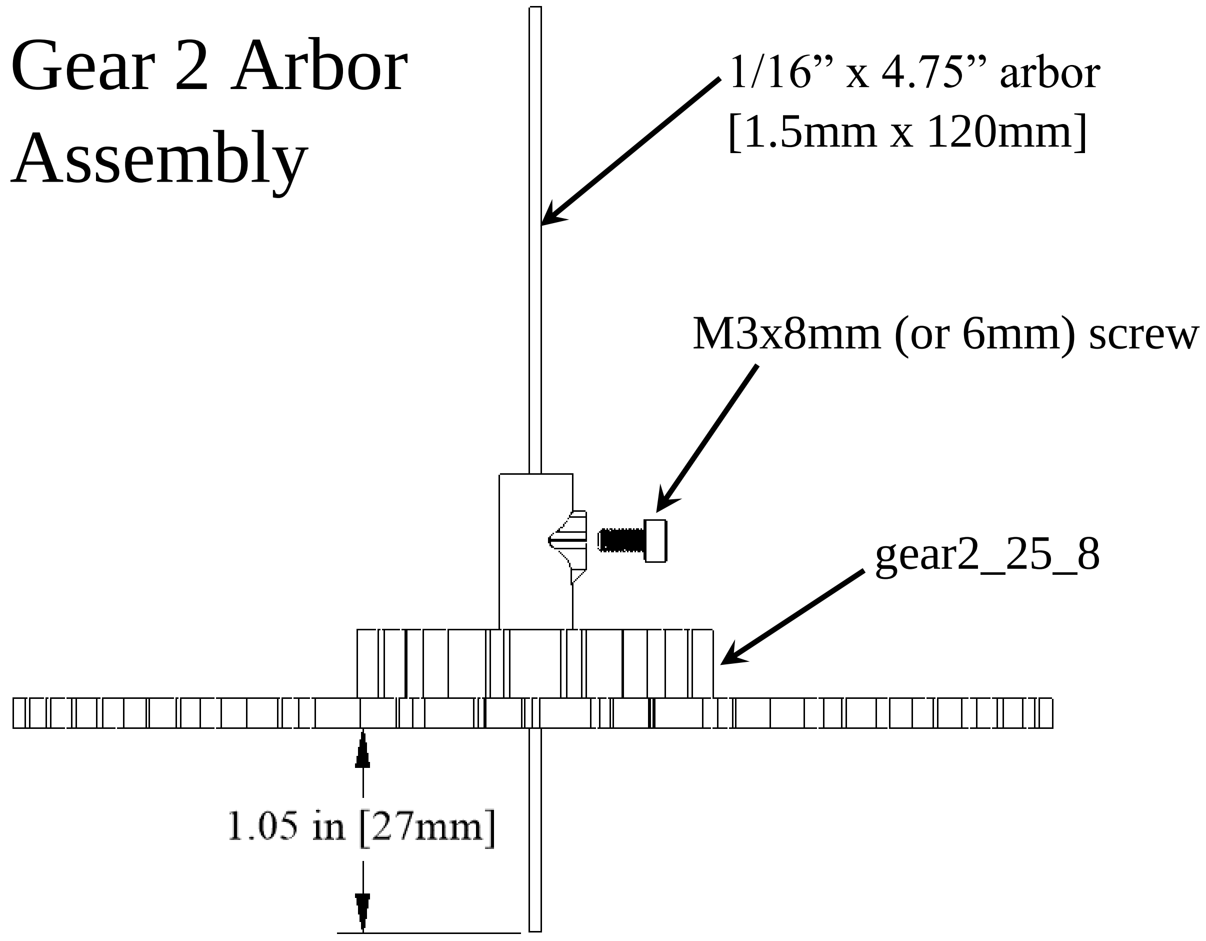


Figure 36 Gear 2 shaft assembly

The gear 5 assembly incorporates a friction clutch to hold when the clock is running and slip when setting the time. Gear5_15 and the shaft collar are tight on the arbor and gear5_30 is allowed to rotate. The pen spring provides a slight amount of pressure to hold gear5_30 steady when the clock is running.

Add the 1/16" (1.6mm) shaft collar to the arbor with 1.65" (42mm) 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_30, and gear5_15 spring to the arbor. Position gear5_15 with 0.3" (8mm) of the arbor exposed. Secure it with an M3x8mm screw. The total thickness of the stack should be around 2.05" (52mm). Gear5_30 should be able to rotate independently from gear5_15 with a slight resistance from the friction clutch.

Here is the completed assembly.

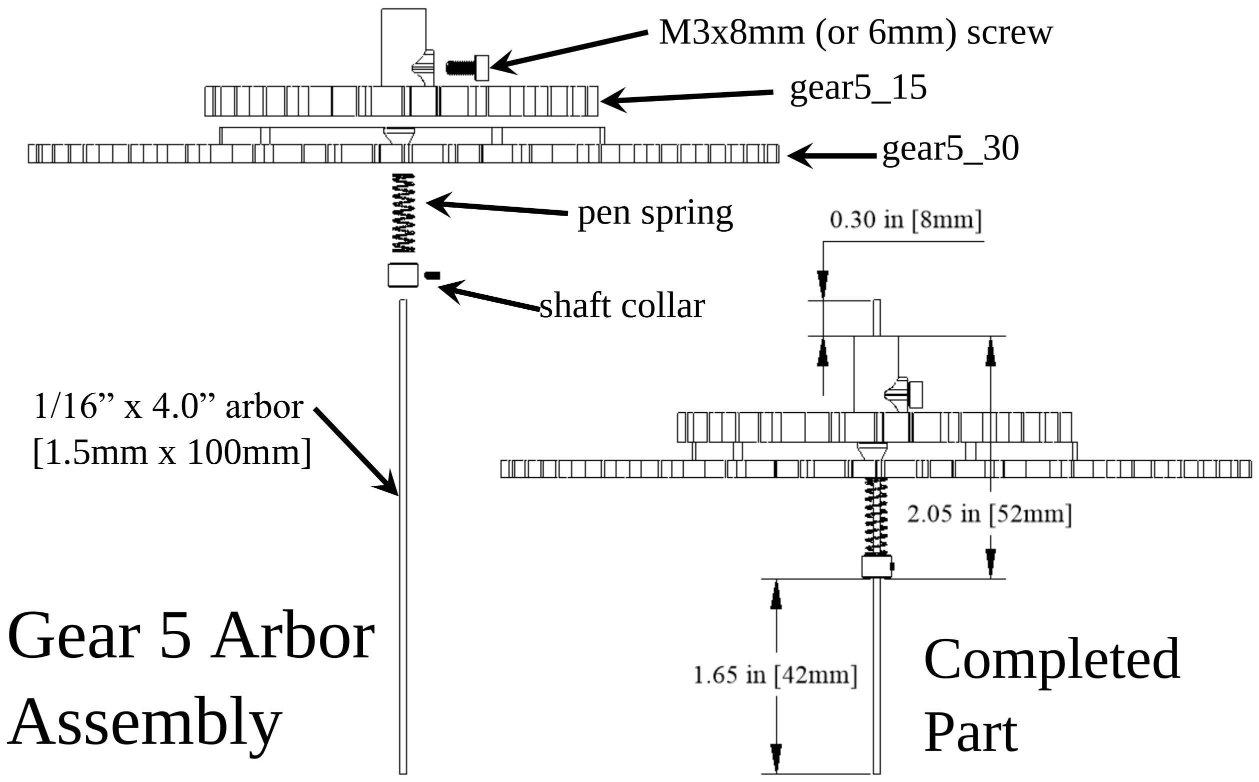


Figure 37: Gear 5 arbor assembly

Medium Frame Assembly

The frame is different between the two clock sizes, so there is a separate section for each clock. Skip to the next section if you are building the large clock.

The largest component of the medium clock frame is 176x154mm and can be printed on a Prusa Mini. The frame legs are always separate components that need to be attached using pins and screws. Four pieces of 1.5mm by 75mm (or 1/16" by 3") music wire are used as pins. The holes are expected to be tight. They may need to be drilled for the music wire to fit.

Insert the music wire into the legs and press them into the back frame. Notice tapered portion at the bottom end of the legs needs to be a the top. A #6x3/4" (or M3x20mm, or M3.5x20mm) wood screw secures each side. Apply slight pressure if needed to ensure that the completed assembly is as flat as possible.

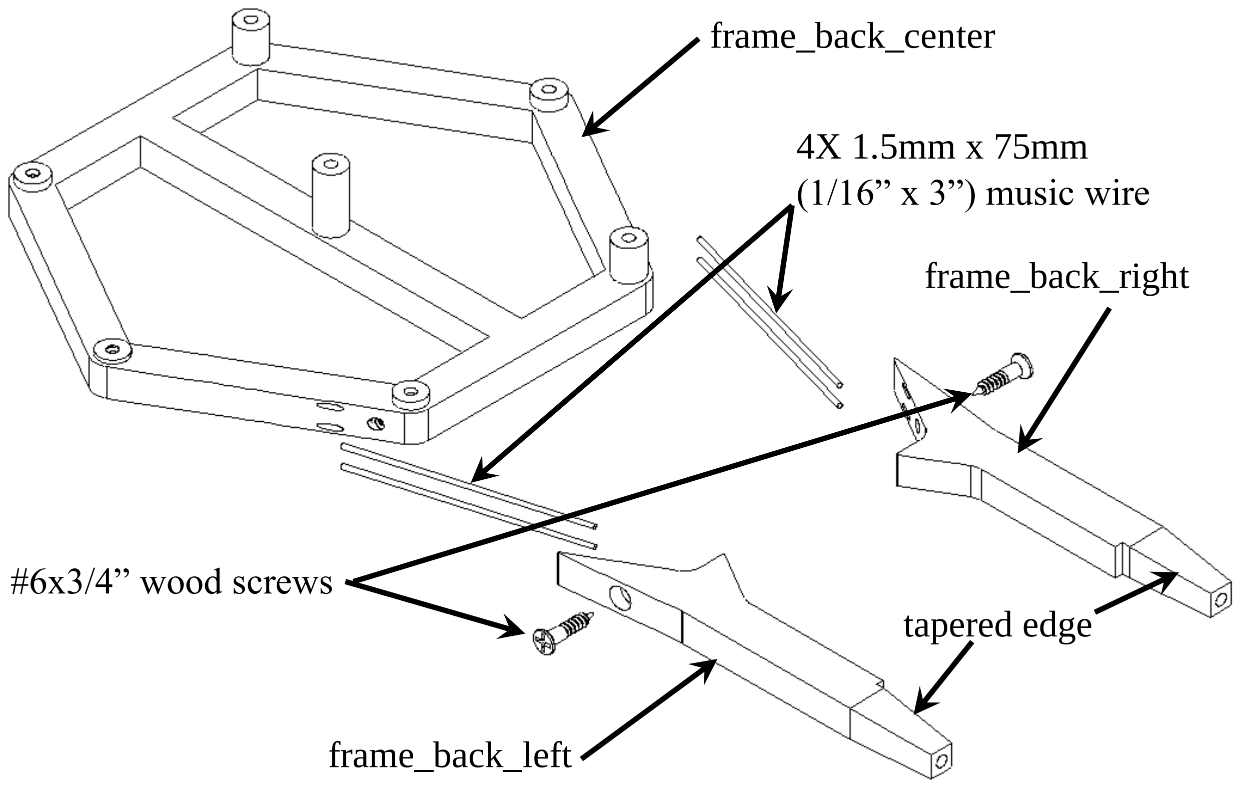


Figure 38: SP10 back frame assembly

Legs are attached to the front dial using the same procedure as the back frame. Four pieces of 1.5mm by 75mm (or 1/16” by 3”) music wire are used as pins. The holes are expected to be tight. They may need to be drilled for the music wire to fit.

Insert the music wire into the legs and press them into the top dial frame. The left and right legs are positioned so the holes for the arbors are at the bottom side. A #6x3/4” (or M3x20mm, or M3.5x20mm) wood screw secures each side. Apply slight pressure if needed to ensure that the completed assembly is as flat as possible.

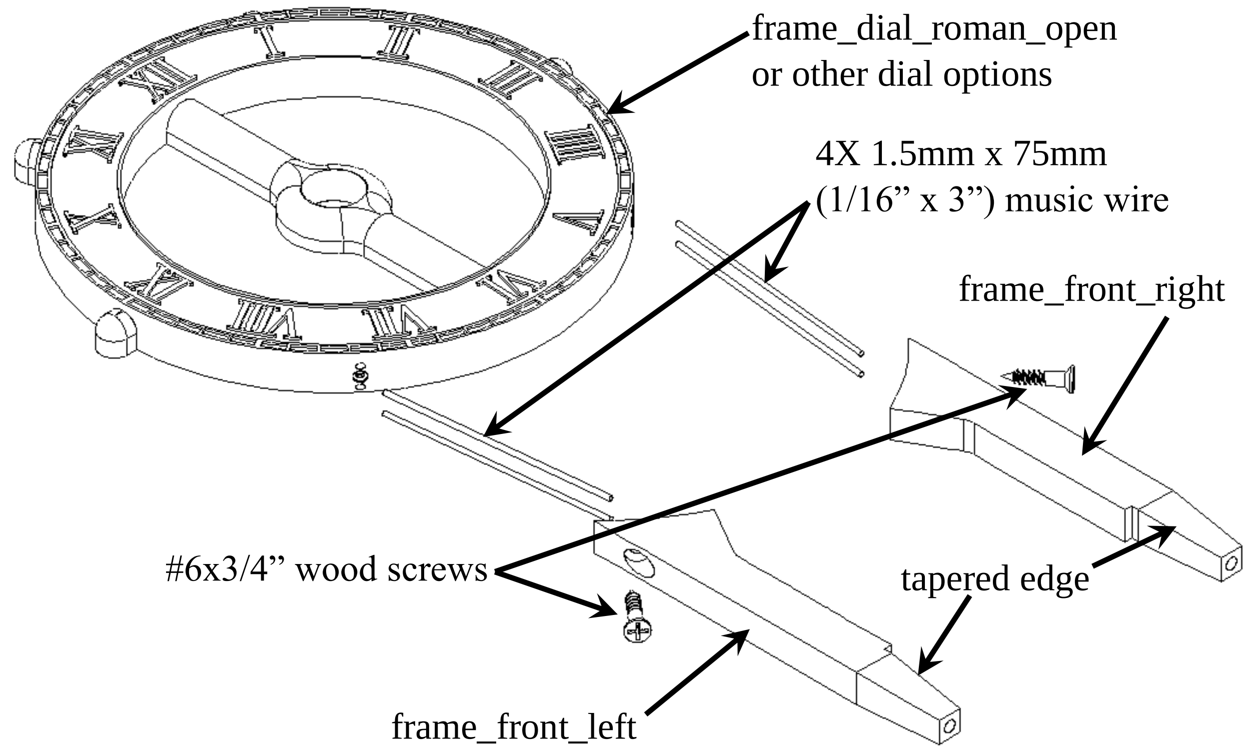


Figure 39: SP10 front frame assembly

Large Frame Assembly

Skip this section if you are building the medium clock.

The “split” version of the frame needs some parts glued together to make up the full width component.

The side tabs are glued in place using 3mm by 25mm (1”) threaded rod or dowels. A cut off M3 or #4-40 screw will work great and have plenty of surface area for the glue. A small wooden dowel or metal pin will also work.

Place the components on a flat surface and glue the parts together making sure that the parts are aligned straight and flat. Notice that frame_back_split_right is slightly taller than frame_back_split_left. Epoxy works great, but nearly any strong glue should be acceptable.

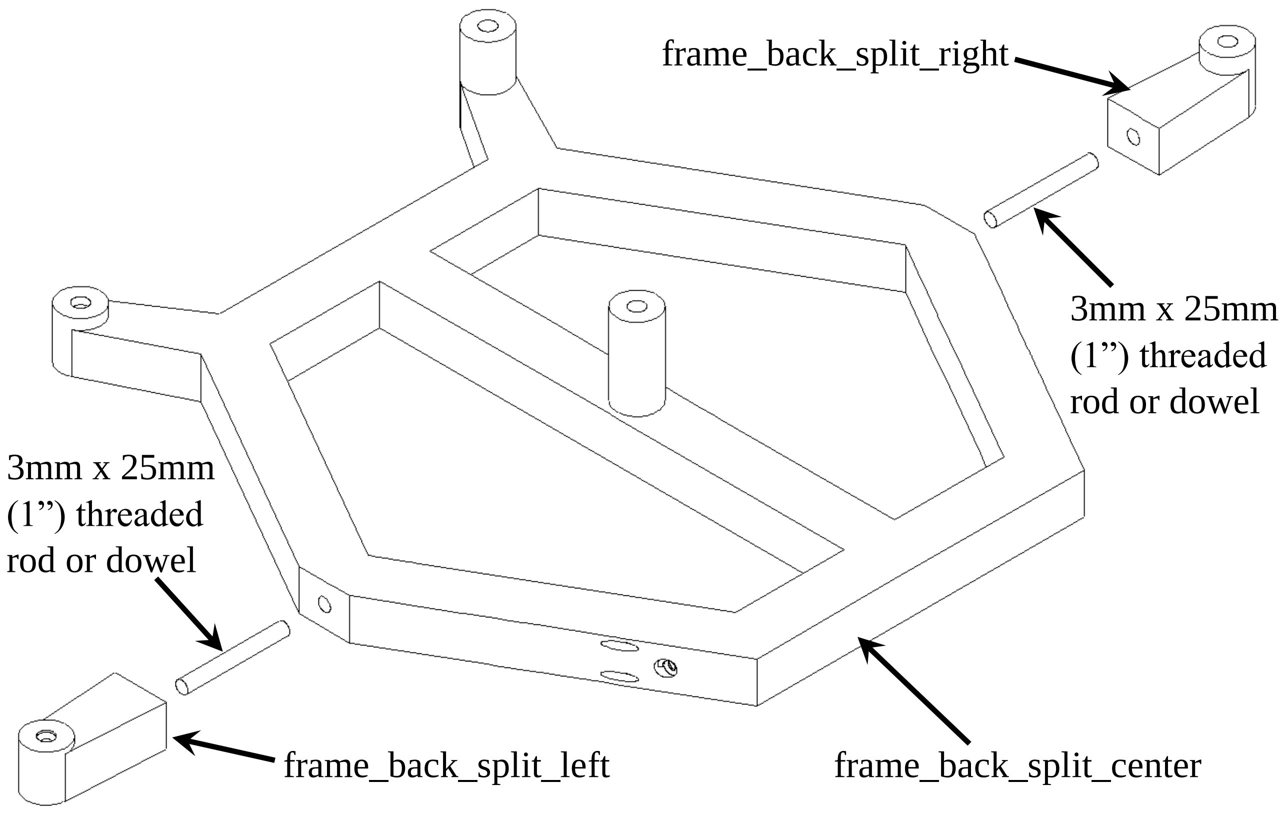

frame_back_split_right
3mm x 25mm (1”) threaded rod or dowel
3mm x 25mm (1”) threaded rod or dowel
frame_back_split_left
frame_back_split_center

Figure 40: SP11 Optional split back frame assembly

The split version of the dial also needs the side tabs glued on. The procedure is the same as the back frame. Use two 3mm x25mm (1”) threaded rod or dowel with epoxy or other strong glue. Select one of the four split dials (roman or numbers, solid or open dial). The side components frame_dial_split_left and frame_dial_split_right are identical, so either one can be placed on either side.

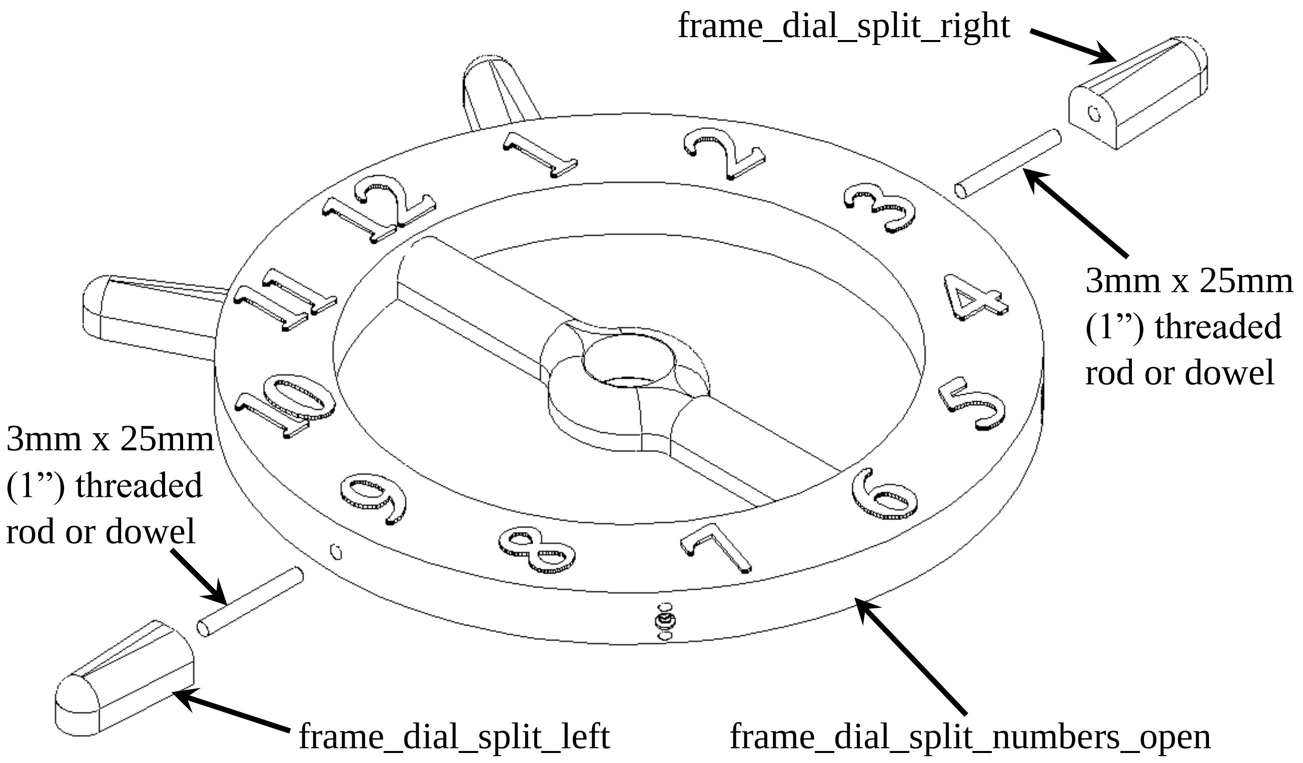

frame_dial_split_right
3mm x 25mm (1”) threaded rod or dowel
3mm x 25mm (1”) threaded rod or dowel
frame_dial_split_left
frame_dial_split_numbers_open

Figure 41: SP11 Optional split front dial assembly

The frame legs need to be attached using music wire pins to build up the complete frame. Four pieces of 1.5mm by 75mm (or 1/16" by 3") music wire are used as pins. The holes are expected to be tight. They may need to be drilled for the music wire to fit.

Insert the music wire into the legs and press them into the back frame. Notice that frame_back_right is taller than frame_back_left. A #6x3/4" (or M3x20mm, or M3.5x20mm) wood screw secures each side. Apply slight pressure if needed to ensure that the completed assembly is as flat as possible.

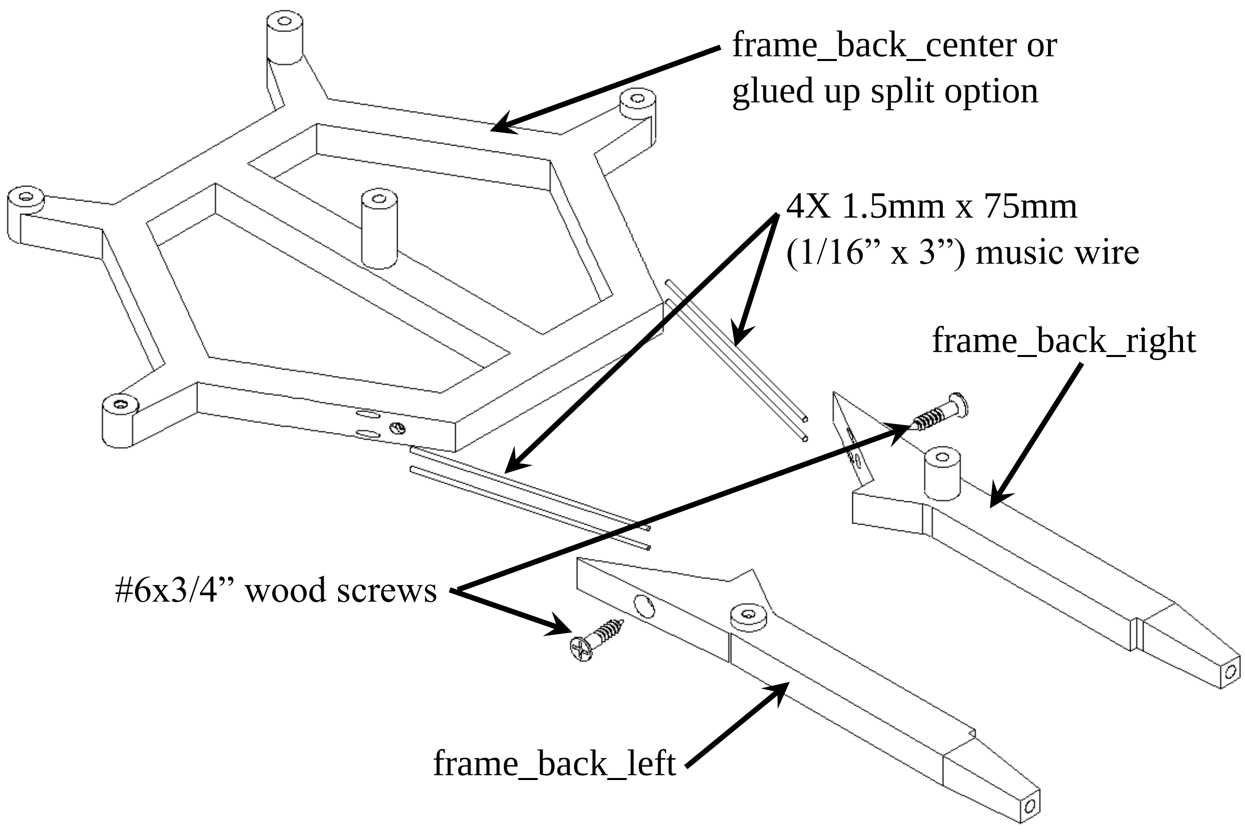

frame_back_center or glued up split option
4X 1.5mm x 75mm (1/16" x 3") music wire
frame_back_right
#6x3/4" wood screws
frame_back_left

Figure 42: SP11 back frame assembly

Legs are attached to the front dial using the same procedure as the back frame. Four pieces of 1.5mm by 75mm (or 1/16” by 3”) music wire are used as pins. The holes are expected to be tight. They may need to be drilled for the music wire to fit.

Insert the music wire into the legs and press them into the top dial frame. The left and right legs are positioned so the holes for the arbors are at the bottom side. A #6x3/4” (or M3x20mm, or M3.5x20mm) wood screw secures each side. Apply slight pressure if needed to ensure that the completed assembly is as flat as possible.

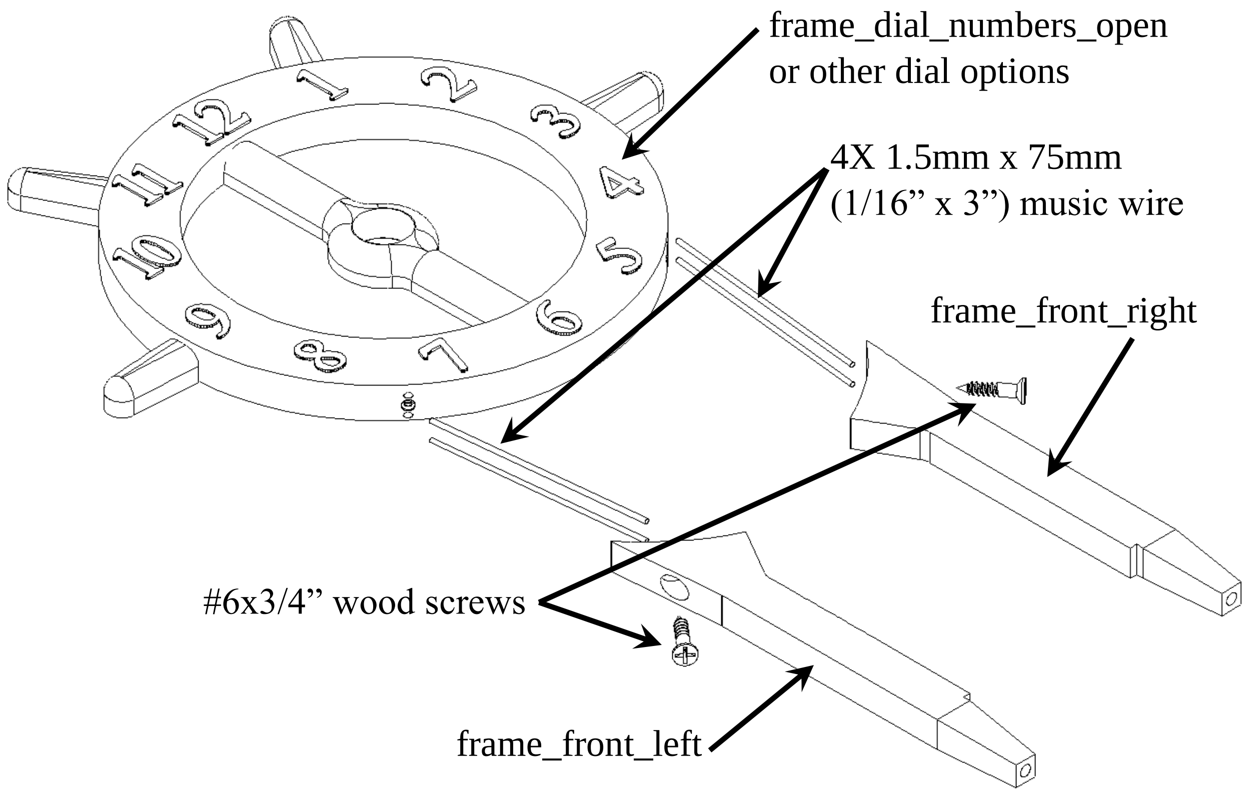


Figure 43: SP11 front frame assembly

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.

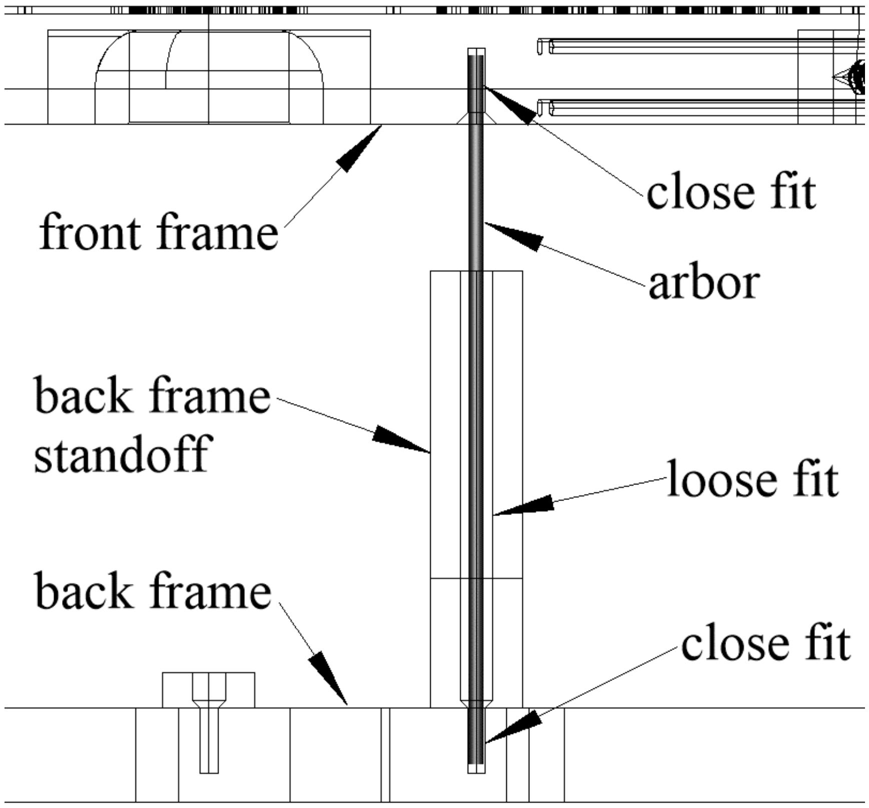


Figure 44 Arbor detail and back frame standoffs

2) Check that the minute hand (gear6_25_8) fits into the hour hand (gear8_30_15) 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 passes 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 given names that help identify their size and order of assembly. This clock has an extra layer of mostly dummy gears around the perimeter to provide dynamic motion. The back layer of dummy gears is identified with a "b" in the file name.

For example, gear5b_25_10 is a back layer gear at position 5 with a 25 tooth gear and a 10 tooth pinion. The diagram below shows the positions of the gears in the clock. All the gears with a "b" in the name get placed first, followed by the remaining gears in a mostly sequential order.

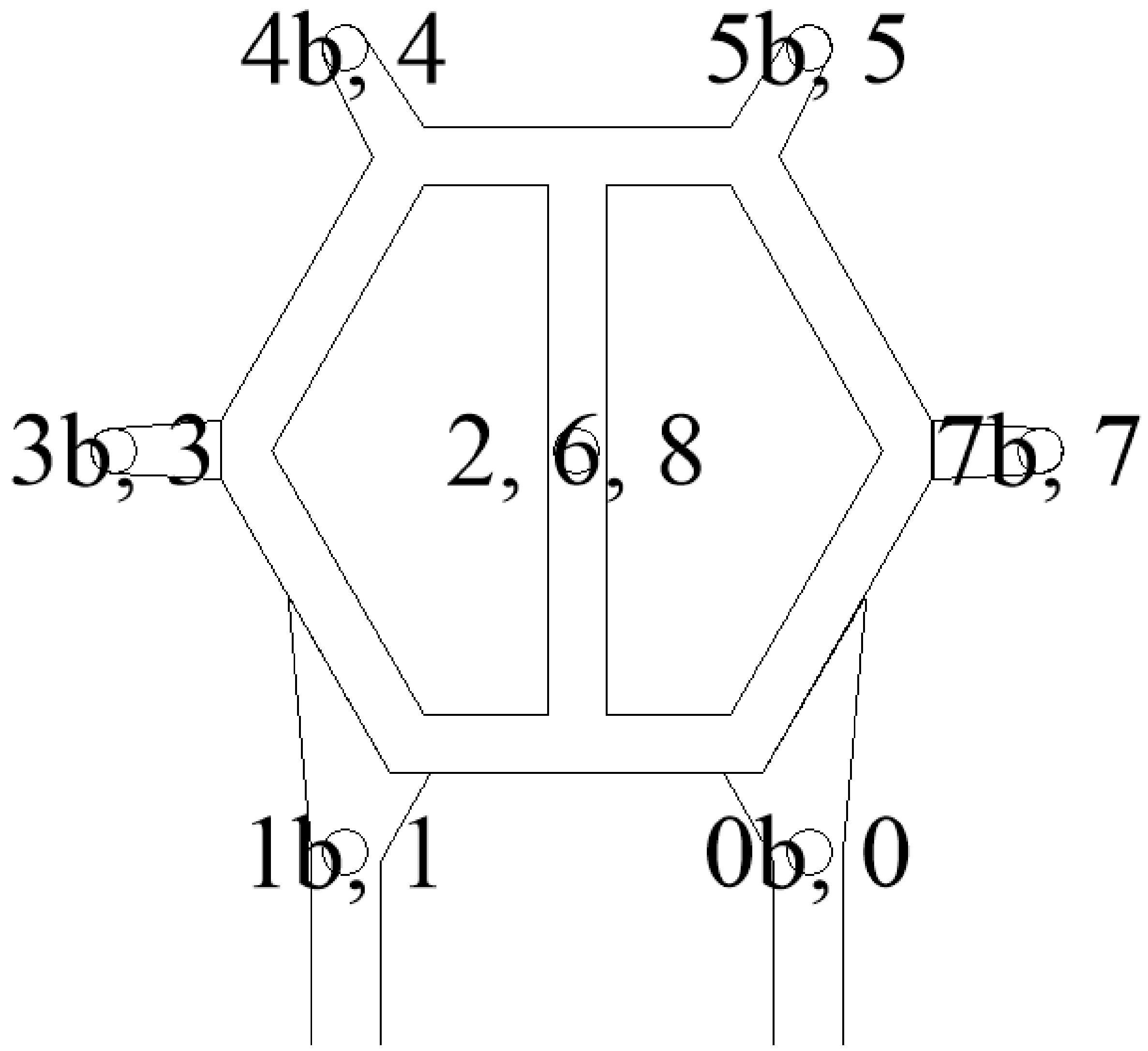


Figure 45: Gear position template

Here are the back layer gears in the approximate positions that they reside in the clock. Gear2_25_8 is also included as the lowest gear on the central arbor.

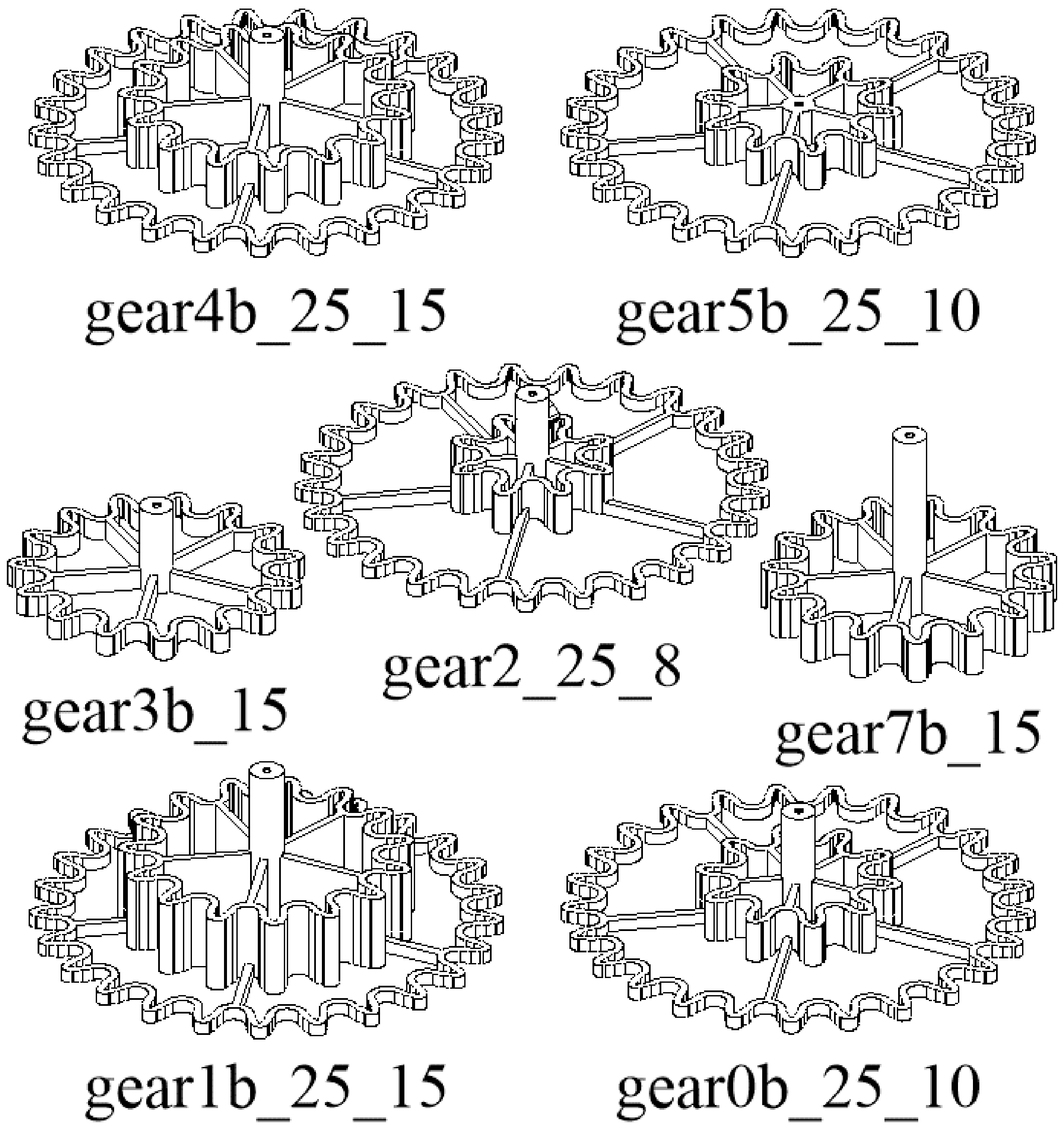


Figure 46: Back layer gears in their approximate positions

These are the remaining top layer gears in their approximate positions. Gear 5_30 and gear5_15 will stack together on the gear 5 arbor. Gear6_25_8 and gear8_30_15 stack on the central arbor.

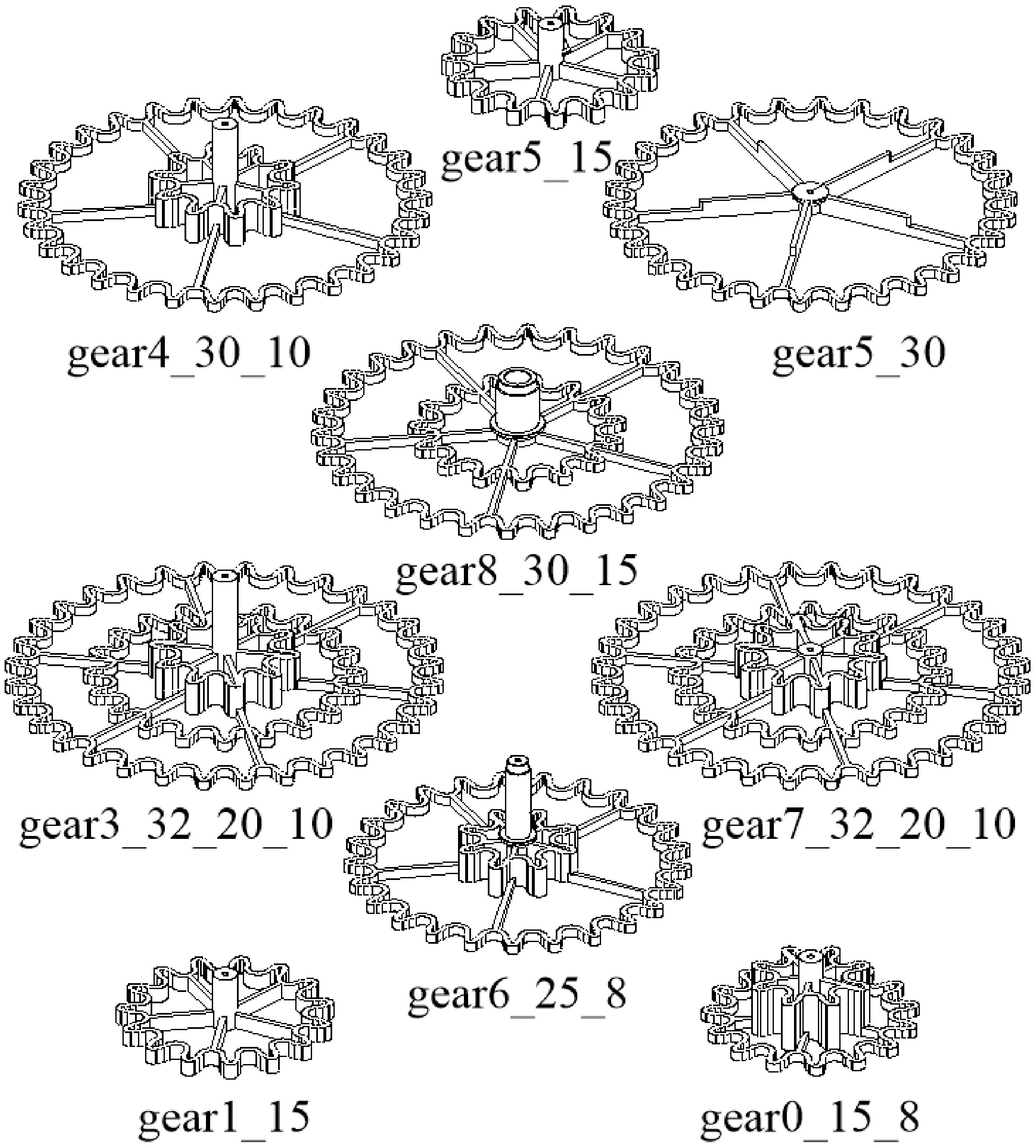


Figure 47: Top layer gears in their approximate positions

The back layer of gears needs to be placed first. These gears need to mesh with a specific alignment to prevent binding. Add 3.5" (90mm) arbors into the three left side holes. Place the three gears shown below with the spokes oriented as shown. The spokes on gear1_25_15 and gear4b_25_15 should be pointed together in a nearly straight line before adding gear3b_15.

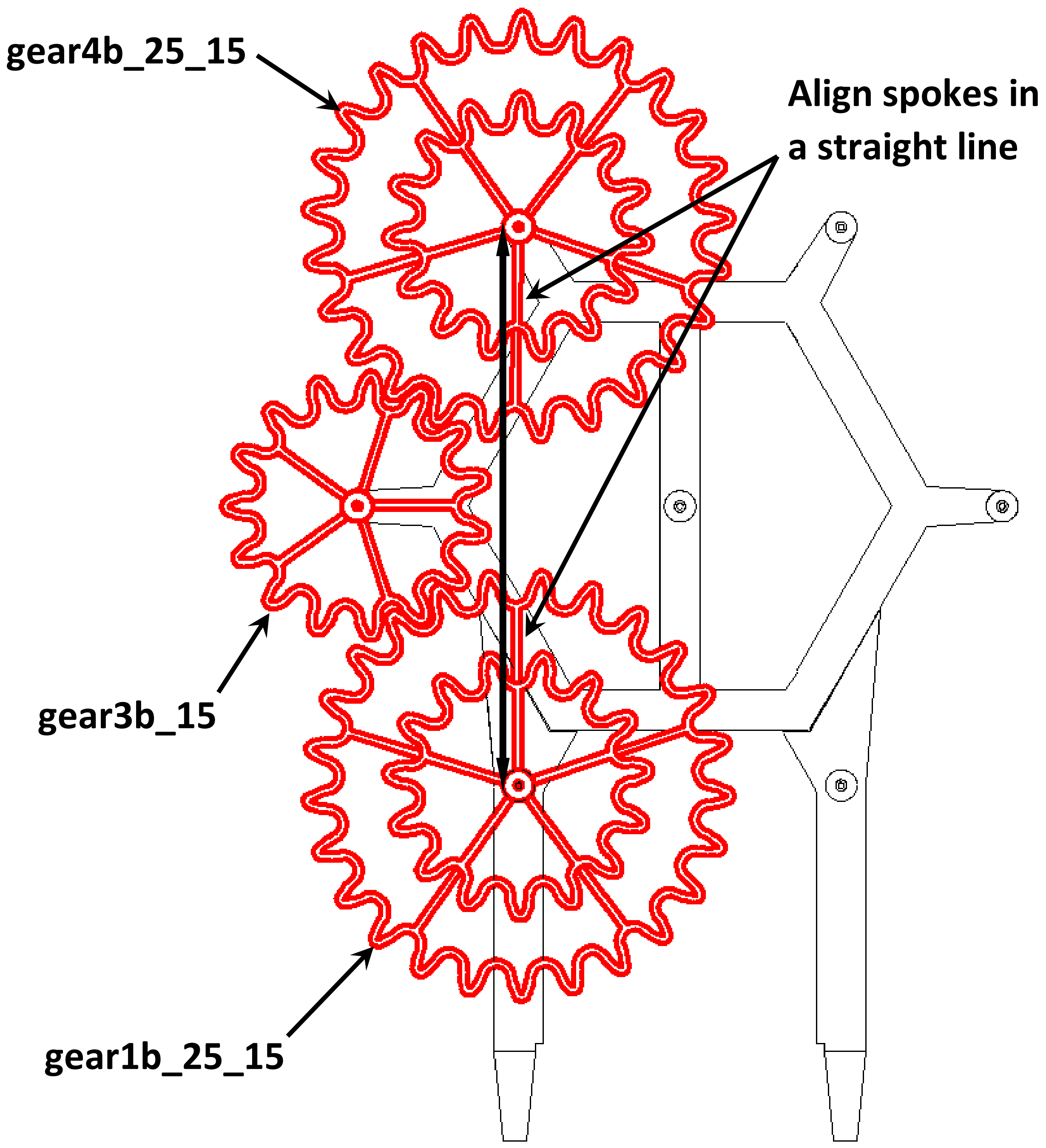


Figure 48: Back layer gears, step 1

The remaining back row gears can be placed next. Add 3.5” (90mm) arbors into the lower right and center right positions. The upper right position uses the 4” (100mm) arbor that is part of the gear 5 assembly. Add gear5b_25_10 in the upper right position using the gear5 assembly or a spare length of music wire. Add the remaining gears on the right side in any orientation.

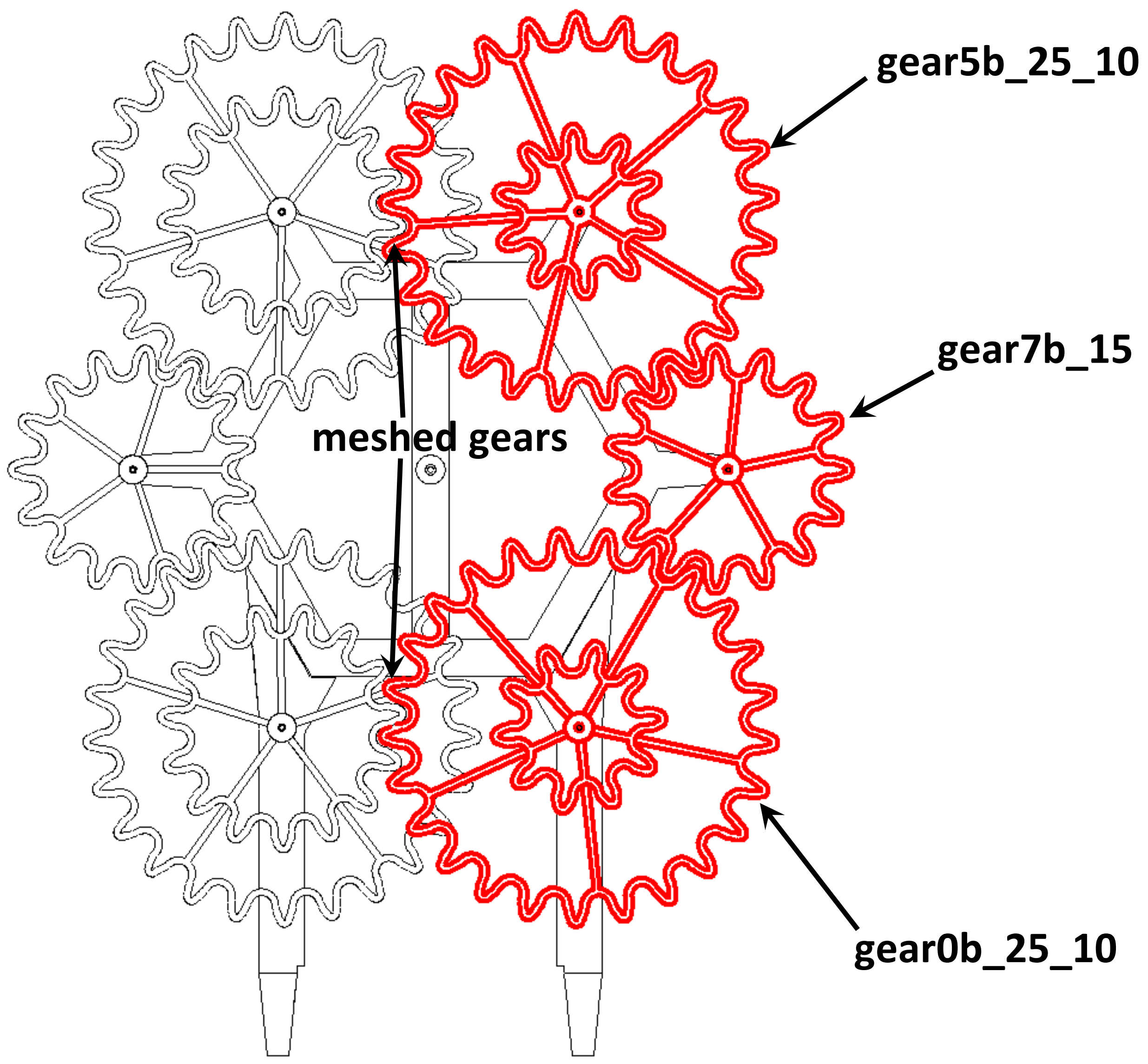


Figure 49: Back layer gears, step 2

Test that the gears rotate smoothly without any binding or noise. Double check the orientation of gear4b_25_15 and gear1b_25_15 if there is any excess friction. Make sure to maintain the orientation of the left side gears for the remaining assembly steps.

The remaining gears will be inserted mostly sequentially starting from the bottom gear2 and working up to the top gear8. There are a lot of gears in this clock and the diagrams may become a bit cluttered. The gear added at each step is highlighted in red.

Start by adding the previously completed gear 2 stack into the center of the clock. It meshes with gear1b_25_15.

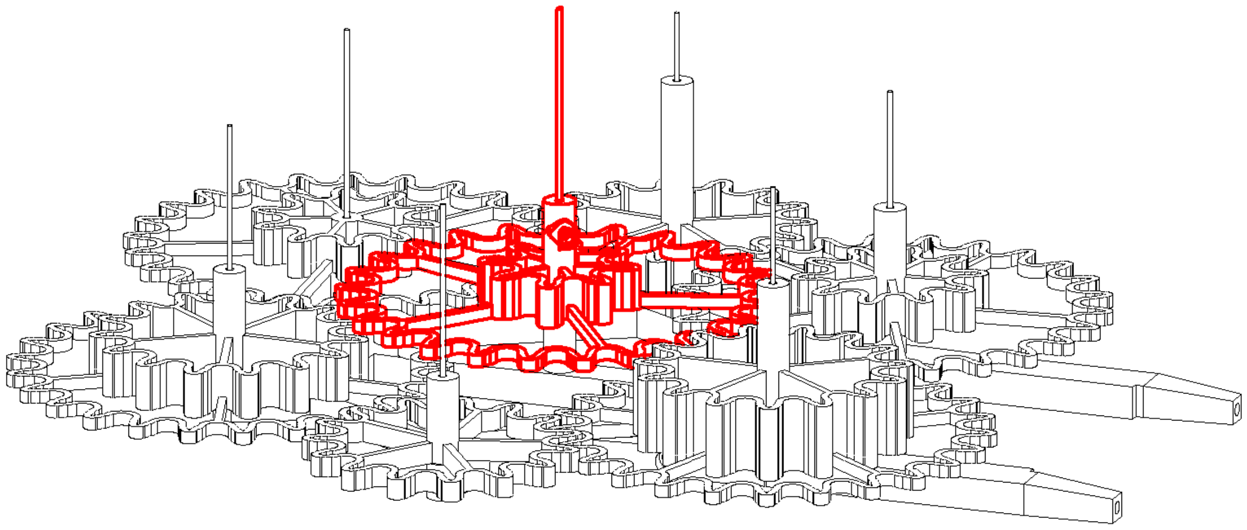


Figure 50: Add gear 2 assembly

Add gear3_32_20_10 to the left center arbor. It meshes with the gear2_25_8 pinion.

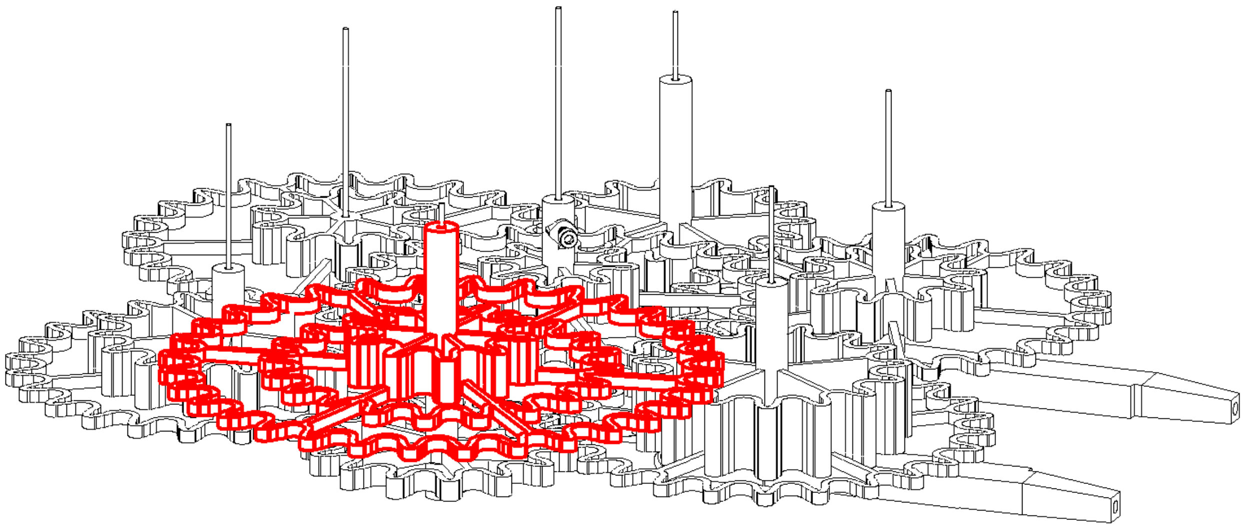


Figure 51: Gear 3

Gear4_30_10 is added to the upper left arbor. It meshes with gear3_32_20_10.

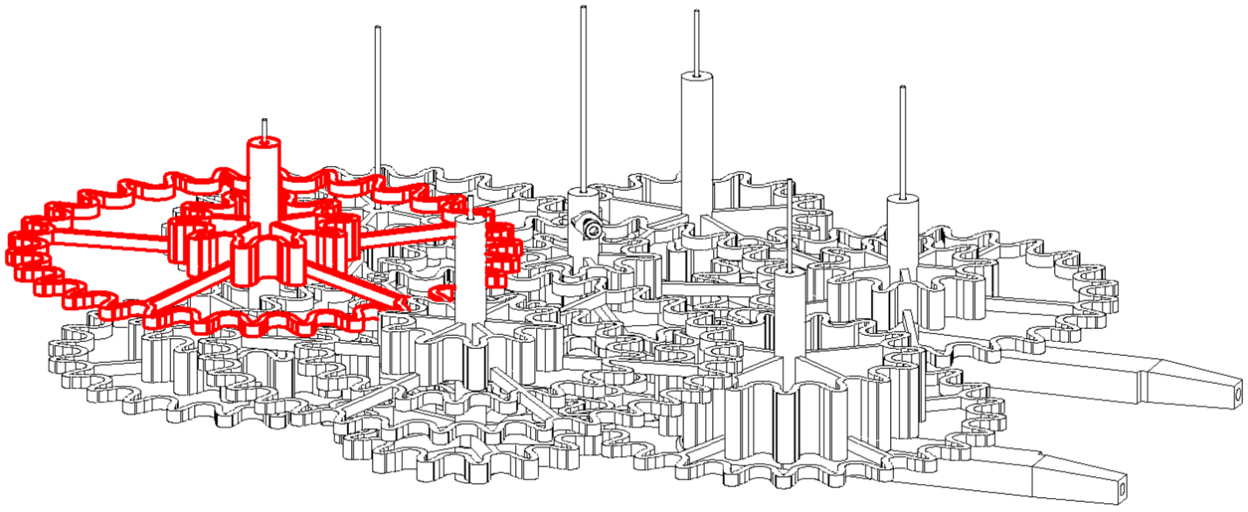


Figure 52: Gear 4

Gears will now be added to the right side of the clock, so the view will be shifted to a new orientation. Add the gear 5 stack into the upper right position. The arbor will pass through the back of the frame. It may help to place the frame onto an empty filament spool so the long gear 5 arbor doesn't scratch the table.

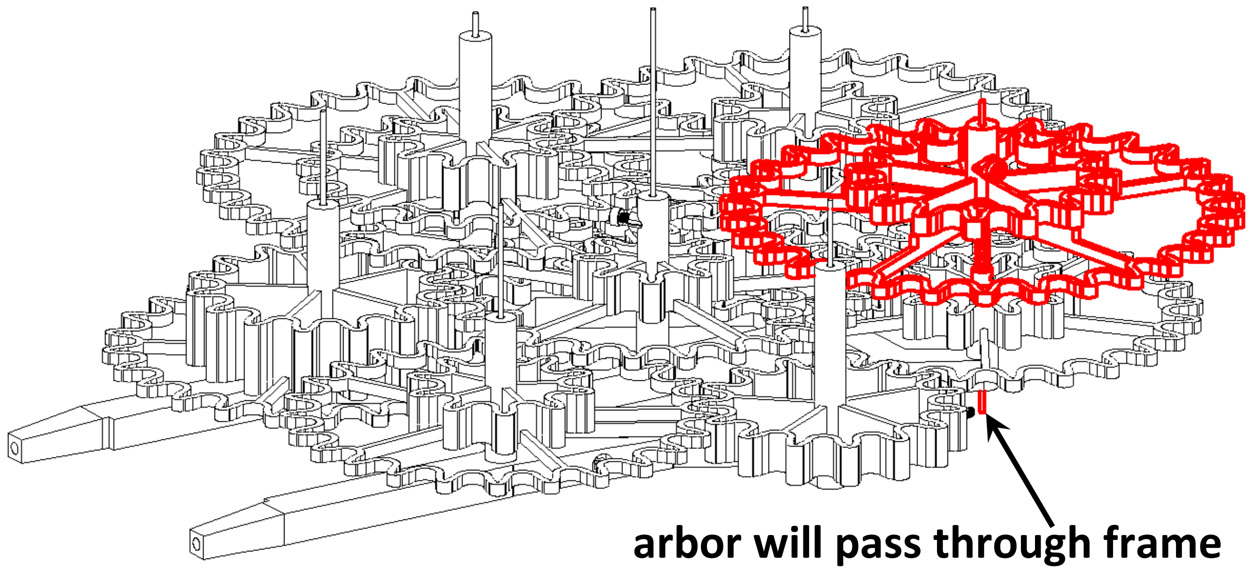


Figure 53: Add gear 5 assembly

The gear 5 arbor sticks through the back of the clock. Add the gear5_knob to the back of the arbor and secure it using an M3x6mm or M3x8mm screw. This is the view from the back of the clock.

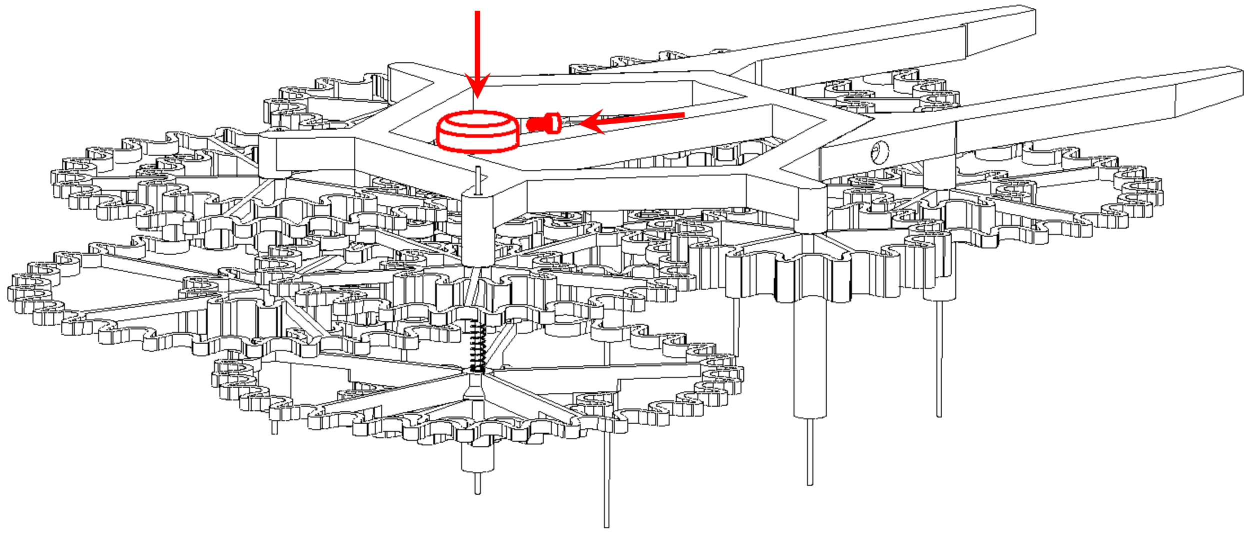


Figure 54: Gear 5 knob added to back of clock

Shifting the orientation back to the top side view. Add gear0_15_8 to the lower right arbor and gear1_15 to the lower left arbor. They don't mesh with anything yet.

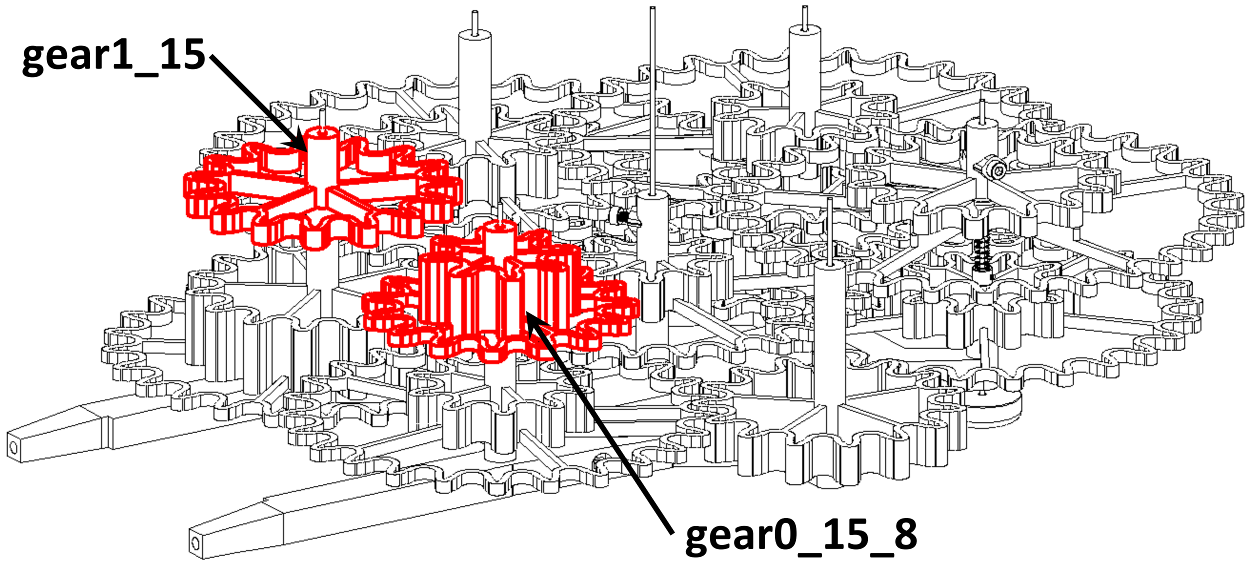


Figure 55: Gear 0 and gear 1

Add gear6_25_8 to the central arbor. It meshes with gear1_15 and gear5_15.

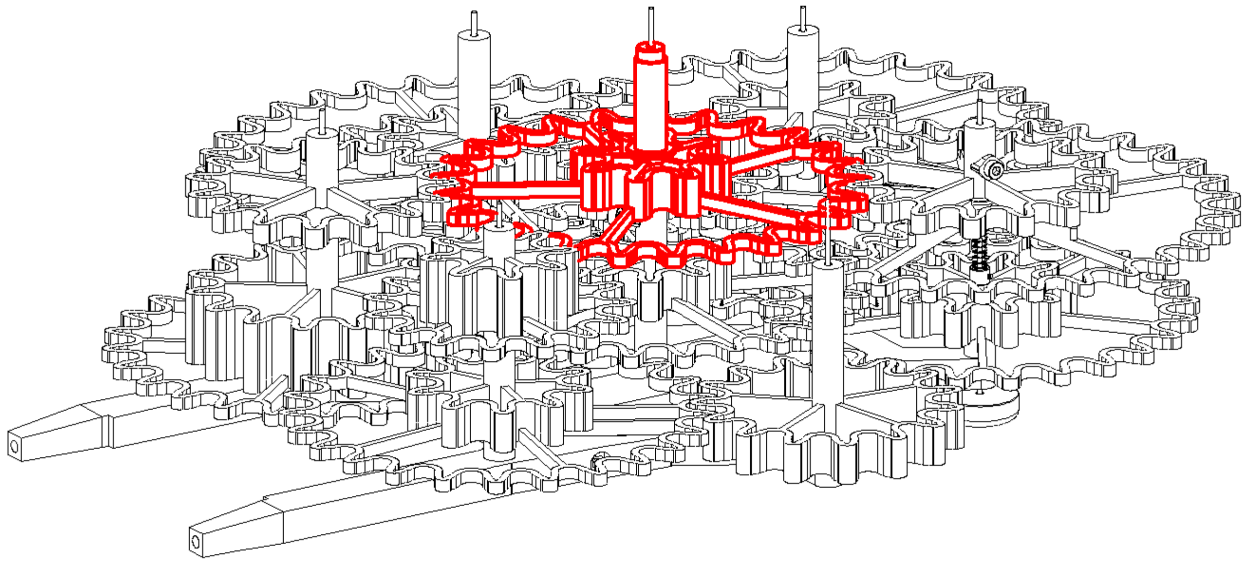


Figure 56: Gear 6

Gear7_32_20_10 gets added to the center right arbor. It meshes with gear0_15_8 and gear6_25_8.

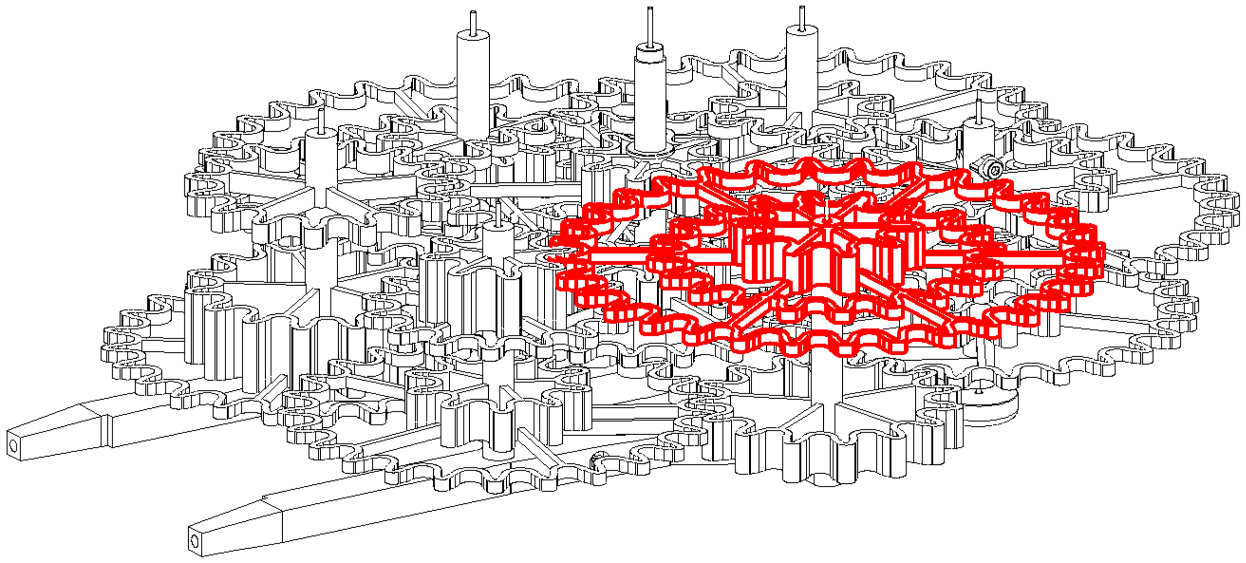


Figure 57: Gear 7

The final gear to add is gear8_30_15 onto the center arbor.

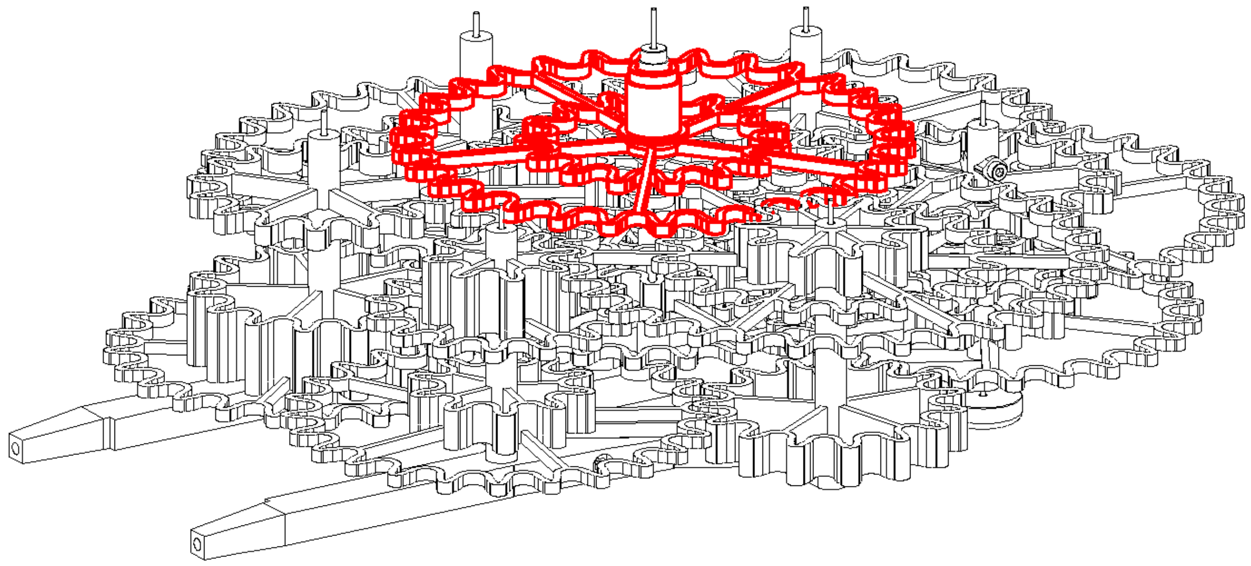


Figure 58: Gear 8

Add the front frame over the center arbor and wiggle the remaining arbors one by one until everything lines up and the frame drops into position. The extended spokes around the dial makes it relatively easy to see when each arbor is lined up properly.

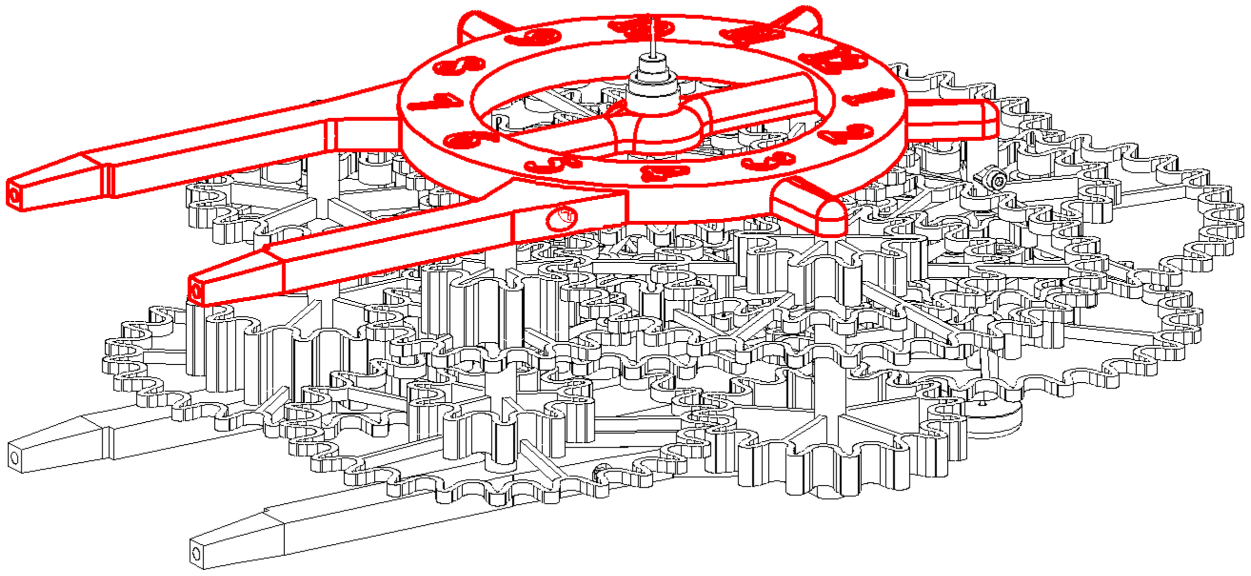


Figure 59: Front dial

Motor Tuning

The final assembly step is to add the motor and base onto the clock. Add the motor into the base_motor_mount and secure it using two M3x8mm screws. Slide the base_motor_cover over the motor and secure it with another M3x8mm screw. This completely encapsulates the motor to help hide it from view.

The medium SP10 clock and large SP11 clock motor assembly is identical except for a different size motor pinion. SP10 uses gear_motor_9 and SP11 uses gear_motor_6. Add the pinion to the stepper motor shaft and center the gear teeth over the small recess in the motor holder. Secure the gear using an M3x8mm screw. It is OK if the stepper motor shaft extends through the gear.

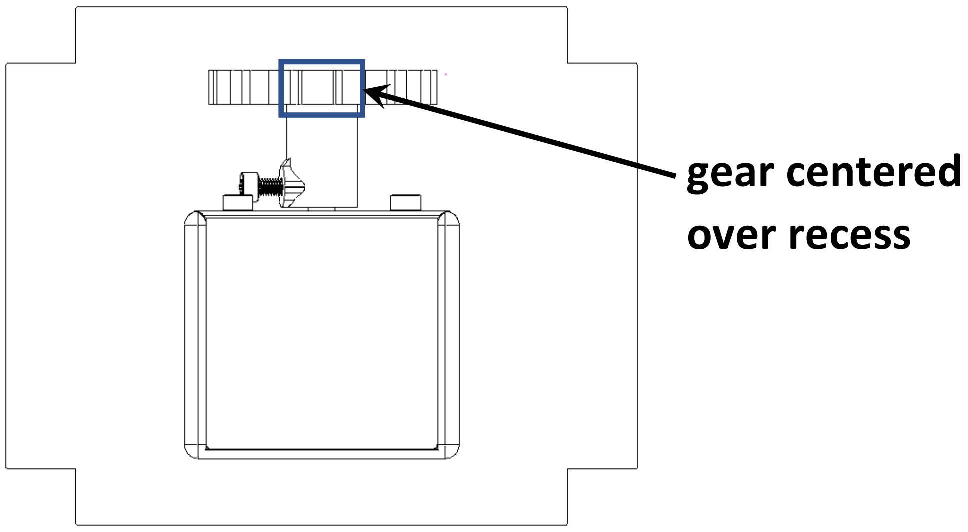


Figure 60: Motor and motor mount

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). Testing will be easier if you place the controller outside the base during the initial assembly. This will make it easier to run the debug tests and make any adjustments. Route the stepper motor cable from the motor, into the base, and out the USB hole. Place the motor mount onto the base and insert the completed upper clock assembly onto the base. The screws to hold the base are optional at this point since you will need to take it apart to add the controller back into the base in a later step.

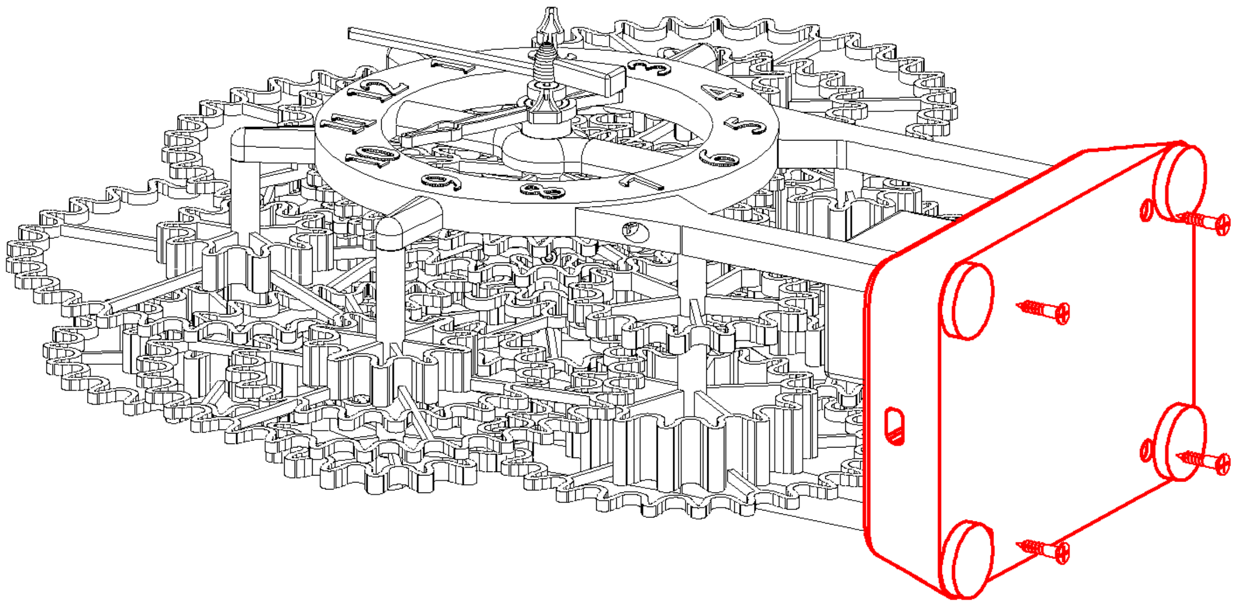


Figure 61: Add the base

Stand the clock upright and connect the stepper motor cable into the previously programmed control module. Insert the appropriate jumpers to set the speed used for your clock. Plug the stepper motor into the 4 pin header on the CNC Shield V4 board or one row of the 8 pin header on the Silent Shield Clock Controller. Plug in the USB cable to provide power and the clock should start moving.

The first step is to adjust the stepper motor current to be as low as possible for the clock to operate. 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.

If the clock runs backwards, then remove USB power and reverse the stepper motor cable on the motor header. If the motor jumps back and forth without rotating, then the cable may need the edit to swap the middle two wires (CNC Shield V4 only). The Silent Shield Clock Controller has a second motor header that can be used instead of the cable wiring swap.

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. The lowest possibly motor power makes it easier to adjust the second hand position.

Final Assembly

After the driver is tested, the clock is ready for final assembly. Remove the base from the clock and move all the components into the base.

Insert the motor control module into the base so the USB plug aligns with the cable opening. Check that the USB cable can plug into the Arduino Nano. Add the appropriate holddown (left, right, or back) to keep the controller in the proper position. The CNC Shield V4 version of the motor controller fits directly into the base. The Silent Shield Clock Controller uses base_retainer_lower and base_retainer_upper to wrap around the controller so it fits snuggly in the base.

Coil up the stepper motor cable so the base can be assembled without pinching the wires. Pay attention to the direction that the stepper motor cable was plugged into the motor header, since it is not keyed. It needs to retain the same orientation as the first test.

Here is what the base and motor assembly would look like with the power on the left side. A base_holddown_(left, right, or back) is used to help keep the motor controller in position. The stepper motor cable is not shown. 1" (25mm) diameter felt pads can be added to the base if desired.

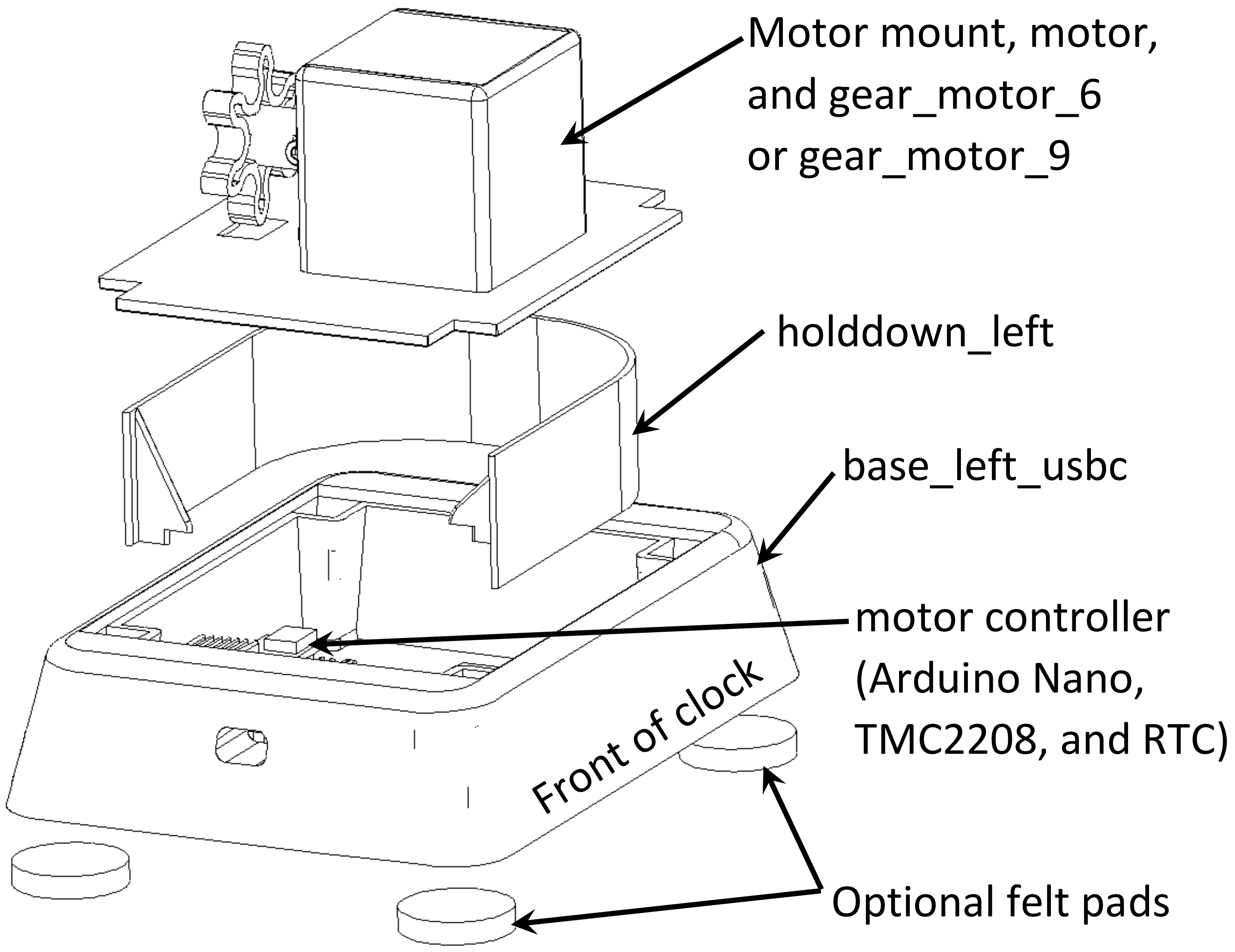

Motor mount, motor, and gear_motor_6 or gear_motor_9
holddown_left
base_left_usbc
motor controller (Arduino Nano, TMC2208, and RTC)
Front of clock
Optional felt pads

Figure 62: 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) Was the motor rotating before it was 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 gears do not bind. Check for friction in the back row of gears, where gear 6 passes through gear 8, and where gear 8 passes 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 if needed.
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 may stall the algorithm. The debug monitor might show the first header line and stop the first time it tries to read the RTC if the RTC is not working properly. Try the debug mode to test the RTC and check if the LED is blinking.
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. Double check that the jumpers are set to the proper speed configuration.
6) Message me with further questions. I will want to know the details of which clock you are building, plus a log of the message displayed in the debug monitor.

Final Notes

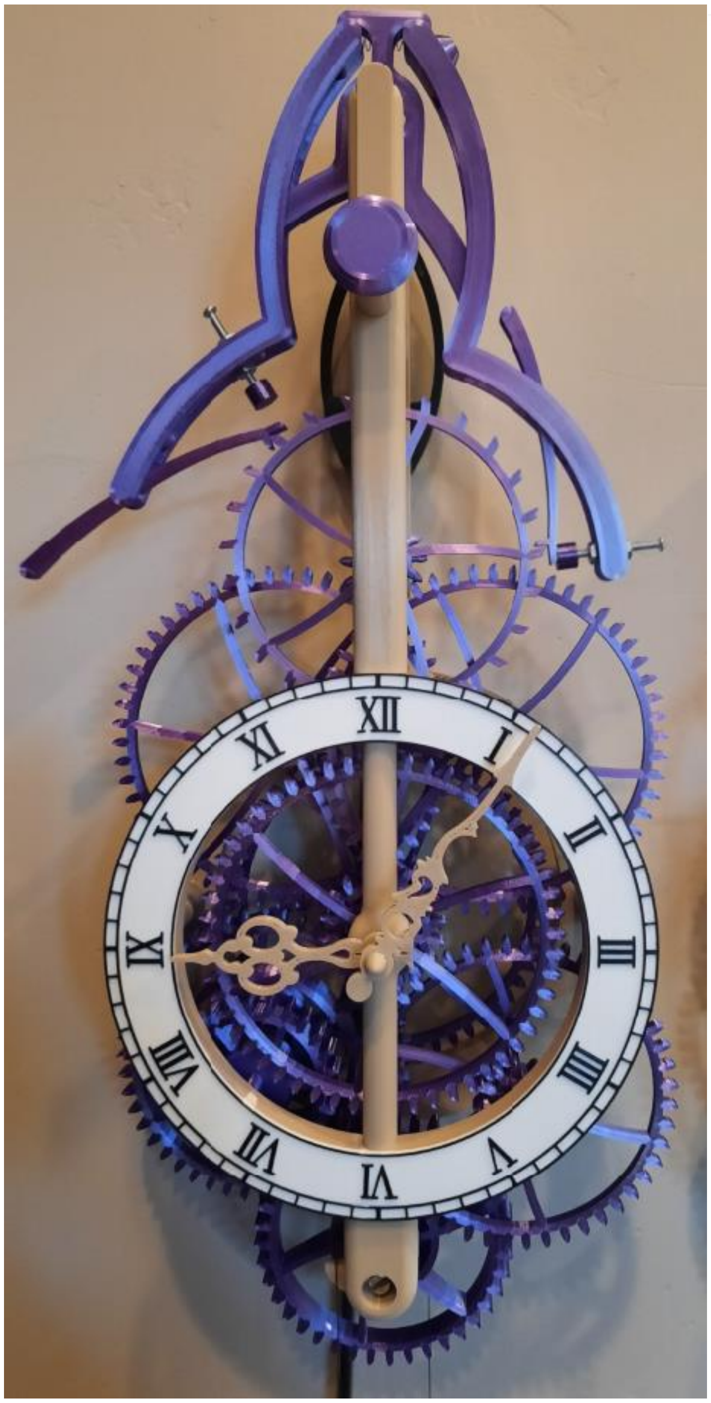
I hope you enjoy building these clocks as much as I enjoyed designing them. The "crazy" gear motion in these clocks looks great. Many additional features make this an extremely functional clock. The gears are nearly silent and the clock maintains great accuracy. The worst case spec is about a minute per year. My first two clocks are holding an accuracy within one second per month. I always get excited to see new designs. The large SP11 clock might become my all-time 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
I created a Facebook group called "3D Printed Clocks" to share images or ask questions. Other Facebook groups with similar interests are "Wooden Geared Clocks" and "Clockmaking using a CNC". My Patreon page is https://www.patreon.com/user/about?u=30981480 if you want to support me further.

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.



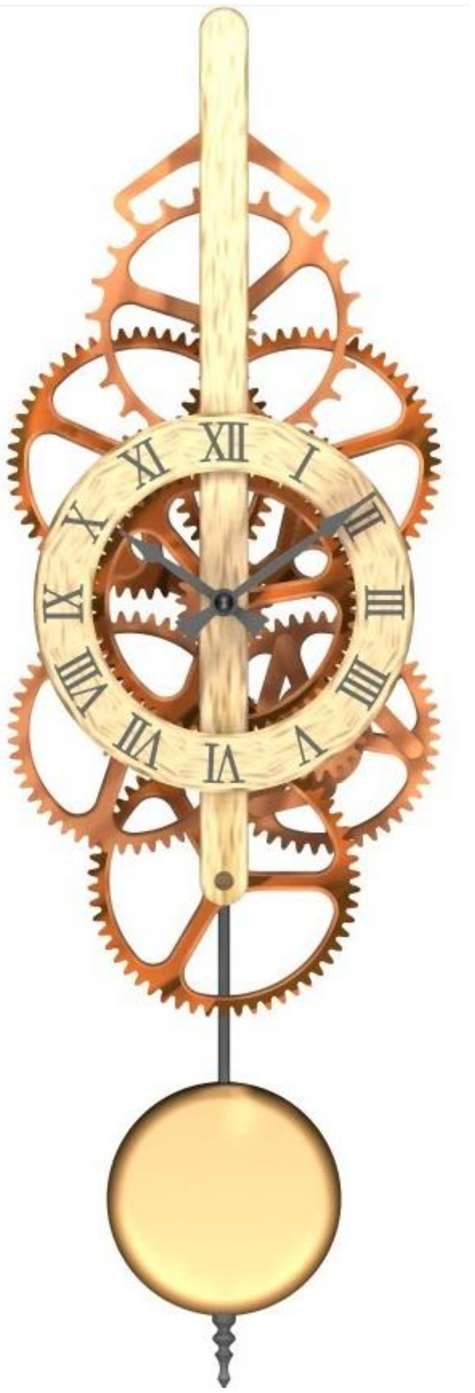


Figure 63: Grasshopper clock and wood clock rendering

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.

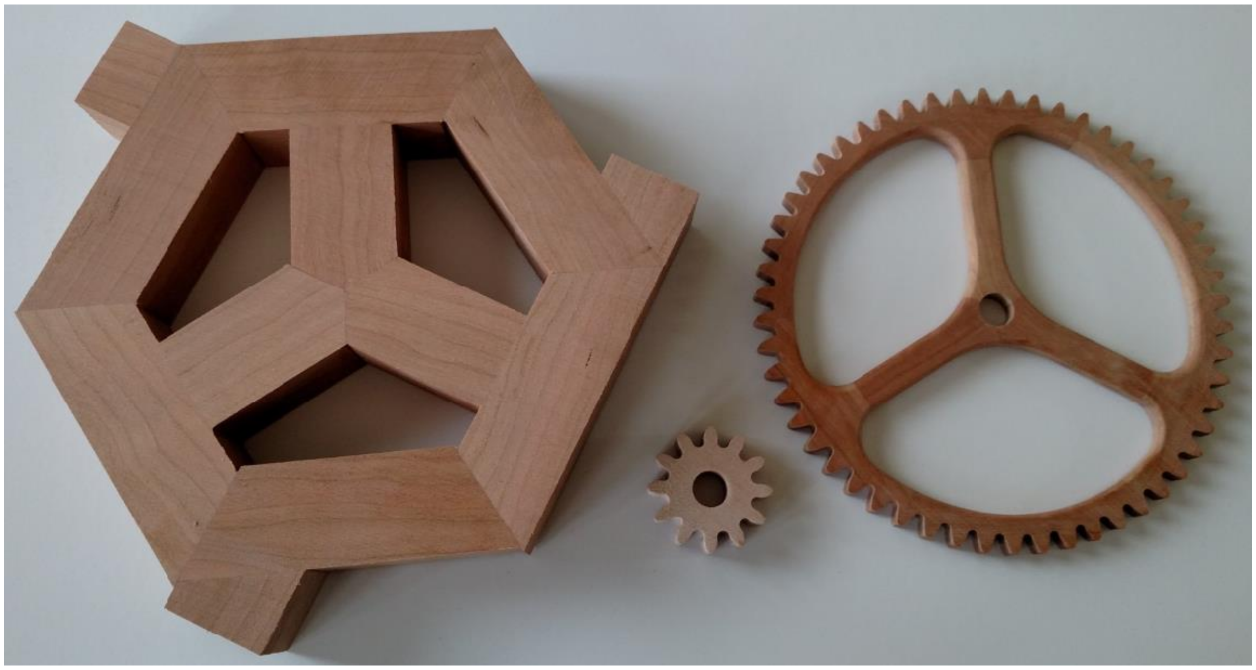


Figure 64: Wooden gear experiments

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 as SP6. This design on the right is SP9.

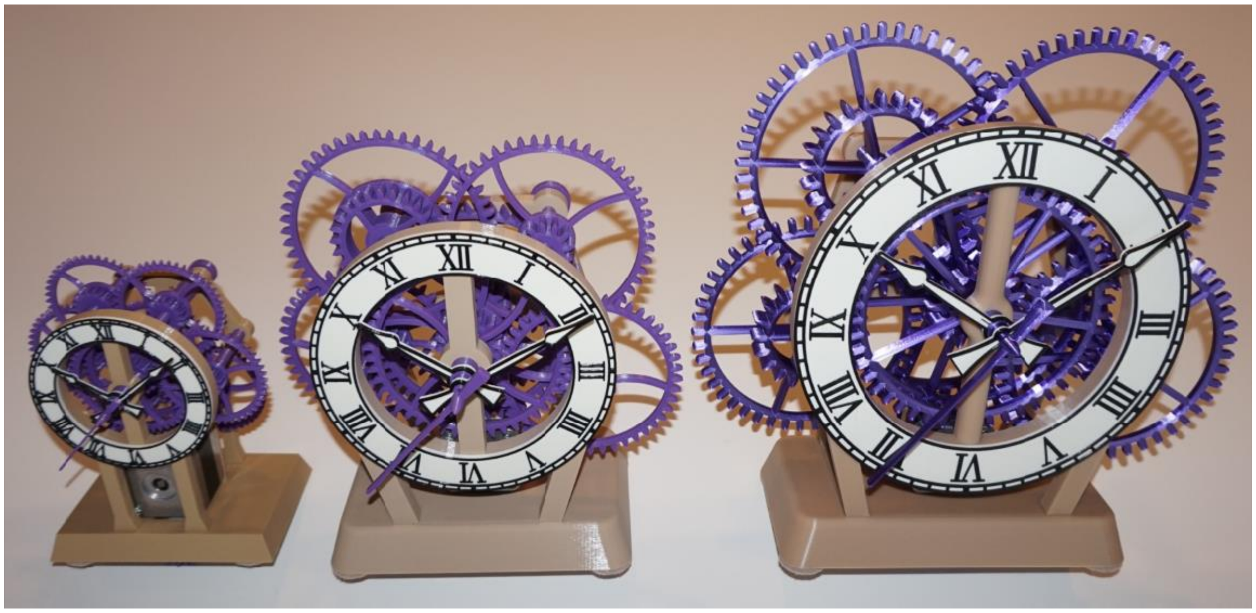


Figure 65: Desk clocks before SP10 and SP11

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

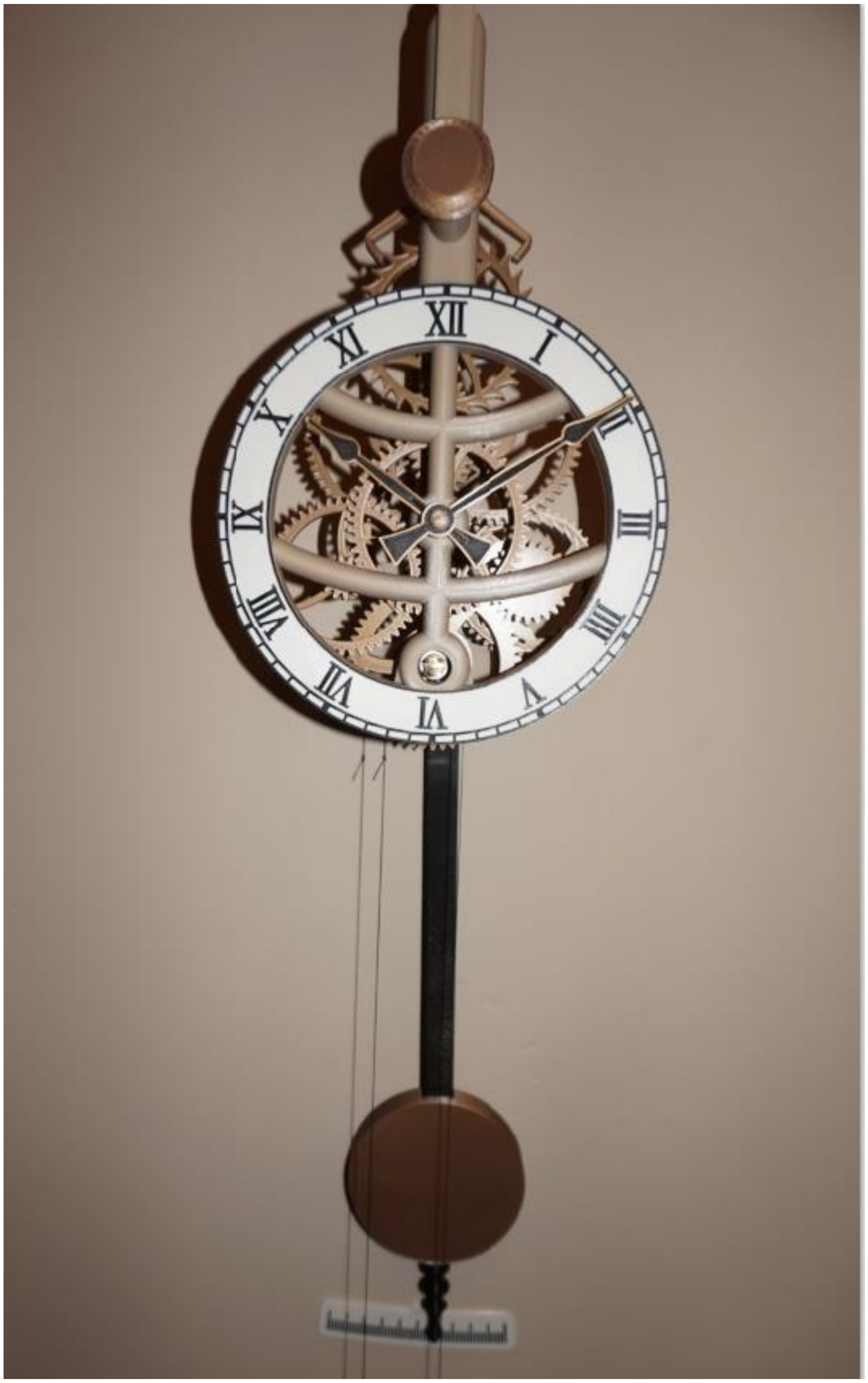


Figure 66: Original Thingiverse design

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

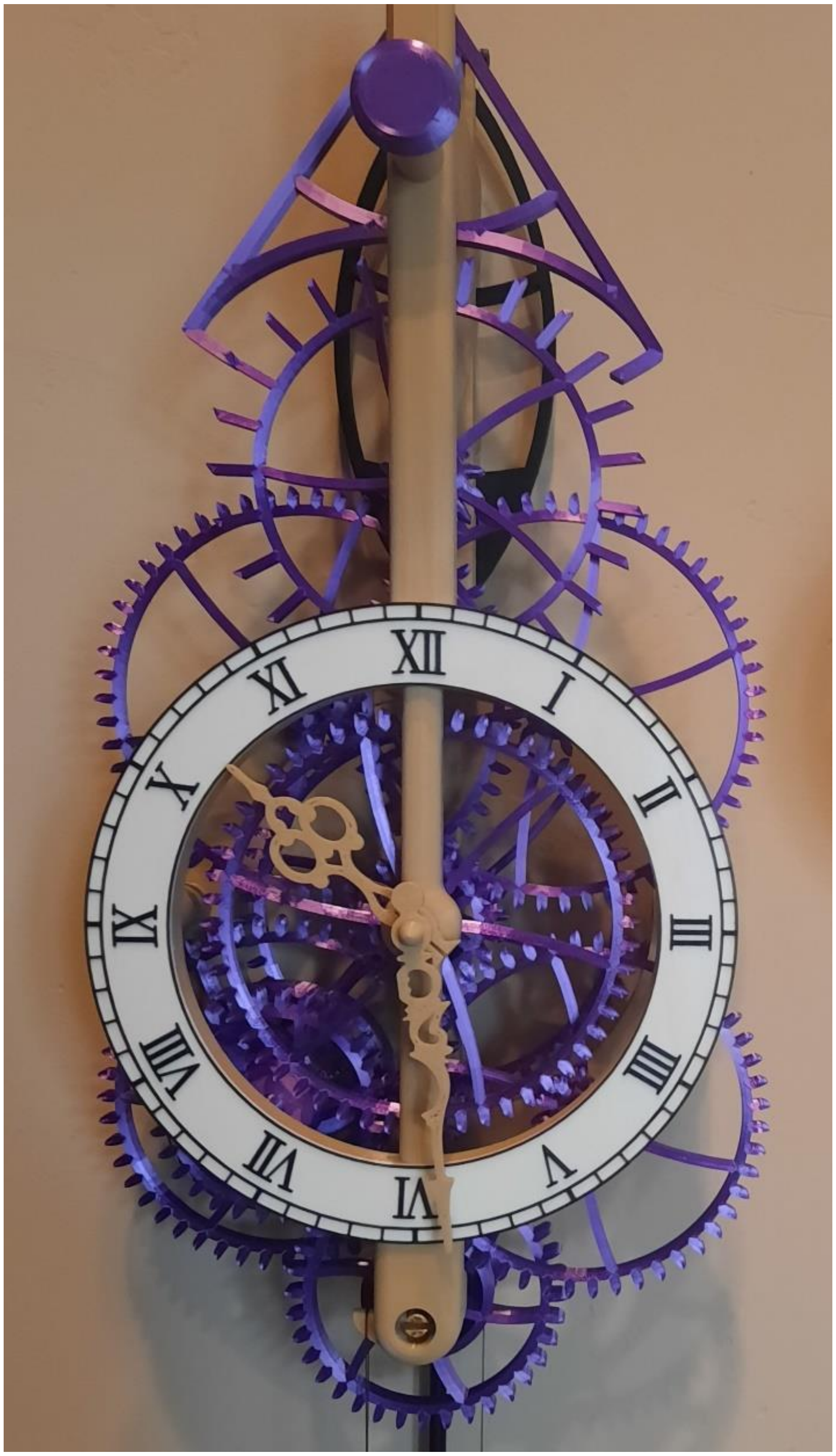


Figure 67: Large pendulum clock

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, while also making it one of my easiest to build.

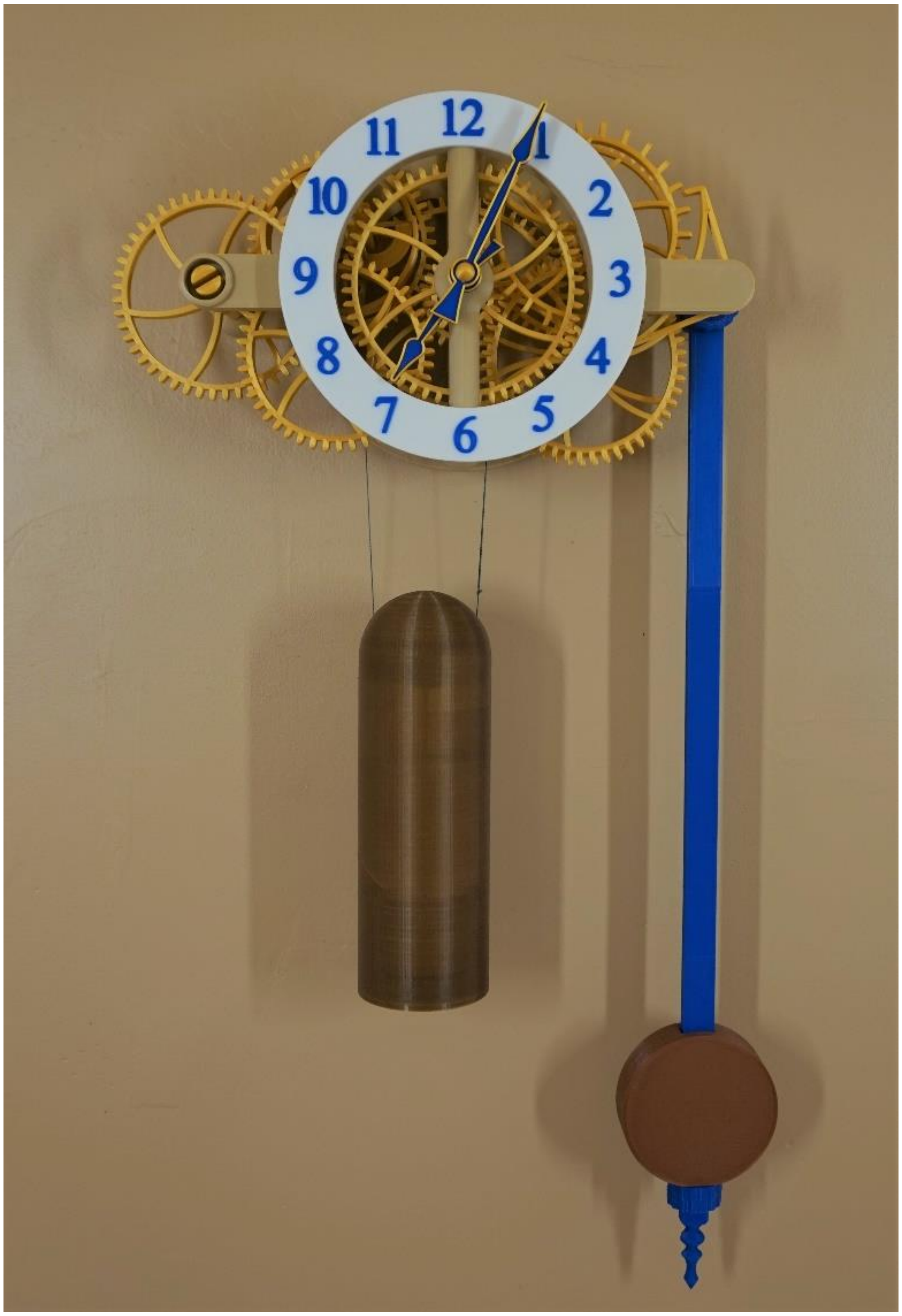


Figure 68: Easy build clock with 32 day runtime